Deploying SAST on a Large Scale

Achim D. Brucker
joint work with Uwe Sodan
{achim.brucker, uwe.sodan}@sap.com

SAP AG, Germany

Has Sony been Hacked this Week?

http://hassonybeenhackedthisweek.com/

Time-line of the Sony Hack(s) (excerpt):

2011-04-20 Sony PSN goes down

2011-05-21 Sony BMG Greece: data 8300 users (SQL Injection)

2011-05-23 Sony Japanese database leaked (SQL Injection)

2011-05-24 Sony Canada: roughly 2,000 leaked (SQL Injection)

2011-06-05 Sony Pictures Russia (SQL Injection)

2011-06-06 Sony Portugal: SQL injection, iFrame injection and XSS

2011-06-20 20th breach within 2 months
177k email addresses were grabbed via a SQL injection

(http://hassonybeenhackedthisweek.com/history)

Consequences:

- account data of close to 100 million individuals exposed
- over 12 million credit and debit cards compromised
- more than 55 class-action lawsuits
- costs of $ 170 million only in 2011

- **Assume an SQL Statement for**

```
selecting all users with “userName” from table “user”
```

A Bluffers Guide to SQL Injection

- **Assume an SQL Statement for**

```
stmt = "SELECT * FROM ‘users‘ WHERE ‘name‘ = ’" + userName + "’;"
```

A Bluffers Guide to SQL Injection

- **Assume an SQL Statement for**

```
stmt = "SELECT * FROM ‘users‘ WHERE ‘name‘ = ’" + userName + "’;"
```

- **What happens if we choose the following *userName*:**

```
userName = "’ or ’1’=’1"
```

A Bluffers Guide to SQL Injection

- **Assume an SQL Statement for**

```
stmt = "SELECT * FROM ‘users‘ WHERE ‘name‘ = ’" + userName + "’;"
```

- **What happens if we choose the following *userName*:**

```
userName = "’ or ’1’=’1"
```

- **Resulting in the following statement:**

```
stmt = "SELECT * FROM ‘users‘ WHERE ‘name‘ = ’’ or ’1’=’1’;"
```

A Bluffers Guide to SQL Injection

- **Assume an SQL Statement for**

```
stmt = "SELECT * FROM ‘users‘ WHERE ‘name‘ = ’" + userName + "’;"
```

- **What happens if we choose the following *userName*:**

```
userName = "’ or ’1’=’1"
```

- **Resulting in the following statement:**

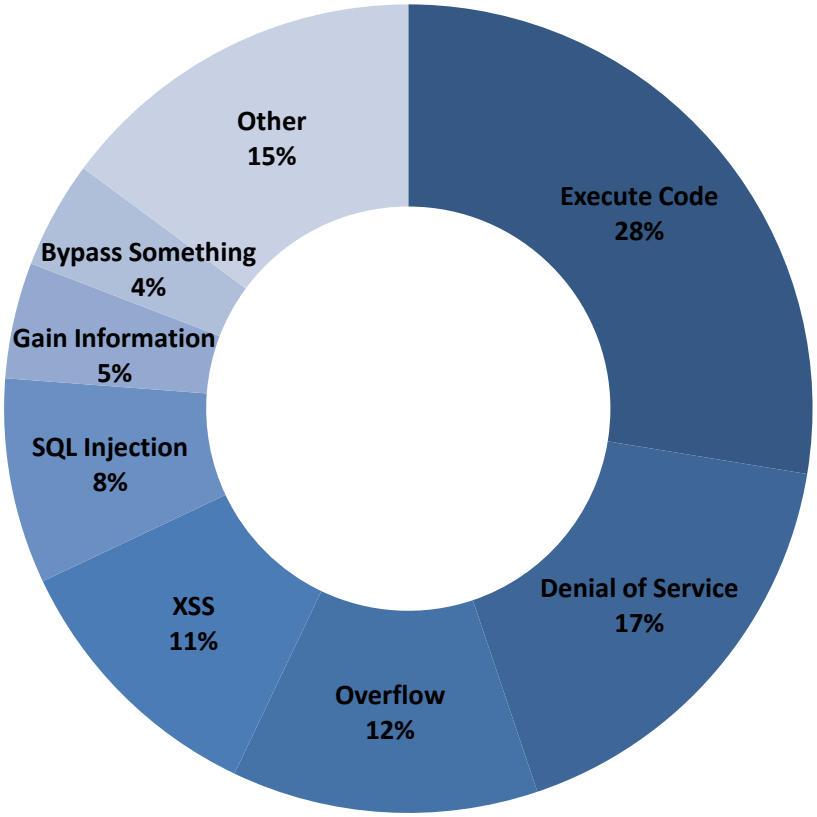
```
stmt = "SELECT * FROM ‘users‘ WHERE ‘name‘ = ’’ or ’1’=’1’;"
```

- **Which is equivalent to**

```
stmt = "SELECT * FROM ‘users‘;"
```

selecting the information of **all users** stored in the table ‘users’!

