Introducing Security Testing to Developers

Experiences and Lessons Learned

Achim D. Brucker （ブルッカー・アキム）

a.brucker@sheffield.ac.uk **https://www.brucker.ch/**

Software Assurance & Security Research
Department of Computer Science, The University of Sheffield, Sheffield, UK
https://logicalhacking.com/

Checkmarx Security Conference Tokyo 2017
実践アプリケーションセキュリティ
December 1st, 2017

{* LogicalHacking *}.com CHECKMARX 東陽テクニカ

Outline

1 About Me

2 Motivation

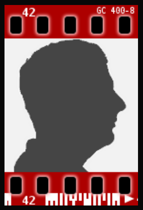
3 Secure Software Development

4 Enabling Developers: From (Mild) Pain to Success

5 Lesson's Learned

About Me

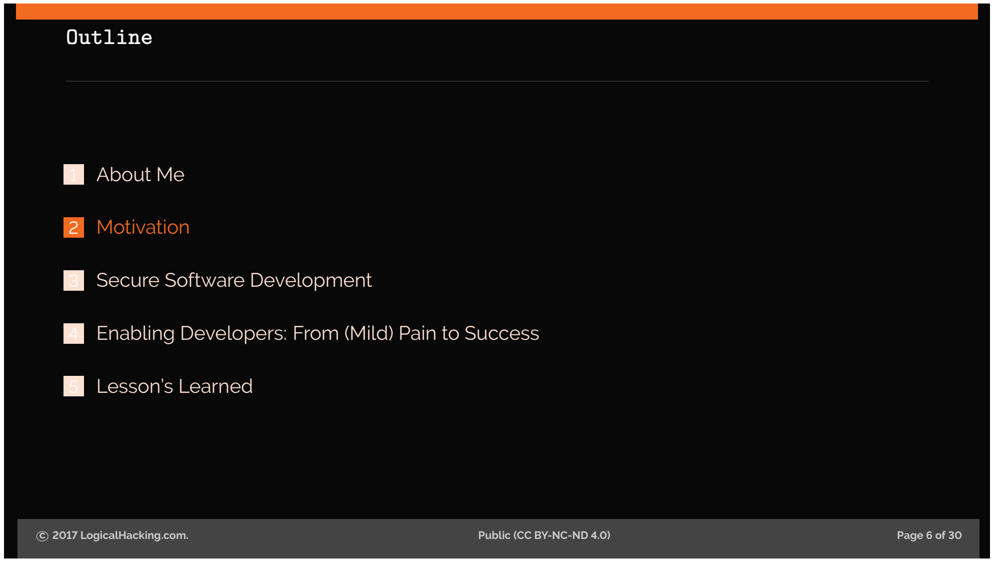
- PhD from ETH Zurich, Switzerland
- Eight year experience in secure enterprise software development:
  - Member of the central security team, SAP SE (Germany)
    - Security Testing Strategist
    - Security Research Expert/Architect
  - Work areas at SAP included:
    - Defining the risk-based Security Testing Strategy
    - Evaluation of security testing tools (e.g., SAST, DAST)
    - Roll-out of security testing tools
    - Identification of white spots and improvements of tools
    - Secure Software Development Life Cycle integration
    - Applied security research
- Since December 2015:
  - Associate Professor, The University of Sheffield, UK
  - Head of the Software Assurance & Security Research Team
  - Available as consultancy & (research) collaborations



https://www.brucker.ch/

SAP SE

- Leader in Business Software
  - Cloud
  - Mobile
  - On premise
- Many different technologies and platforms, e.g.,
  - In-memory database and application server (Hana)
  - Netweaver for ABAP and Java
- More than 25 industries
- 63% of the world's transaction revenue touches an SAP system
- Over 68 000 employees worldwide (over 25 000 software developers)
- Headquarters: Walldorf (Heidelberg), Germany



Outline

1. About Me
2. Motivation
3. Secure Software Development
4. Enabling Developers: From (Mild) Pain to Success
5. Lesson's Learned

Example (LinkedIn, May 2016)

- 164 million email addresses and passwords
- Data leaked in 2012, data sold in 2016
- Leaked Data
  - E-mail addresses
  - Passwords

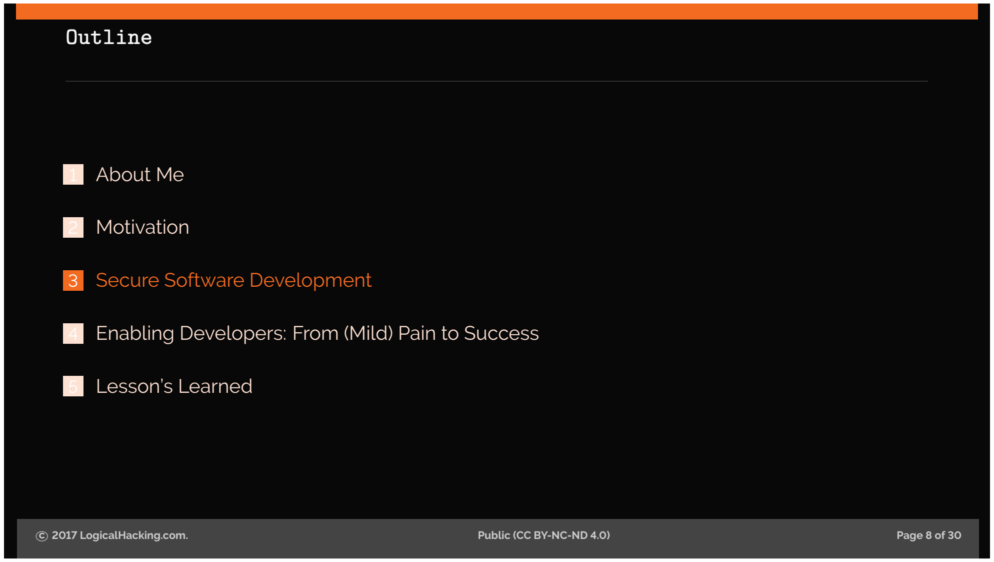
Example (TalkTalk, October 2015)

- nearly 157,000 customer records leaked
- nearly 16,000 records included bank details
- more than 150,000 customers lost
  (home services market share fall by 4.4 percent
  in terms of new customers)
- Costs for TalkTalk: ca. £60 million (ca. 90億円)

Example (Ashley Madison, July 2015)

- More than 30 million email addresses & much more
- Leaked data:
  - Date of birth
  - E-mail addresses
  - Ethnicities, Genders
  - Sexual preferences
  - Home addresses, Phone numbers
  - Payment histories
  - Passwords, usernames, security questions and answers
  - Website activity

Forbes
<?php
PLEX
vtech
TRAI
The Lord of the Rings Online
StarNet
Win7
Vista
XSplit
NAUGHTY AMERICA
SPY



Outline

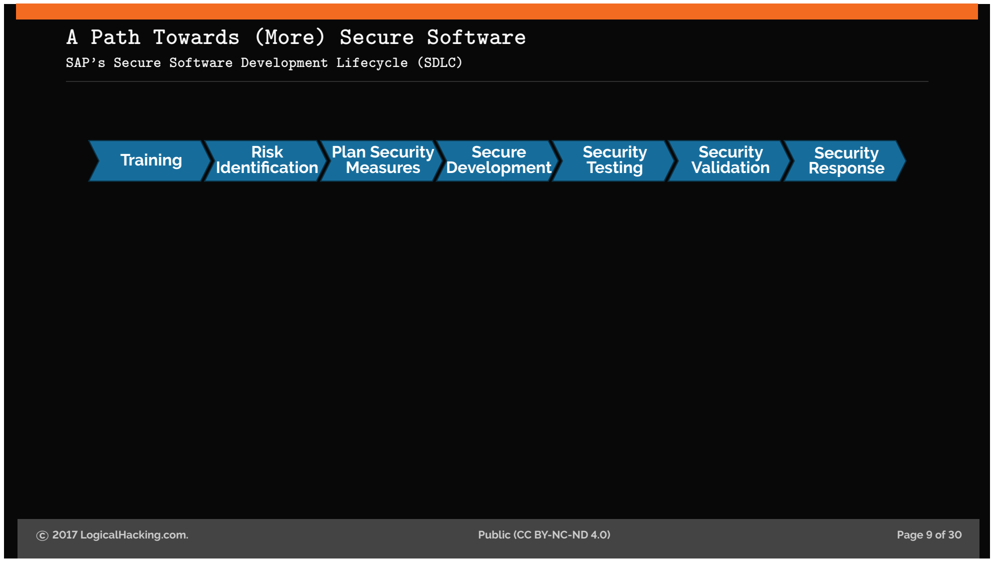
1 About Me

2 Motivation

3 Secure Software Development

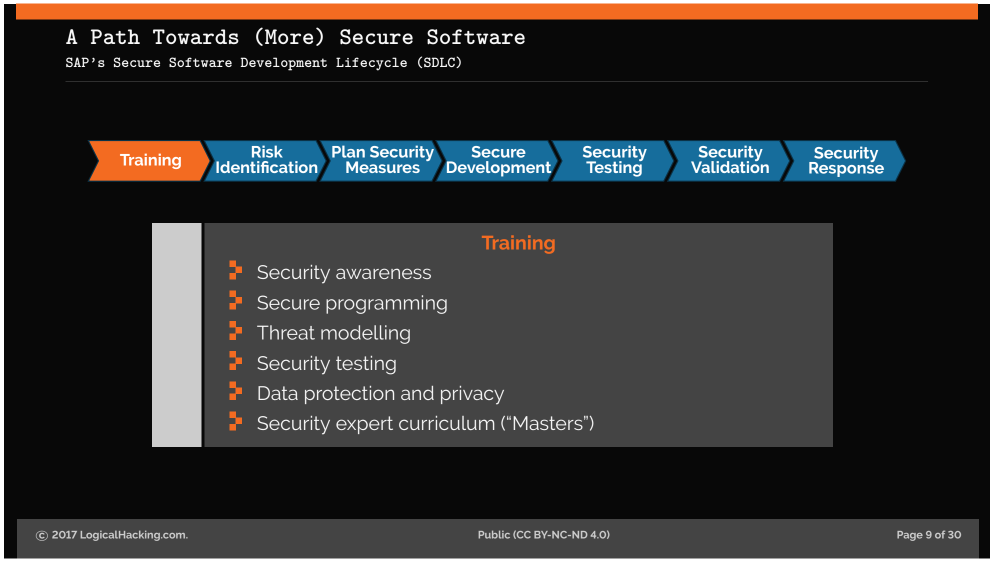
4 Enabling Developers: From (Mild) Pain to Success

5 Lesson's Learned



A Path Towards (More) Secure Software

SAP's Secure Software Development Lifecycle (SDLC)

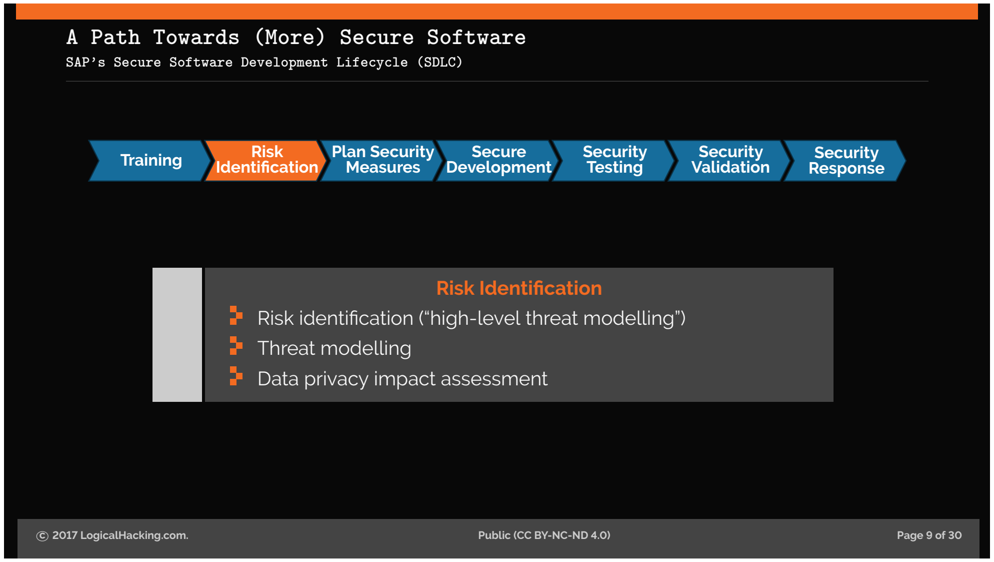


A Path Towards (More) Secure Software

SAP's Secure Software Development Lifecycle (SDLC)

Training

- Security awareness
- Secure programming
- Threat modelling
- Security testing
- Data protection and privacy
- Security expert curriculum ("Masters")



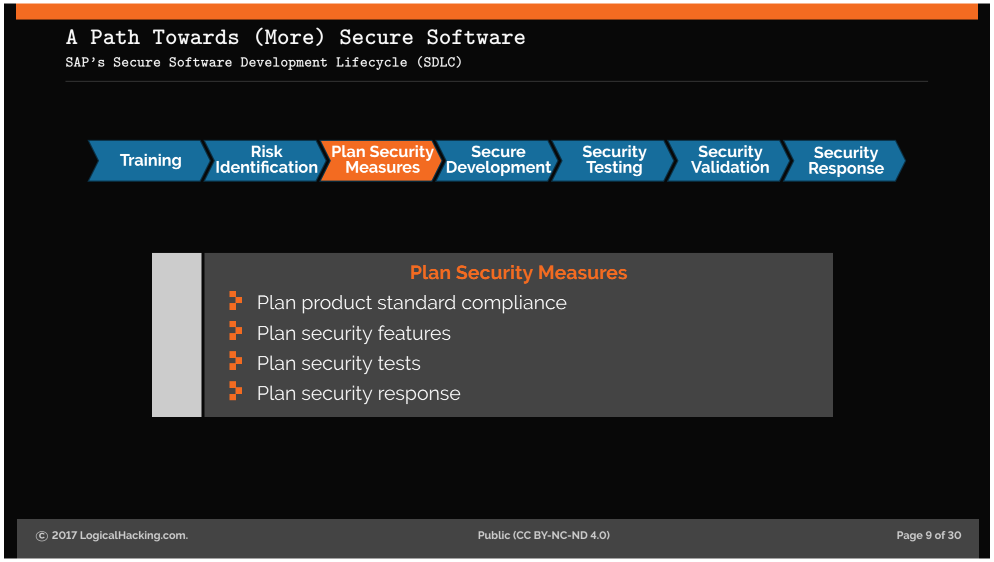
A Path Towards (More) Secure Software

SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Risk Identification

- Risk identification (“high-level threat modelling”)
- Threat modelling
- Data privacy impact assessment



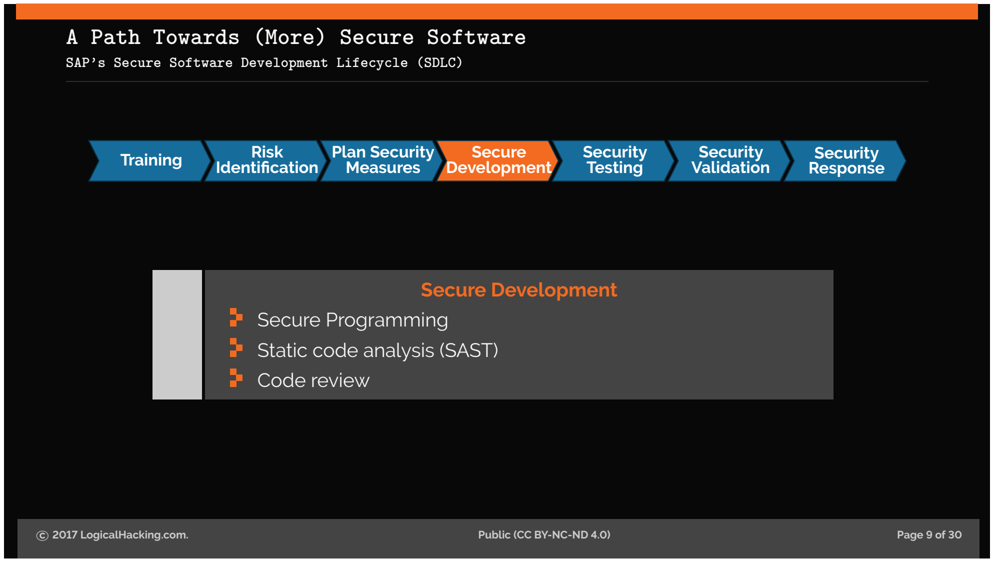
A Path Towards (More) Secure Software

SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Plan Security Measures

- Plan product standard compliance
- Plan security features
- Plan security tests
- Plan security response