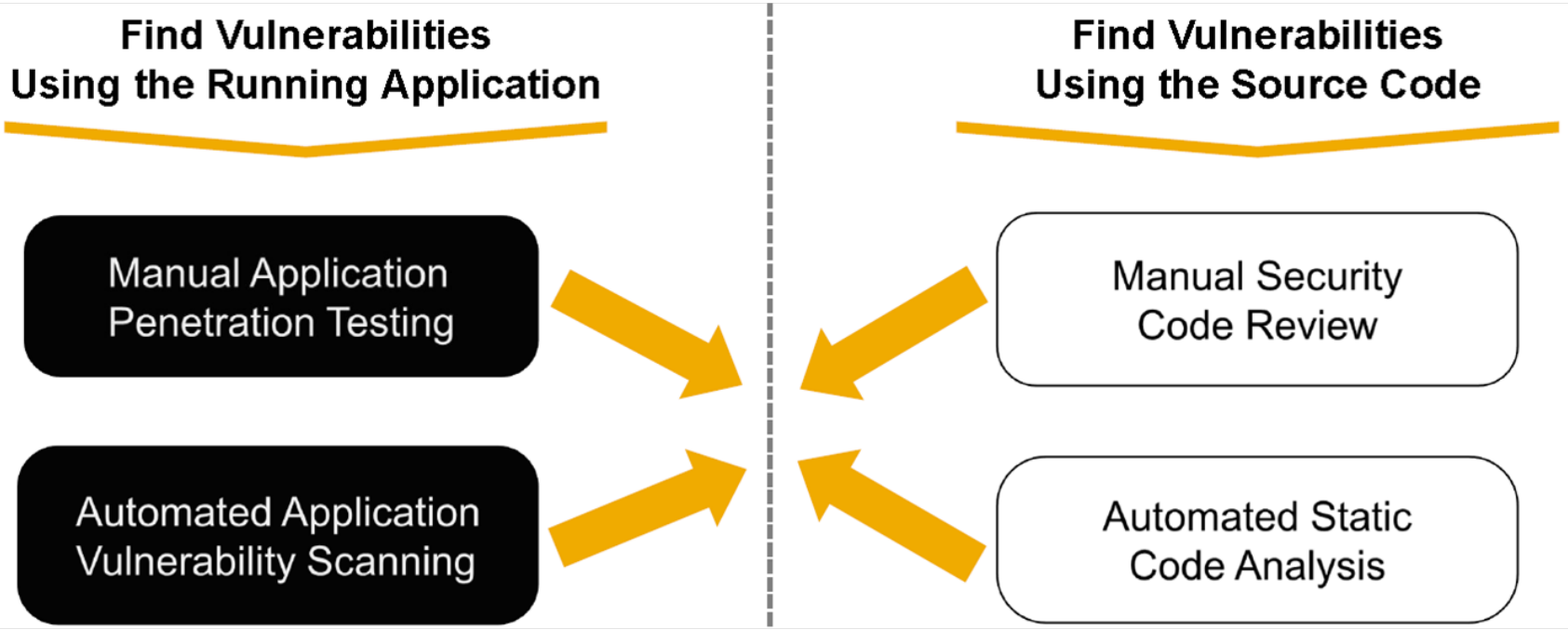
Vulnerability types of CVE reports since 1999



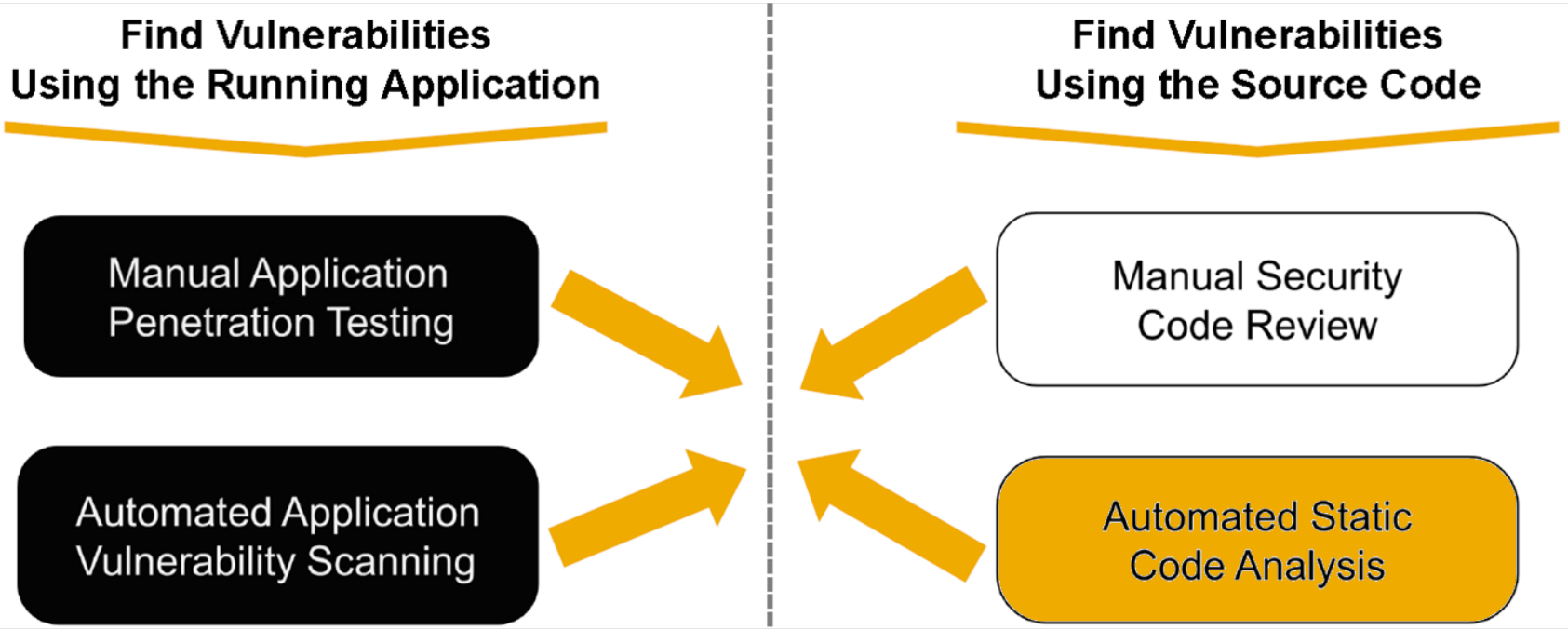
- Causes for most vulnerabilites are
  - programming errors
  - configuration errors
- Patching is
  - expensive
  - may introduce new bugs

How can we ensure that no vulnerable code is shipped?