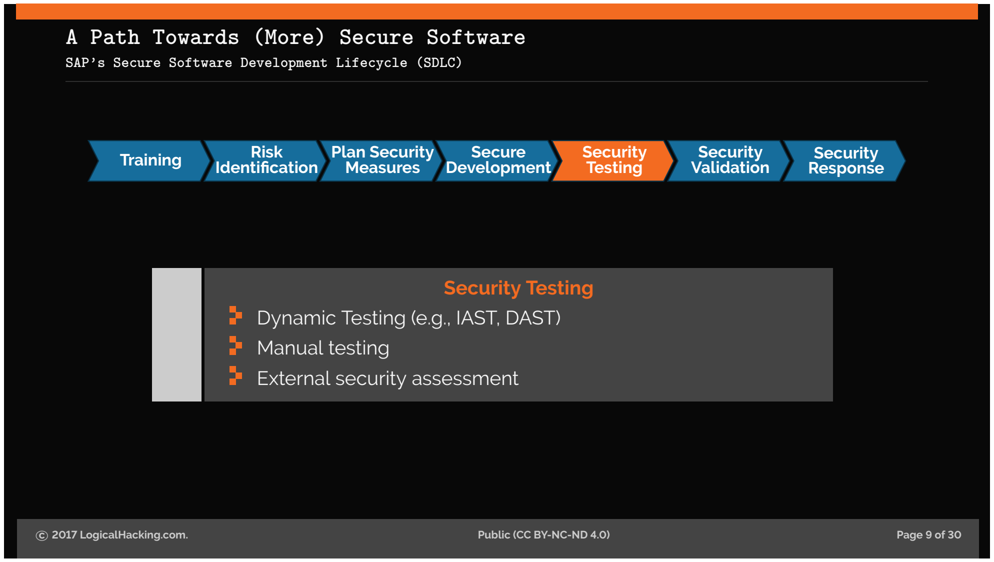
A Path Towards (More) Secure Software

SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Secure Development

- Secure Programming
- Static code analysis (SAST)
- Code review



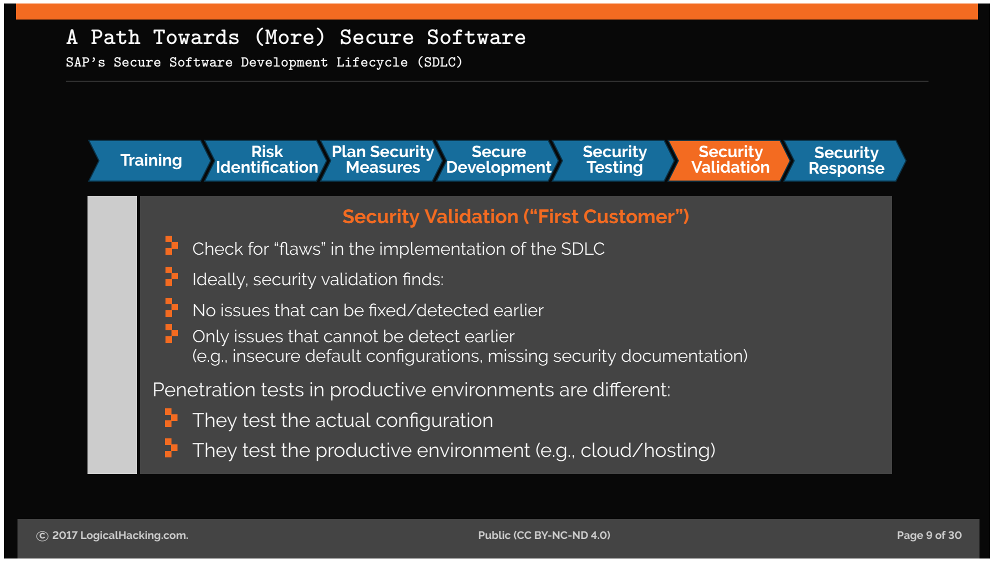
A Path Towards (More) Secure Software

SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Security Testing

- Dynamic Testing (e.g., IAST, DAST)
- Manual testing
- External security assessment



A Path Towards (More) Secure Software

SAP's Secure Software Development Lifecycle (SDLC)

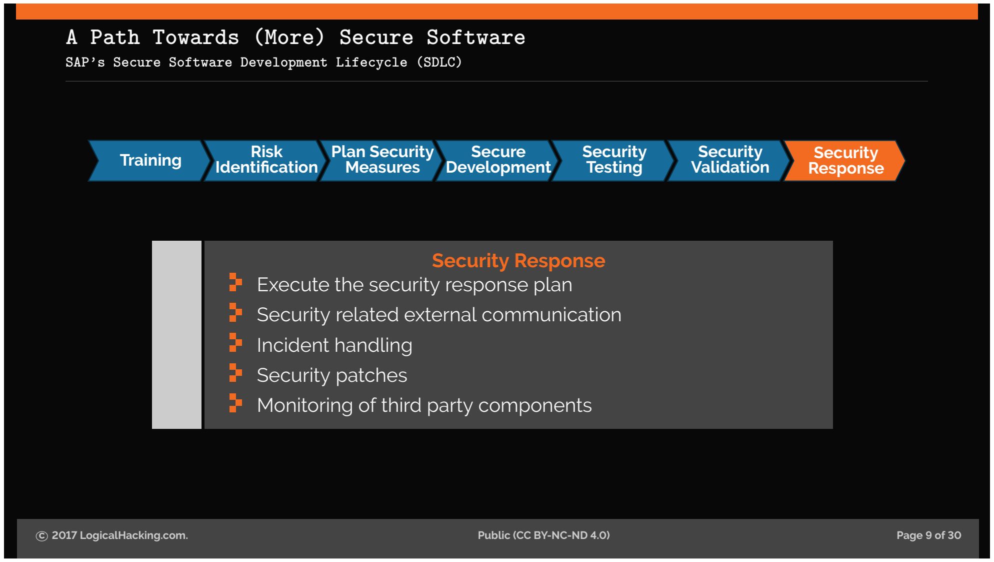
Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Security Validation ("First Customer")

- Check for "flaws" in the implementation of the SDLC
- Ideally, security validation finds:
- No issues that can be fixed/detected earlier
- Only issues that cannot be detect earlier
  (e.g., insecure default configurations, missing security documentation)

Penetration tests in productive environments are different:

- They test the actual configuration
- They test the productive environment (e.g., cloud/hosting)



A Path Towards (More) Secure Software

SAP's Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Security Response

- Execute the security response plan
- Security related external communication
- Incident handling
- Security patches
- Monitoring of third party components

A Path Towards (More) Secure Software

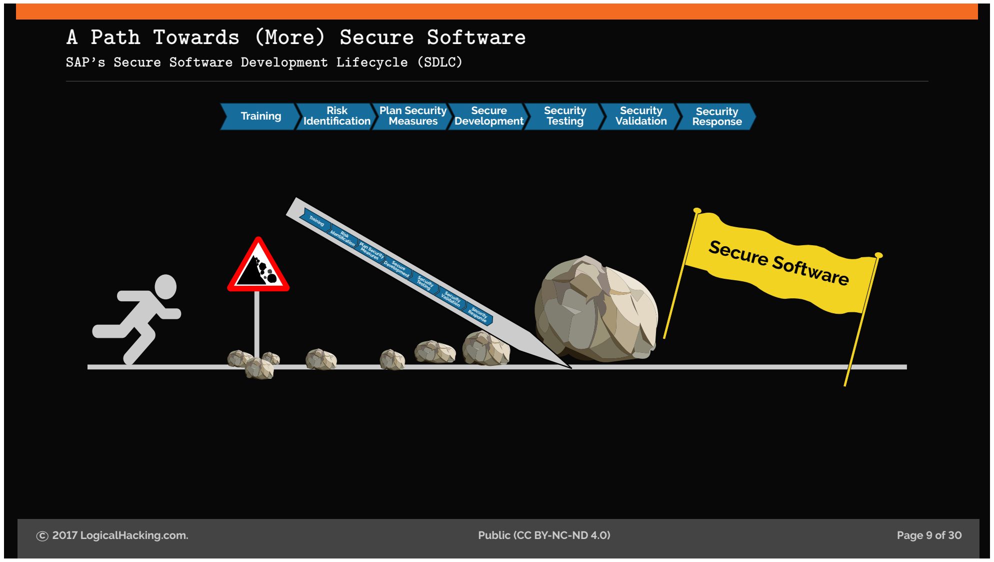
SAP's Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

A Path Towards (More) Secure Software

SAP's Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response



A Path Towards (More) Secure Software

SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

A Path Towards (More) Secure Software

SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Secure Software

© 2017 LogicalHacking.com. Public (CC BY-NC-ND 4.0) Page 9 of 30

A Path Towards (More) Secure Software

SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Secure Software

© 2017 LogicalHacking.com. Public (CC BY-NC-ND 4.0) Page 9 of 30

A Path Towards (More) Secure Software

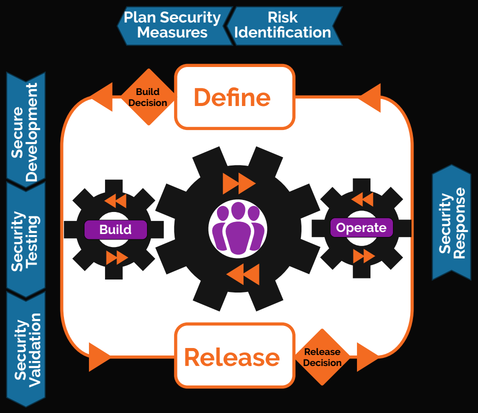
SAP’s Secure Software Development Lifecycle (SDLC)

Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Secure Software

© 2017 LogicalHacking.com. Public (CC BY-NC-ND 4.0) Page 9 of 30

Secure Software Development Lifecycle for Cloud/Agile



Plan Security Measures

Risk Identification

Secure Development

Security Testing

Security Validation

Build Decision

Define

Build

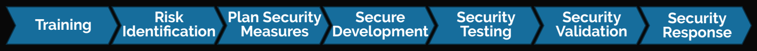
Operate

Release

Release Decision

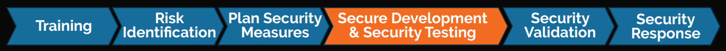
Security Response

Secure Software Lifecycle: My Vision



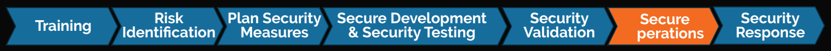
Training → Risk Identification → Plan Security Measures → Secure Development → Security Testing → Security Validation → Security Response

Secure Software Lifecycle: My Vision



Training → Risk Identification → Plan Security Measures → Secure Development & Security Testing → Security Validation → Security Response

Secure Software Lifecycle: My Vision

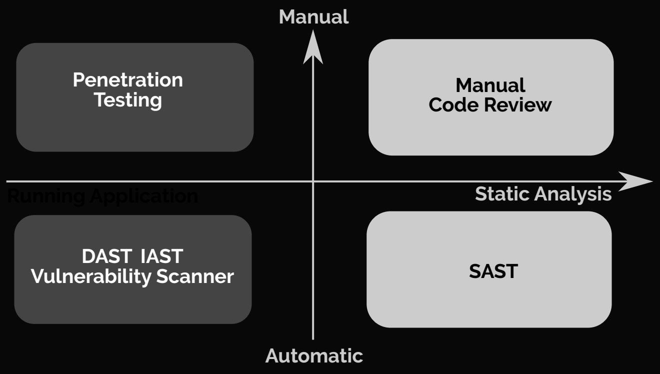


Training → Risk Identification → Plan Security Measures → Secure Development & Security Testing → Security Validation → Secure perations → Security Response

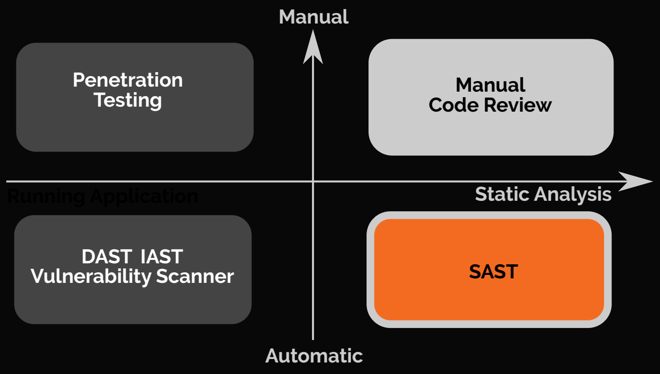
Outline

1. About Me
2. Motivation
3. Secure Software Development
4. Enabling Developers: From (Mild) Pain to Success
5. Lesson's Learned

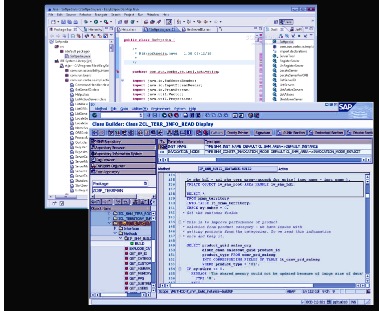
Finding Security Vulnerabilities



Finding Security Vulnerabilities



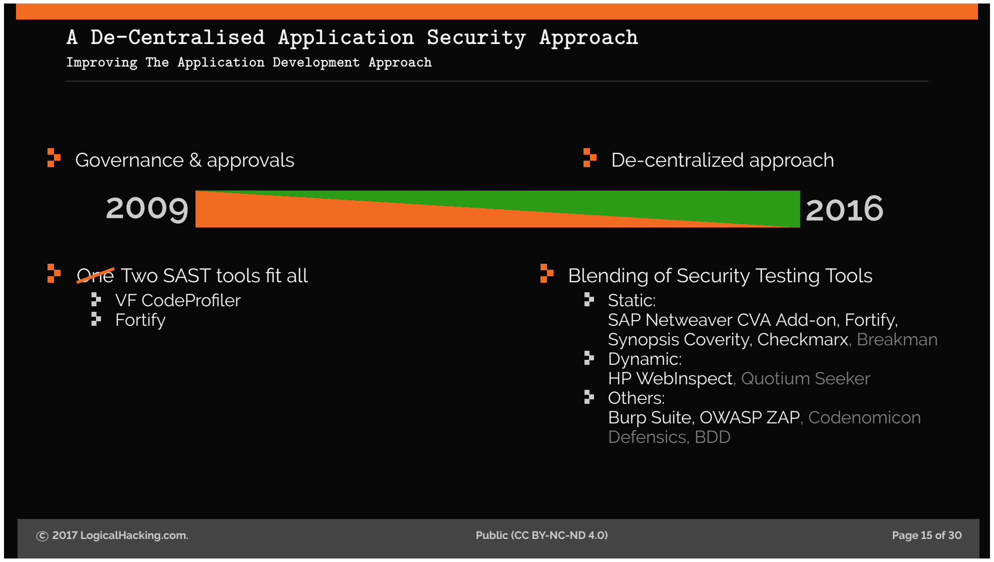
In 2010: Static Analysis Becomes Mandatory



SAST tools used:

Language	Tool	Vendor
ABAP Others	CodeProfiler Fortify	Virtual Forge HP

- Since 2010: SAST mandatory for all products
- Within two years, multiple billions lines analysed
- Constant improvement of tool configuration
- Further details:
  Deploying Static Application Security Testing on a Large Scale. In GI Sicherheit 2014. Lecture Notes in Informatics, 228, pages 91-101, GI, 2014.



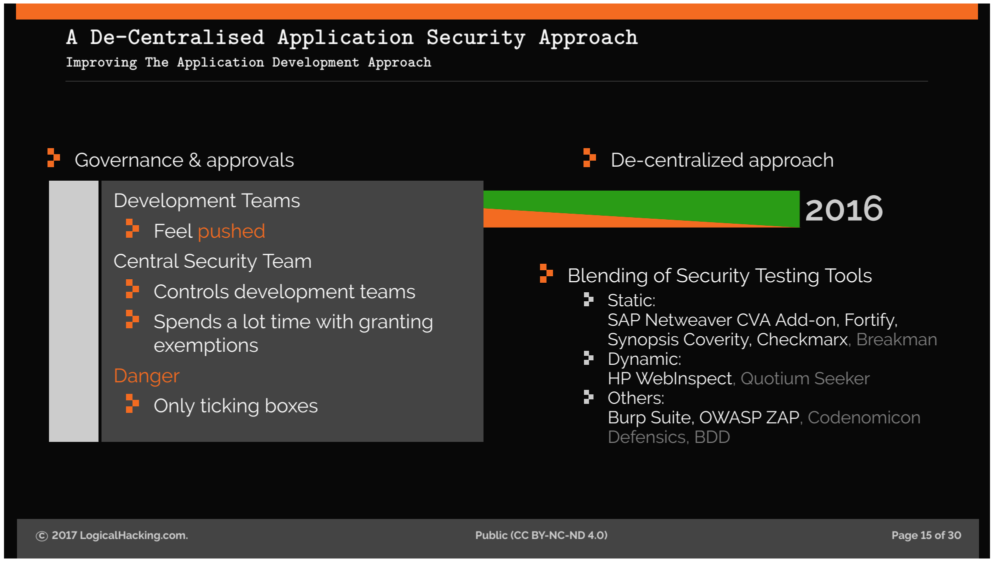
A De-Centralised Application Security Approach

Improving The Application Development Approach

- Governance & approvals
- De-centralized approach

2009 — 2016

- ~~One~~ Two SAST tools fit all
  - VF CodeProfiler
  - Fortify
- Blending of Security Testing Tools
  - Static:
    SAP Netweaver CVA Add-on, Fortify, Synopsis Coverity, Checkmarx, Breakman
  - Dynamic:
    HP WebInspect, Quotium Seeker
  - Others:
    Burp Suite, OWASP ZAP, Codenomicon Defensics, BDD



A De-Centralised Application Security Approach

Improving The Application Development Approach

- Governance & approvals
- De-centralized approach

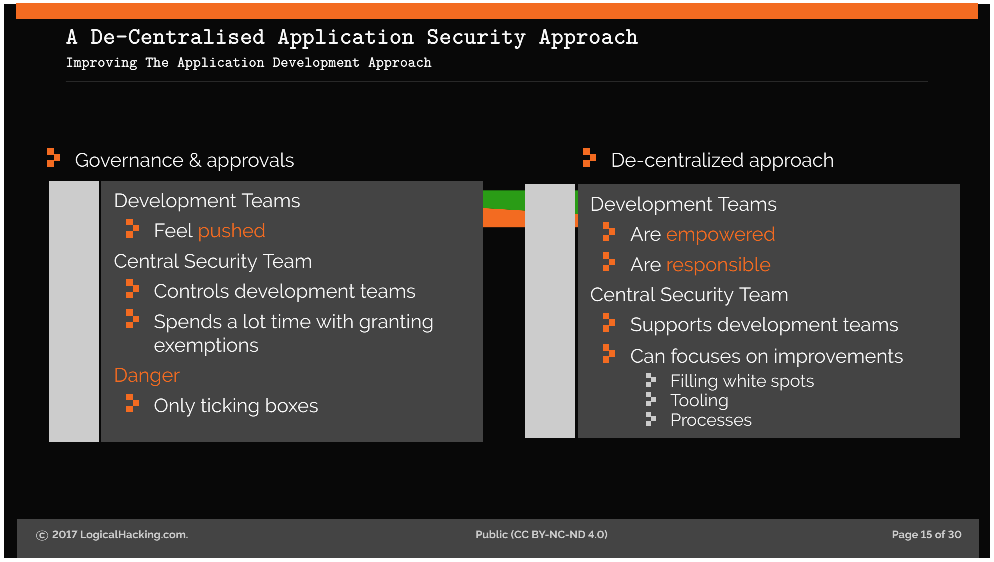
Development Teams
- Feel pushed

Central Security Team
- Controls development teams
- Spends a lot time with granting exemptions

Danger
- Only ticking boxes

2016

- Blending of Security Testing Tools
  - Static:
    SAP Netweaver CVA Add-on, Fortify, Synopsis Coverity, Checkmarx, Breakman
  - Dynamic:
    HP WebInspect, Quotium Seeker
  - Others:
    Burp Suite, OWASP ZAP, Codenomicon Defensics, BDD