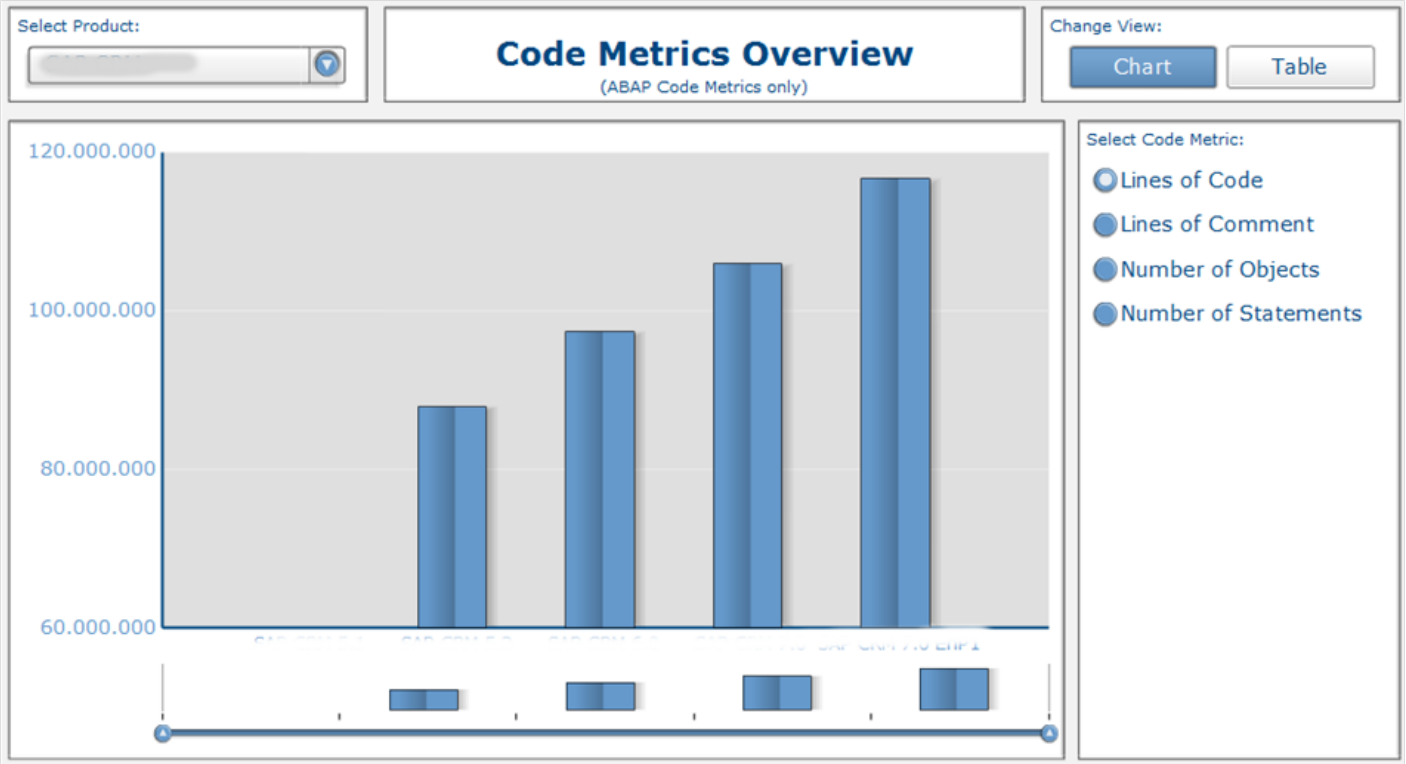
Finding Security Vulnerabilities



Finding Security Vulnerabilities

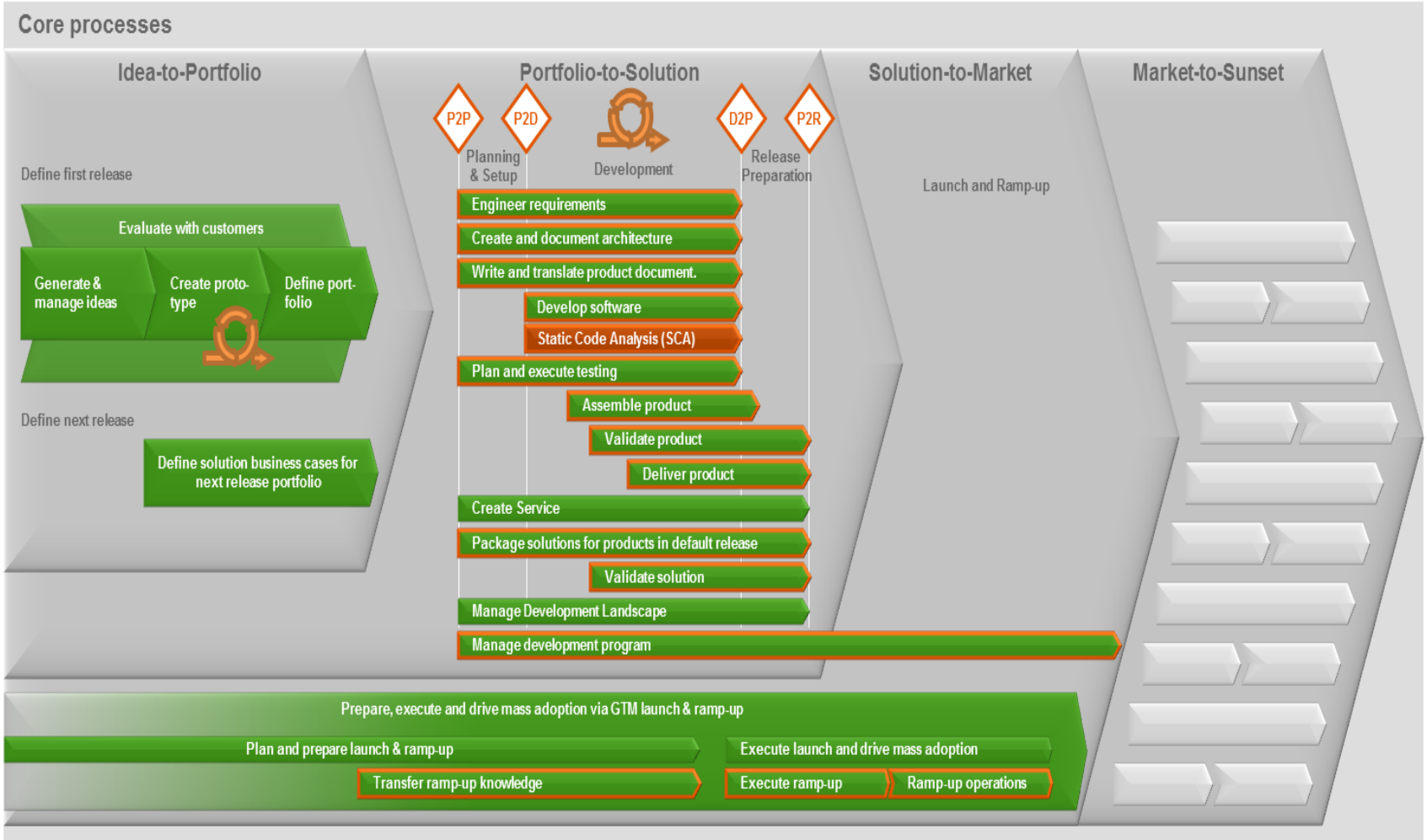


Evolution of Source Code

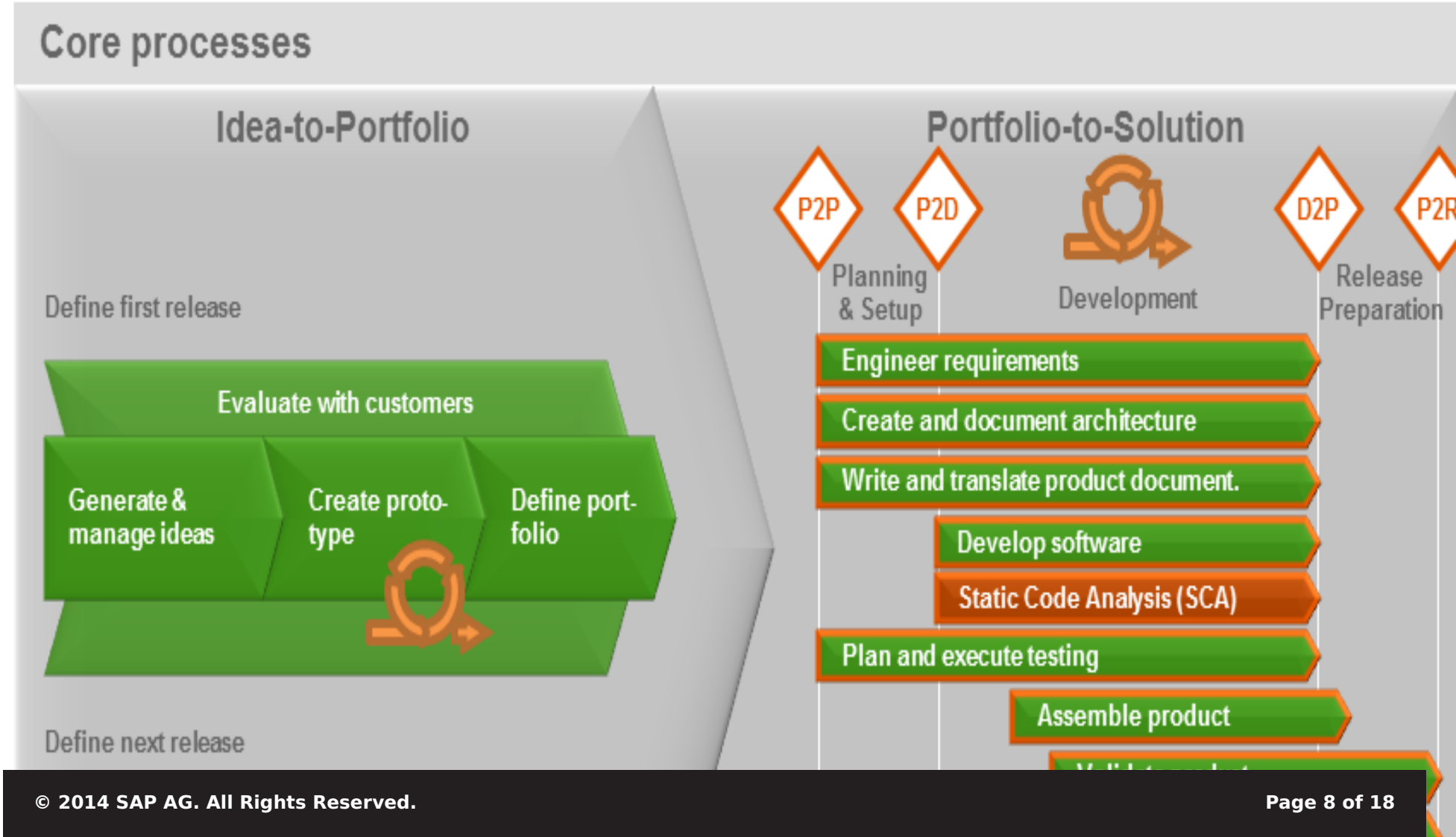


- Increase in
  - code size
  - code complexity
  - number of products
  - product versions

SAST as Part of SAP's SDL



SAST as Part of SAP's SDL



So Everything is Secure Now, Right?

> Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.
>
> Undisclosed sales engineer from a SAST tool vendor.

So Everything is Secure Now, Right?

> Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.
>
> Undisclosed sales engineer from a SAST tool vendor.

Yes, this tools exists! It is called Code Assurance Tool (cat):

So Everything is Secure Now, Right?

> Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.
>
> Undisclosed sales engineer from a SAST tool vendor.