A De-Centralised Application Security Approach

Improving The Application Development Approach

- Governance & approvals
- De-centralized approach

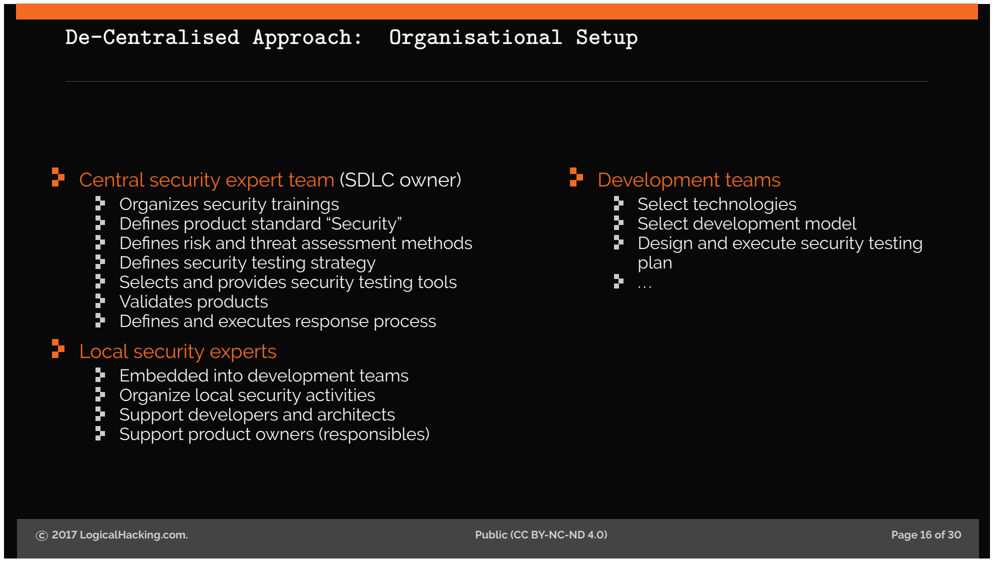
Development Teams
- Feel pushed

Central Security Team
- Controls development teams
- Spends a lot time with granting exemptions

Danger
- Only ticking boxes

Development Teams
- Are empowered
- Are responsible

Central Security Team
- Supports development teams
- Can focuses on improvements
  - Filling white spots
  - Tooling
  - Processes



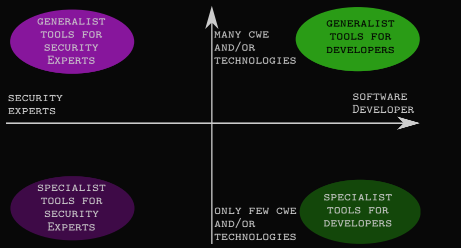
De-Centralised Approach: Organisational Setup

- Central security expert team (SDLC owner)
  - Organizes security trainings
  - Defines product standard "Security"
  - Defines risk and threat assessment methods
  - Defines security testing strategy
  - Selects and provides security testing tools
  - Validates products
  - Defines and executes response process
- Local security experts
  - Embedded into development teams
  - Organize local security activities
  - Support developers and architects
  - Support product owners (responsibles)
- Development teams
  - Select technologies
  - Select development model
  - Design and execute security testing plan
  - …

Security Team Focus: Security Testing for Developers

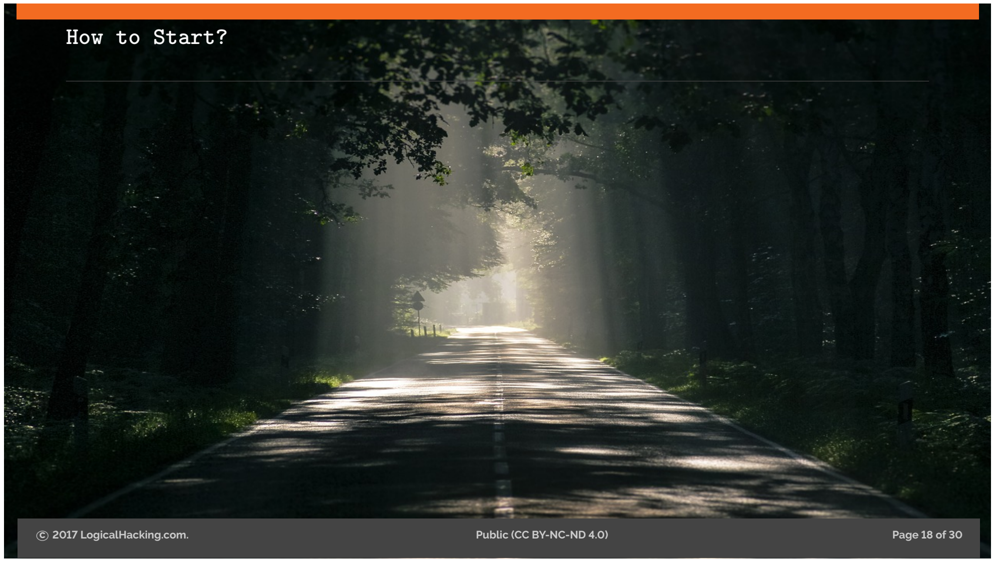
Security testing tools for developers, need to

- Be applicable from the start of development
- Automate the security knowledge
- Be integrated into dev world, e.g.,
  - IDE (instant feedback)
  - Continuous integration
- Provide easy to understand fix recommendations
- Declare their “sweet spots”


GENERALIST TOOLS FOR SECURITY EXPERTS
MANY CWE AND/OR TECHNOLOGIES
GENERALIST TOOLS FOR DEVELOPERS
SECURITY EXPERTS
SOFTWARE DEVELOPER
SPECIALIST TOOLS FOR SECURITY EXPERTS
ONLY FEW CWE AND/OR TECHNOLOGIES
SPECIALIST TOOLS FOR DEVELOPERS

https://logicalhacking.com/blog/2016/10/25/classifying-security-testing-tools/

How to Start?



Develop a Culture of Security Champions

- Make security interesting
  - Offer education/talks
  - Gamification
- Encourage (volunteers!) security champions
  - Do not force them, they should volunteer
  - Provide incentives
- Build a community
  - Organize knowledge transfer
  - Meet in person
- Empower your security champions
  - Trust their decisions
  - Include them decisions (selection of new tools, process changes, etc.)
- Each developer should know a security champion personally

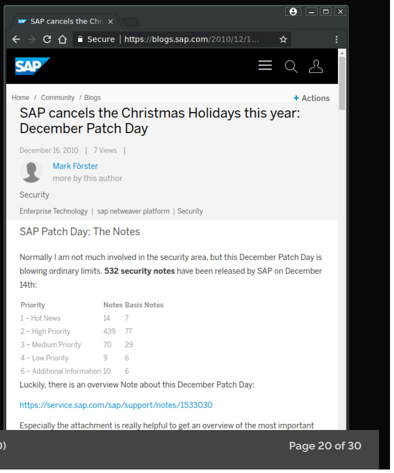
Start Slow, Grow and Improve Fast

Start slow:

- Start with a limited scope
  - Only one team
  - Only a subset of vulnerability types
  - Introduce only one tool at a time
- Focus first on newly developed code
  - but develop a plan for fixing old code as well

Grow and improve fast:

- Encourage teams to
  - share their success stories
  - to help each other
- Make tools available easily
  - Central budgeting
  - Integration into build/repository infrastructure



SAP cancels the Christmas Holidays this year: December Patch Day

December 16, 2010 | 7 Views |

Mark Förster
more by this author

Security

Enterprise Technology | sap netweaver platform | Security

SAP Patch Day: The Notes

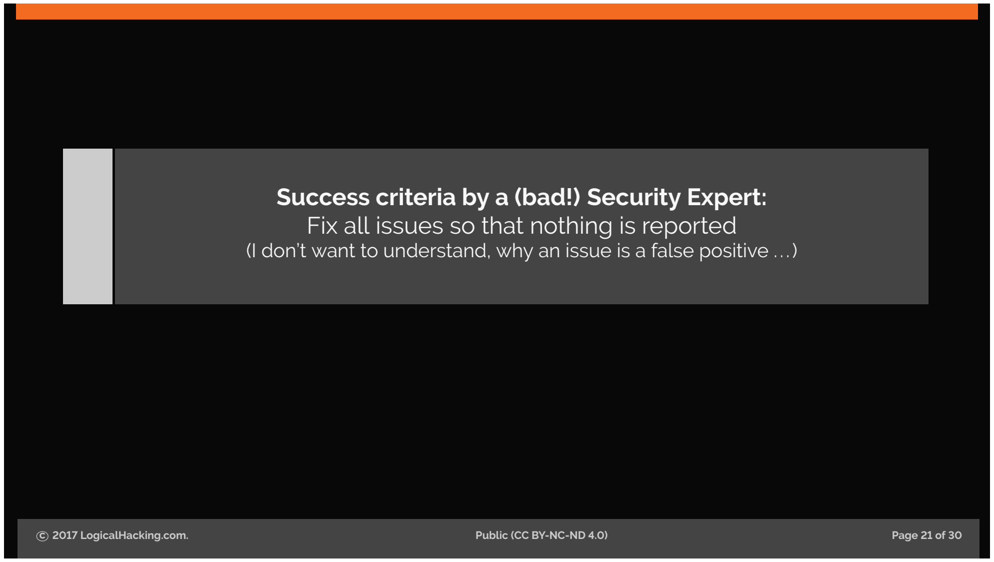
Normally I am not much involved in the security area, but this December Patch Day is blowing ordinary limits. **532 security notes** have been released by SAP on December 14th:

Priority	Notes	Basis Notes
1 – Hot News	14	7
2 – High Priority	439	77
3 – Medium Priority	70	29
4 – Low Priority	9	6
6 – Additional Information	10	6

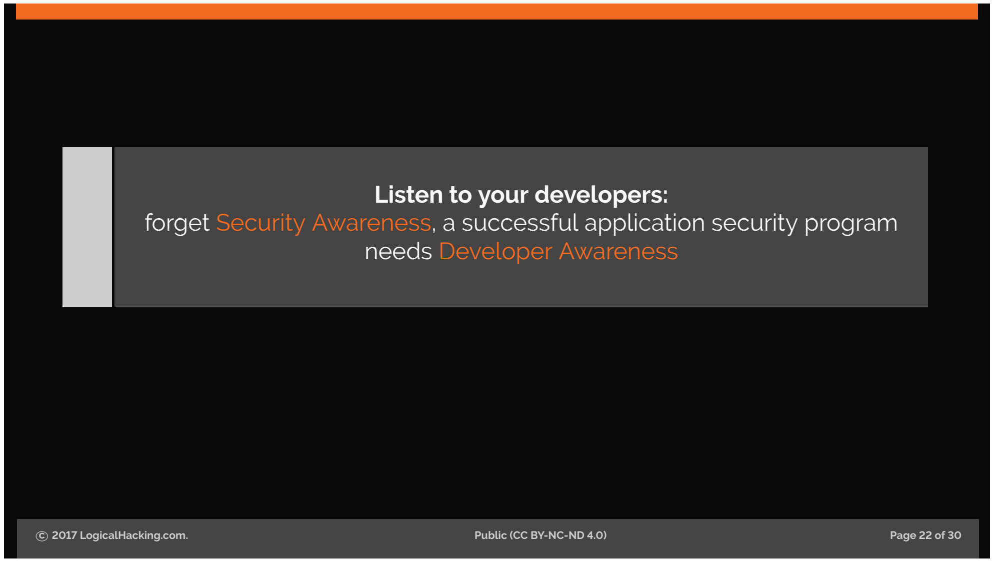
Luckily, there is an overview Note about this December Patch Day:

https://service.sap.com/sap/support/notes/1533030

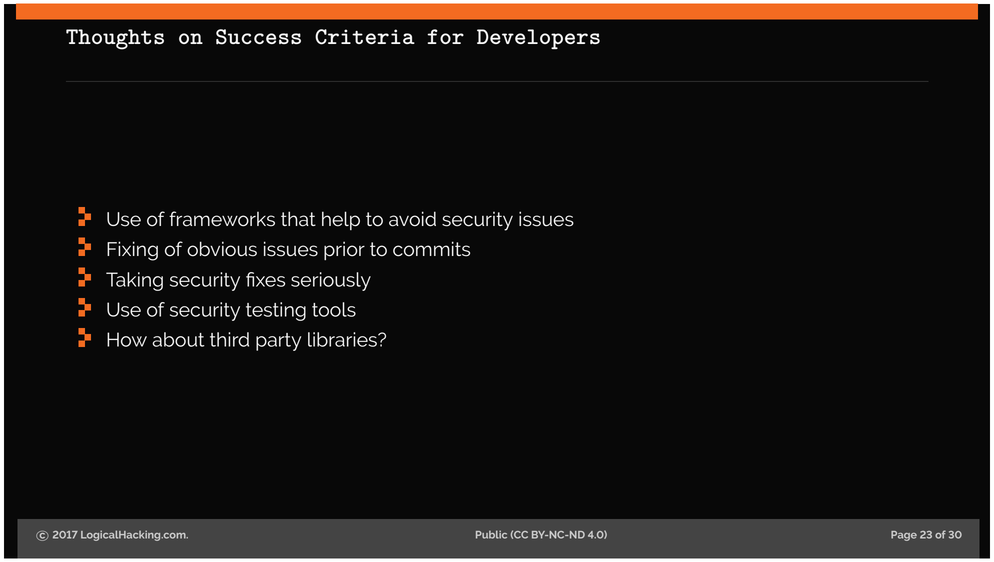
Especially the attachment is really helpful to get an overview of the most important



Success criteria by a (bad!) Security Expert:
Fix all issues so that nothing is reported
(I don't want to understand, why an issue is a false positive …)

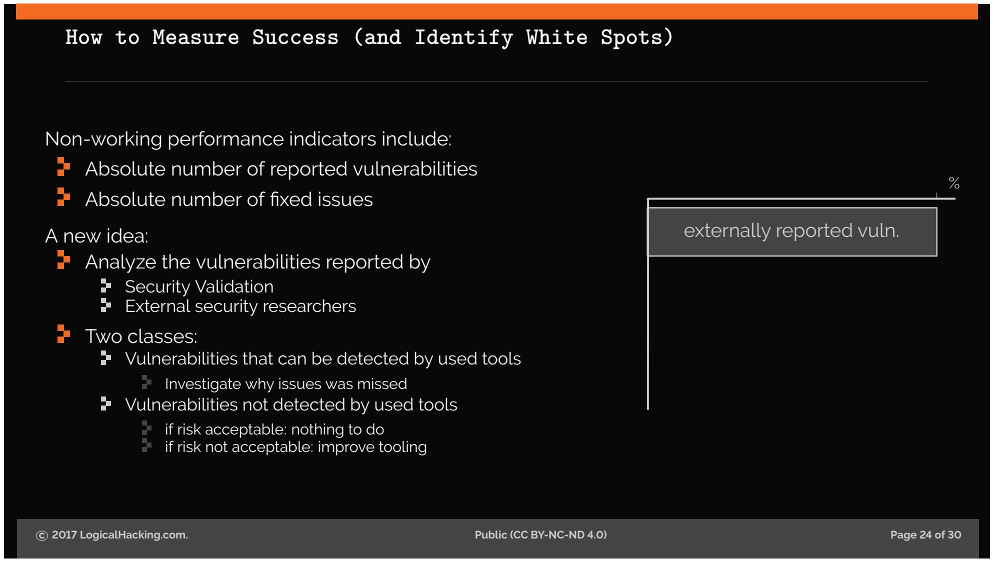


Listen to your developers:
forget Security Awareness, a successful application security program needs Developer Awareness



Thoughts on Success Criteria for Developers

- Use of frameworks that help to avoid security issues
- Fixing of obvious issues prior to commits
- Taking security fixes seriously
- Use of security testing tools
- How about third party libraries?



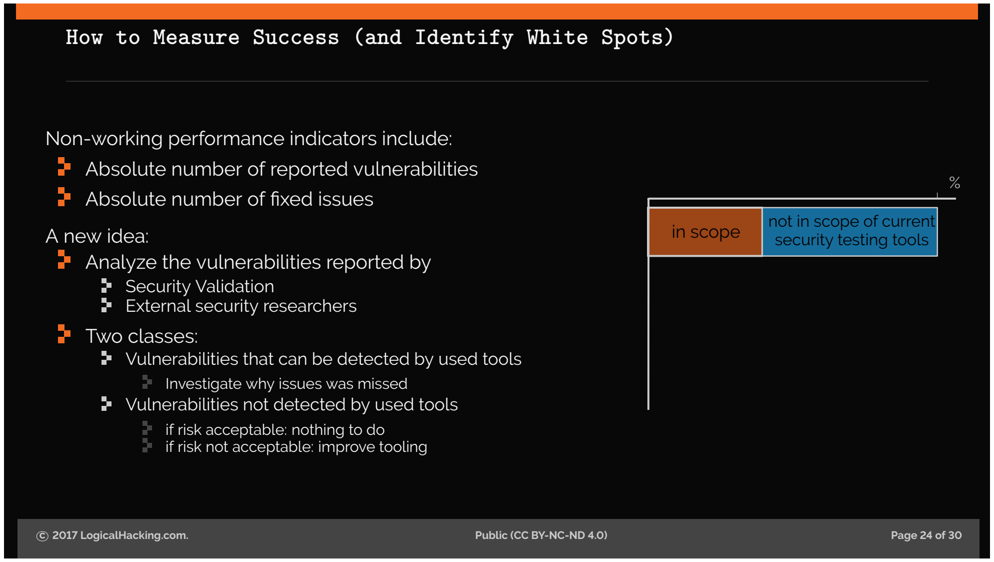
How to Measure Success (and Identify White Spots)

Non-working performance indicators include:

- Absolute number of reported vulnerabilities
- Absolute number of fixed issues

A new idea:

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Two classes:
  - Vulnerabilities that can be detected by used tools
    - Investigate why issues was missed
  - Vulnerabilities not detected by used tools
    - if risk acceptable: nothing to do
    - if risk not acceptable: improve tooling



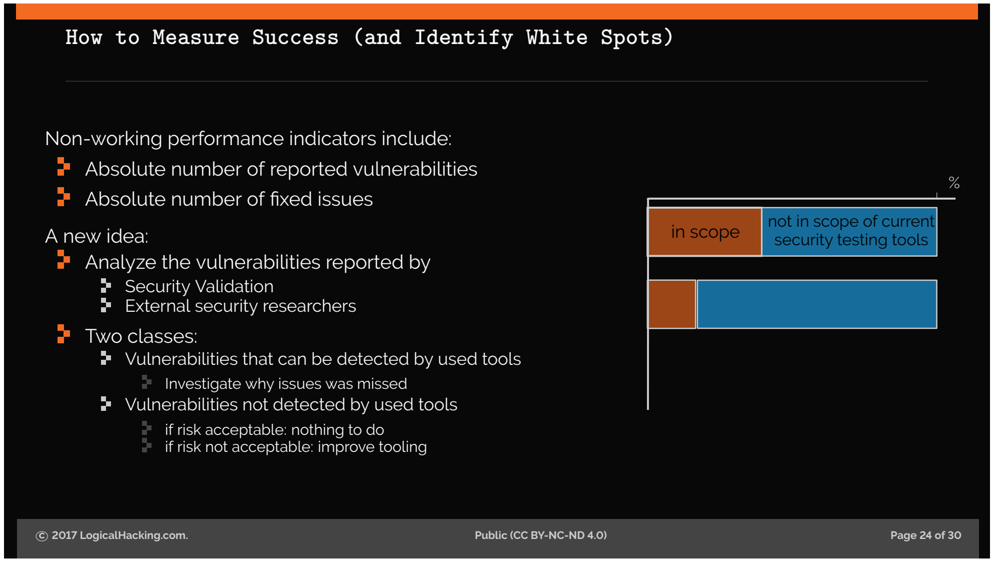
How to Measure Success (and Identify White Spots)

Non-working performance indicators include:

- Absolute number of reported vulnerabilities
- Absolute number of fixed issues

A new idea:

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Two classes:
  - Vulnerabilities that can be detected by used tools
    - Investigate why issues was missed
  - Vulnerabilities not detected by used tools
    - if risk acceptable: nothing to do
    - if risk not acceptable: improve tooling



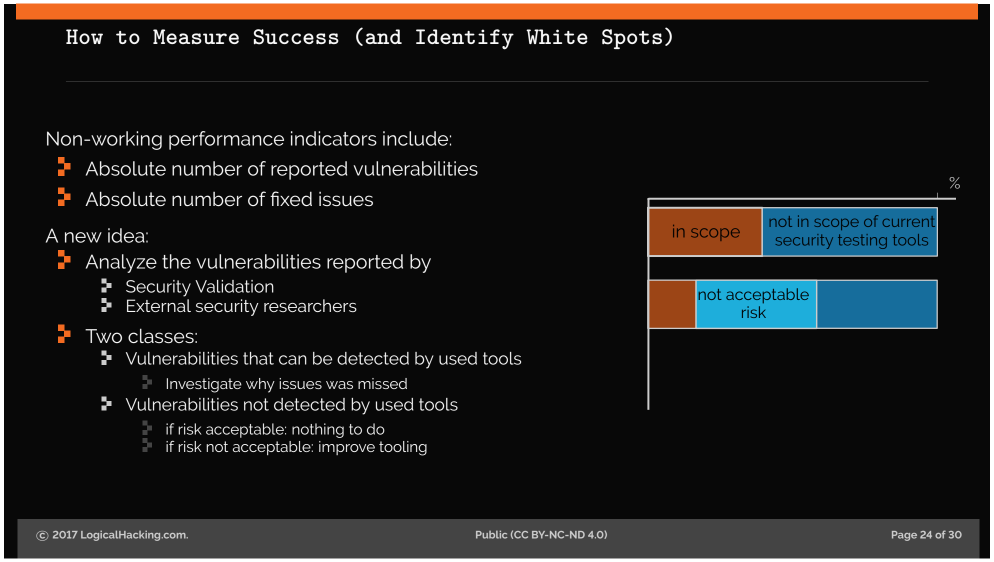
How to Measure Success (and Identify White Spots)

Non-working performance indicators include:

- Absolute number of reported vulnerabilities
- Absolute number of fixed issues

A new idea:

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Two classes:
  - Vulnerabilities that can be detected by used tools
    - Investigate why issues was missed
  - Vulnerabilities not detected by used tools
    - if risk acceptable: nothing to do
    - if risk not acceptable: improve tooling



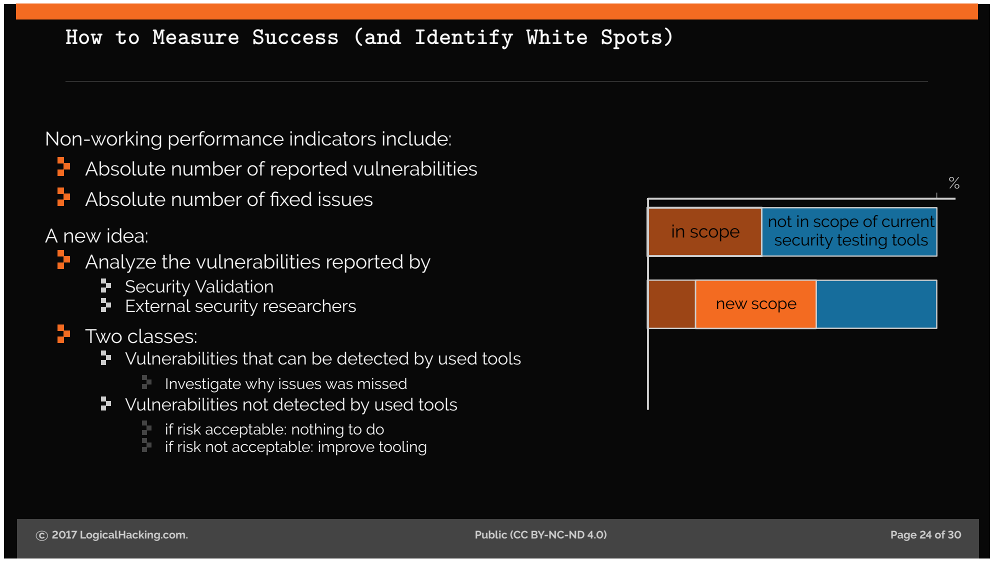
How to Measure Success (and Identify White Spots)

Non-working performance indicators include:

- Absolute number of reported vulnerabilities
- Absolute number of fixed issues

A new idea:

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Two classes:
  - Vulnerabilities that can be detected by used tools
    - Investigate why issues was missed
  - Vulnerabilities not detected by used tools
    - if risk acceptable: nothing to do
    - if risk not acceptable: improve tooling



How to Measure Success (and Identify White Spots)

Non-working performance indicators include:

- Absolute number of reported vulnerabilities
- Absolute number of fixed issues

A new idea:

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Two classes:
  - Vulnerabilities that can be detected by used tools
    - Investigate why issues was missed
  - Vulnerabilities not detected by used tools
    - if risk acceptable: nothing to do
    - if risk not acceptable: improve tooling

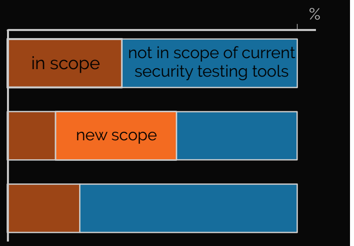
How to Measure Success (and Identify White Spots)

Non-working performance indicators include:

- Absolute number of reported vulnerabilities
- Absolute number of fixed issues

A new idea:

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Two classes:
  - Vulnerabilities that can be detected by used tools
    - Investigate why issues was missed
  - Vulnerabilities not detected by used tools
    - if risk acceptable: nothing to do
    - if risk not acceptable: improve tooling



How to Measure Success (and Identify White Spots)

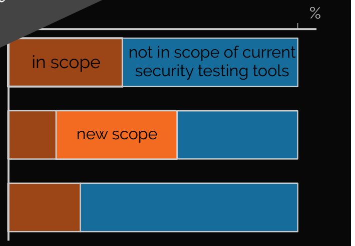
Non-working performance indicators include:

- Absolute number of reported vulnerabilities
- Absolute number of fixed issues

A new idea:

- Analyze the vulnerabilities r
  - Security Validation
  - External secu
- Two class
  - detected by used tools
    - issues was missed
  - s not detected by used tools
    - isk acceptable: nothing to do
    - if risk not acceptable: improve tooling

"Success criteria:" Percentage of vulnerabilities not covered by currently used security testing tools increases, i.e., the used tools are used effectively!