Yes, this tools exists! It is called Code Assurance Tool (cat):

- The cat tool reports each line, that might contain a vulnerability:

brucker@fujikawa – /usr/src/modules/tp-smapi

File Edit View Search Terminal Help

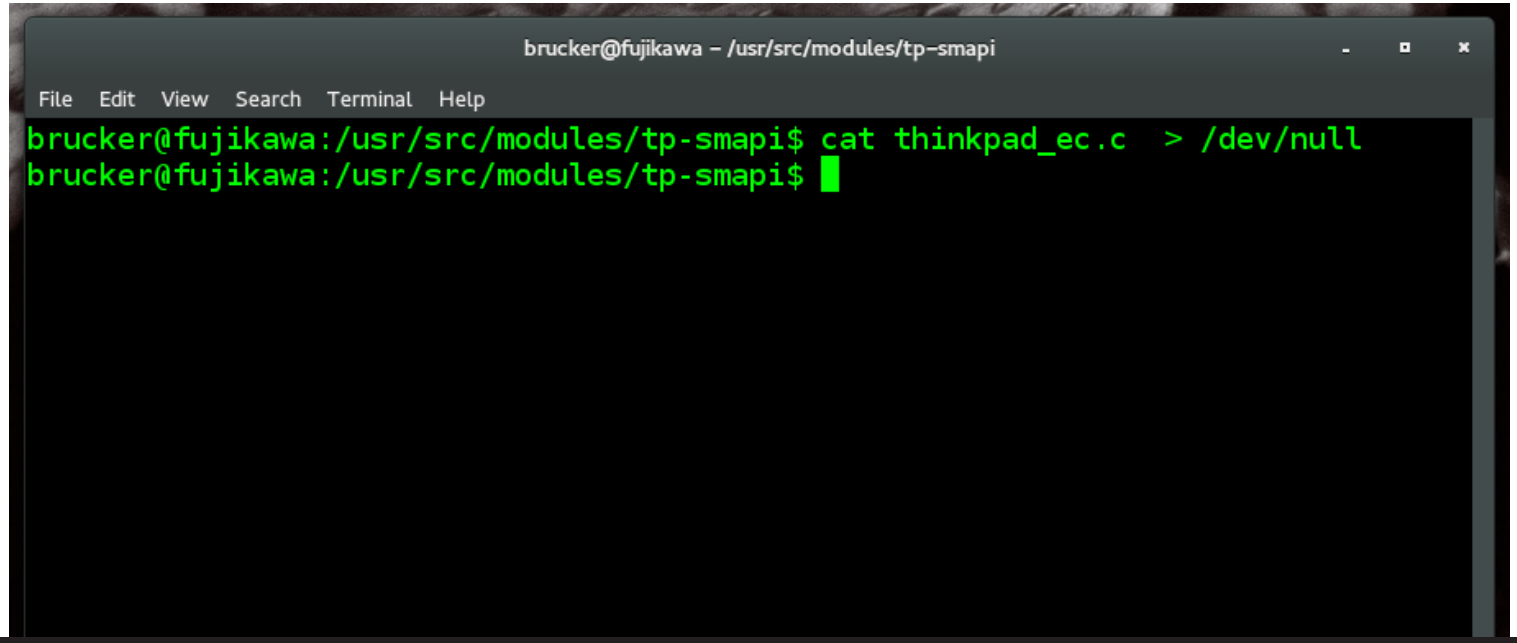
```
brucker@fujikawa:/usr/src/modules/tp-smapi$ cat thinkpad_ec.c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/dmi.h>

static int thinkpad_ec_request_row(const struct thinkpad_ec_row *args)
{
        u8 str3;
        int i;

        /* EC protocol requires write to TWR0 (function code): */
        if (!(args->mask & 0x0001)) {
                printk(KERN_ERR MSG_FMT("bad args->mask=0x%02x", args->mask));
                return -EINVAL;
        }
```

So Everything is Secure Now, Right?

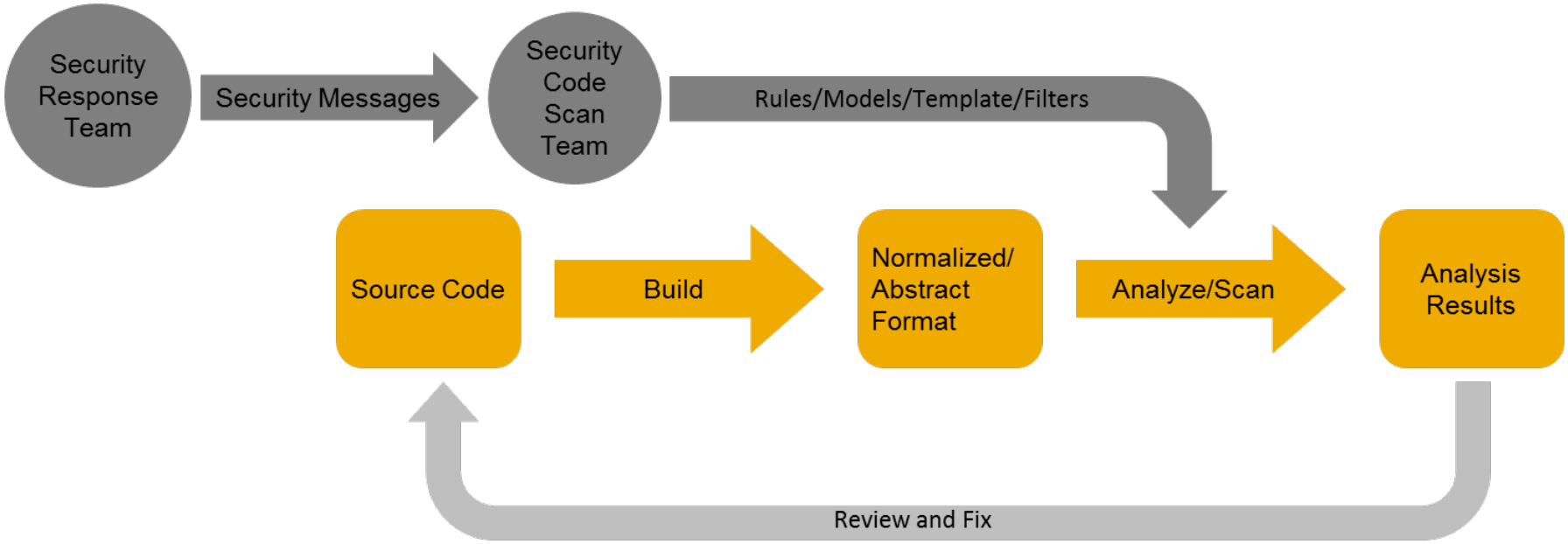
> Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.
>
> Undisclosed sales engineer from a SAST tool vendor.

Yes, this tools exists! It is called Code Assurance Tool (cat):

- The cat tool reports each line, that might contain a vulnerability:
- It supports also a mode that reports no false positives:



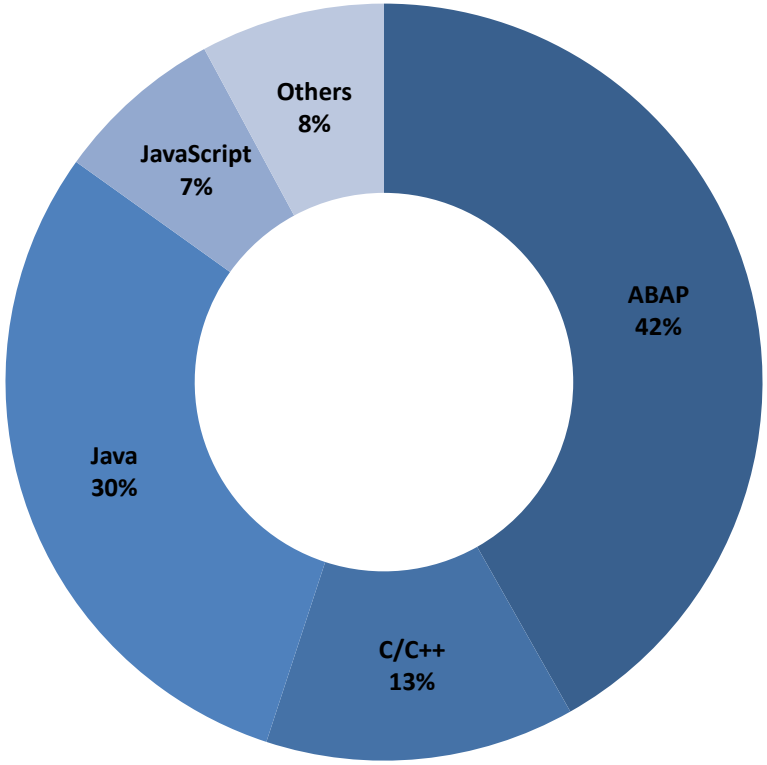
Continuous Improvements



Further input channels:

- Development teams
- Internal research
- Scan reviews
- Code reviews
- . . .

SAST Solutions Applied at SAP



Language	Tool	Vendor
ABAP	CVA (SLIN_SEC)	SAP
C/C++	Coverity	Coverity
Others	Fortify	HP

- Mandatory for all products
- Multiple billions lines analyzed

Other important pillars of SAP's SDL:

- Secure programming training's
- Pen tests on the final product

In addition:

- Own research (e.g., JS, Mobile)
- Ongoing evaluation of
  - alternative tools and
  - complementary techniques.