Outline

1 About Me

2 Motivation

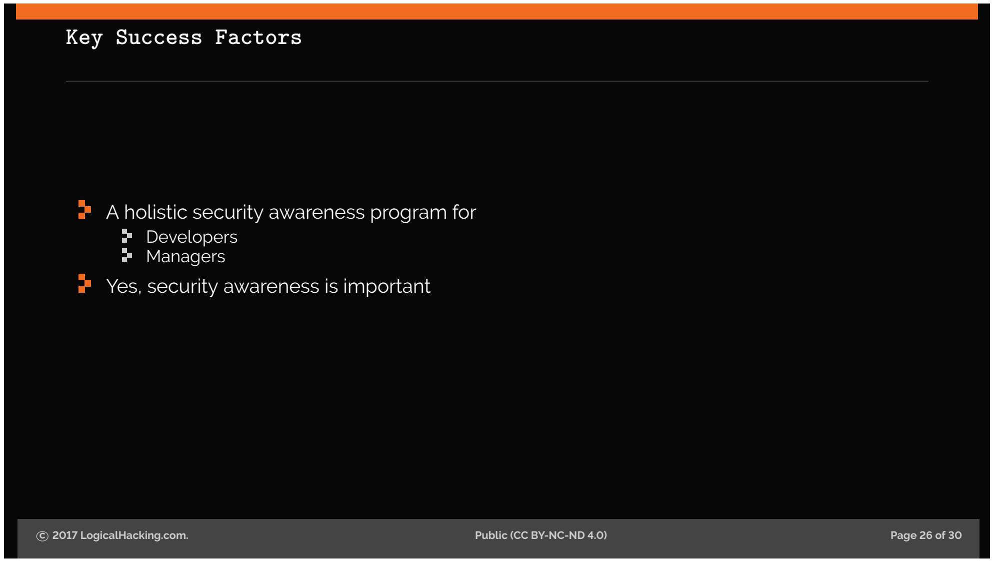
3 Secure Software Development

4 Enabling Developers: From (Mild) Pain to Success

5 Lesson's Learned

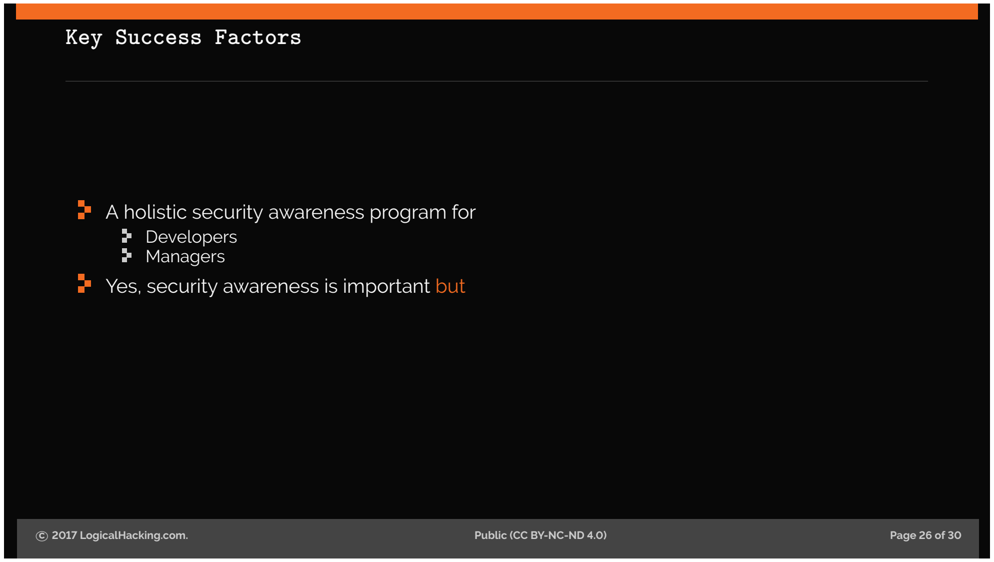
Key Success Factors

- A holistic security awareness program for
  - Developers
  - Managers



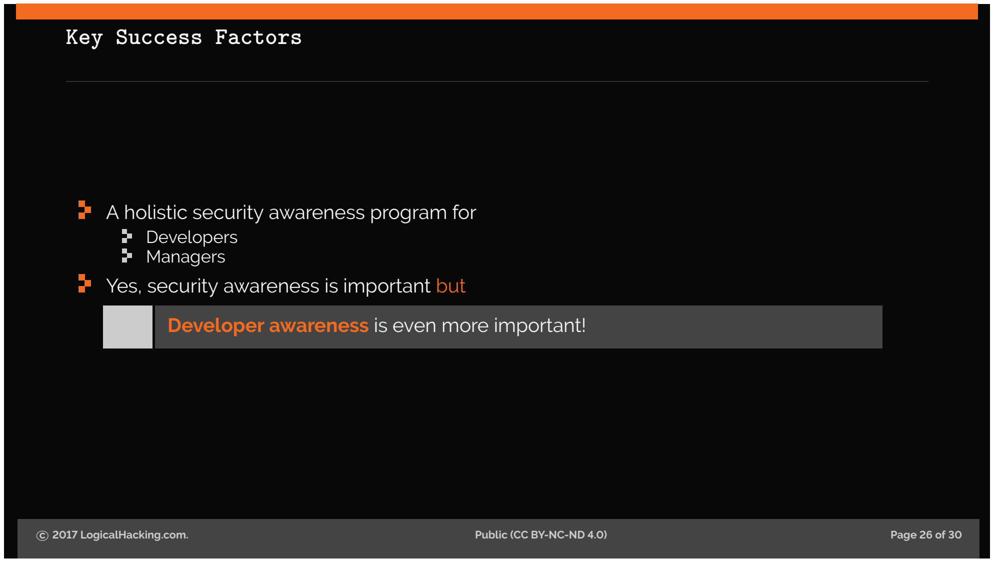
Key Success Factors

- A holistic security awareness program for
  - Developers
  - Managers
- Yes, security awareness is important



Key Success Factors

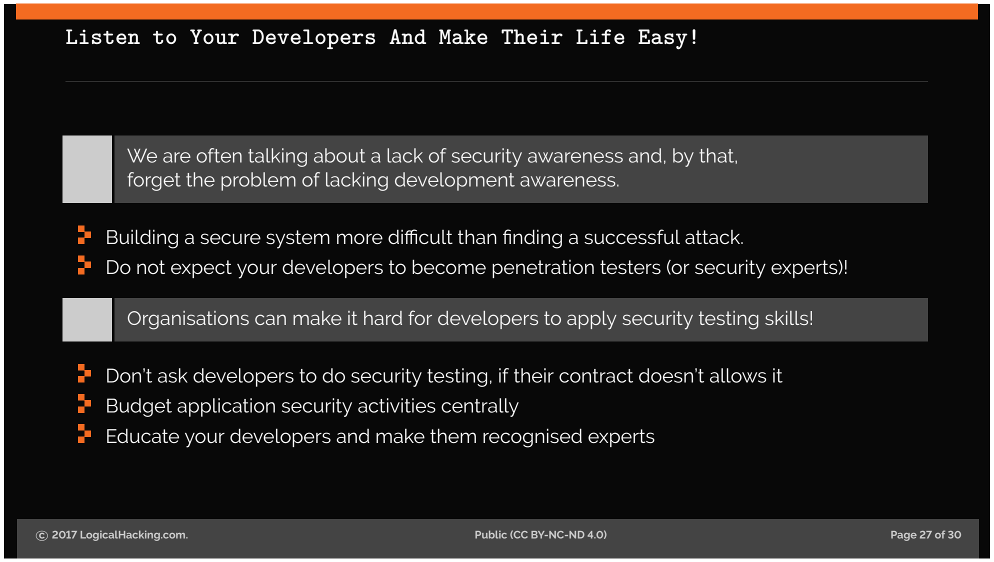
- A holistic security awareness program for
  - Developers
  - Managers
- Yes, security awareness is important but



Key Success Factors

- A holistic security awareness program for
  - Developers
  - Managers
- Yes, security awareness is important but

> **Developer awareness** is even more important!



Listen to Your Developers And Make Their Life Easy!

> We are often talking about a lack of security awareness and, by that, forget the problem of lacking development awareness.

- Building a secure system more difficult than finding a successful attack.
- Do not expect your developers to become penetration testers (or security experts)!

> Organisations can make it hard for developers to apply security testing skills!

- Don't ask developers to do security testing, if their contract doesn't allows it
- Budget application security activities centrally
- Educate your developers and make them recognised experts

Recommendations for Selecting Security Testing Tools

Select tools that are

- easy to integrate into your development process and tools
  - central scan infrastructure
  - source code upload, CLI, Jenkins, github, …
- easy to use by developers
  - easy to understand descriptions of findings
  - actionable fix recommendations
  - integrates teaching
- easy to adapt to your security policies and prioritisation
  - report issues that are relevant for you
  - focus developers effort on the issues that are critical for you
- allow for tracking your success
  - tool internal reporting
  - interfaces to your own reporting infrastructure

Final Remarks

What works well:

- Delegate power **and** accountability to development teams
- Multi-tiered model of security experts:
  - local experts for the local implementation of secure development
  - global experts that support the local security experts (champions):
    - act as consultant in difficult/non-standard situations
    - evaluate, purchase, and operate widely used security testing tools
    - can mediate between development teams and response teams
- Strict separation of
  - security testing supporting developers and
  - security validation

What does not work well:

- Forcing tools, processes, etc. on developers
- Penetration testing as "secure development" approach
  - Penetration has its value (e.g., as security integration test)

ご清聴ありがとうございました。

Contact:

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

a.brucker@sheffield.ac.uk
@adbrucker
https://de.linkedin.com/in/adbrucker/
https://www.brucker.ch/
https://logicalhacking.com/blog/

Bibliography

Ruediger Bachmann and Achim D. Brucker.
Developing secure software: A holistic approach to security testing.
Datenschutz und Datensicherheit (DuD), 38(4):257–261, April 2014.

Achim D. Brucker and Uwe Sodan.
Deploying static application security testing on a large scale.
In Stefan Katzenbeisser, Volkmar Lotz, and Edgar Weippl, editors, GI *Sicherheit 2014*, volume 228 of *Lecture Notes in Informatics*, pages 91–101. GI, March 2014.

Michael Felderer, Matthias Büchler, Martin Johns, Achim D. Brucker, Ruth Breu, and Alexander Pretschner.
Security testing: A survey.
Advances in Computers, 101:1–51, March 2016.

Document Classification and License Information