Open Issues

- Estimating the risk of not fixing security issues is hard
  - How to prioritize security vs. functionality
  - In case of doubt, functionality wins
- Pushing SAST across the software supply chain
  - Consumed software (OSS, third-party products)
  - SAP Customers, partners, and OEM products
- Huge and hybrid multi-language applications
  - Client-server applications
  - Web-frameworks
- Dynamic programming paradigms and languages
  - JavaScript, Ruby, etc.
- Lack of standardized regression test suites
  - Different tools
  - Different versions of the same tool

Lessons Learned: Recommendations (1/3)

Follow the recommendations given by Chandra et al:

- Start small
  - Start with one pilot
  - Succeed with pilot before larger roll-out
- Go for the throat
  - Start with the main security threat
- Appoint a champion
  - Identify a developer that knows all parts of the application
  - Make this developer your tool champion
- Measure the outcome
  - Track and measure the generated data
- Make it your own
  - Adapt the tool to your needs
  - SAST tools are not “off-the-shelf” products

Lessons Learned: Recommendations (2/3)

Based on our experiences, we add:

- Plan and invest enough resources
  - Introducing SAST requires significant resources
  - Integration, Analysis, Education, . . .
- Plan and invest enough infrastructure
  - If the tools are slow, nobody will use them
- Do understand your developers as your friends
  - Do not follow the “security review” model
  - SAST tools should be understood as “debug tool”
- Execute scans regularly
  - SAST is not a one-time effort

Lessons Learned: Recommendations (3/3)

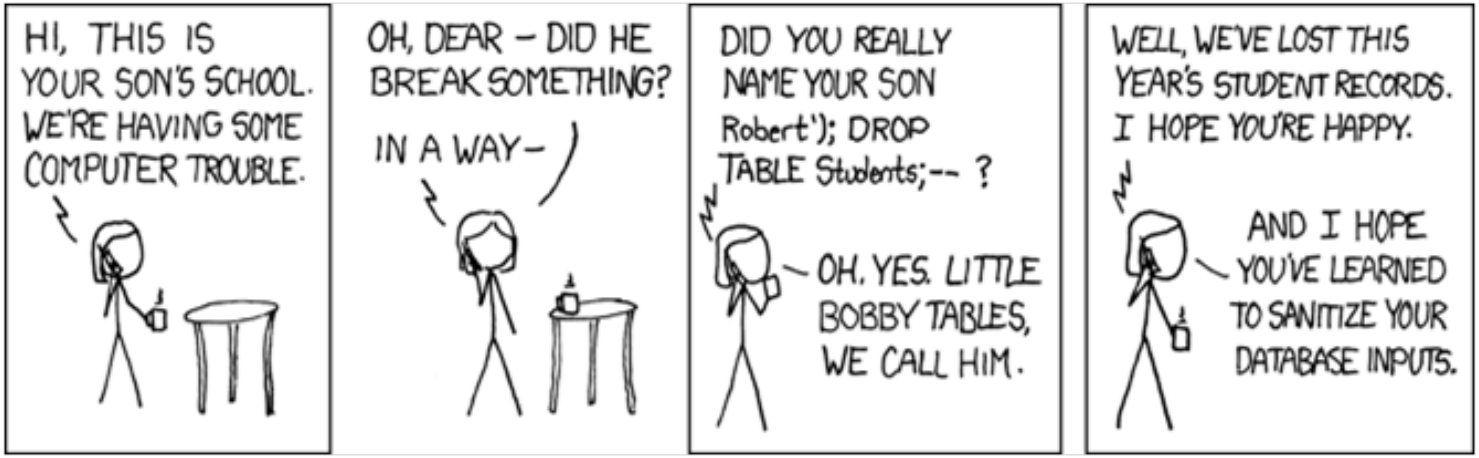
- Plan your changes and updates
  - All changes to the tools might change the results
- Do get support (and commitment) from your management
  - Introducing SAST will cost money and effort
  - Minimize the risk of discussing “security vs. features”
- Do not stop here.
  - Introducing SAST is only the first step
  - Use complementary techniques, e.g.,
    - Threat modeling
    - Dynamic testing tools
    - Penetration tests
    - . . .

Conclusion

> You cannot pay people well enough, to do proper code audits. I tried it.
>
> Yaron Minsky, Jane Street Capital

- We can confirm the results of Scandariato et al that show that SAST is the most effective and efficient security testing method
- Embed your SAST efforts into a holistic security testing strategy

Thank you!



http://xkcd.com/327/

Bibliography I

Ruediger Bachmann and Achim D. Brucker.
Developing secure software: A holistic approach to security testing.
Datenschutz und Datensicherheit, March 2014.

Achim D. Brucker and Uwe Sodan.
Deploying static application security testing on a large scale.
In gi *Sicherheit 2014*, Lecture Notes in Informatics. gi, March 2014.

© 2014 SAP AG. All rights reserved