1000 Projects later
Security Code Scans at SAP

OWASP
The Open Web Application Security Project
• **Ruediger Bachmann**
  is a Development Architect at SAP AG working, as member of the central code analyses team, in the areas application security and code analysis.

• **Dr. Achim D. Brucker**
  is a Senior Researcher in the Security & Trust Program of SAP Research as well as a member of the central code analysis team of SAP AG. His research interests includes static and dynamic security testing.
Agenda

Why is SAP using Static Code Analysis?
Secure Development Lifecycle at SAP
Static Code Analysis at SAP
Challenges and Outlook
• Started rollout in June 2010
• Centrally guided by a project team
  – Definition of Security Requirements
  – Establishment of Scan Infrastructure
• Support of the most important languages
• SAP development and third party code
Vulnerability Disclosures Growth by Year
1996-2011

Source: IBM X-Force® Research and Development
Evolution of Code

Select Product:
SAP CRM

Code Metrics Overview
(ABAP Code Metrics only)

Select Code Metric:
- Lines of Code
- Lines of Comment
- Number of Objects
- Number of Statements

Change View:
Chart
Table
Security Testing

OWASP
The Open Web Application Security Project

Find Vulnerabilities Using the Running Application

Manual Application Penetration Testing

Automated Application Vulnerability Scanning

Find Vulnerabilities Using the Source Code

Manual Security Code Review

Automated Static Code Analysis
Dynamic Security Testing

Characteristics
- Black box approach
- Sends input to applications and analyses response

Advantages
- Provides concrete examples (attacks)
- Analyze dataflows across multiple components

Disadvantages
- Coverage unclear
- Requires test system
• Characteristics
  – White box approach
  – Analyses abstraction of the source (binary)

• Advantages
  – Explores all data paths / control flows
  – Can analyse single modules (unit test)

• Disadvantages
  – High false positive rate (not exploitable findings)
  – Does not consider application environment
Why is SAP using Static Code Analysis?

Secure Development Lifecycle at SAP

Static Code Analysis at SAP

Challenges and Outlook
• Education:
The prerequisite for achieving a high security quality

• Security awareness:
Reducing the number of “built-in” security problems

• Trained persons:
Analyze and fix vulnerabilities much more efficiently

• Trainings:
Secure Programming, Build & Scan, Auditing, ....
Secure Development Lifecycle (SDLC) at SAP

• Structure the investment of time and resources
  – to safeguard a high level of security
  – to ensure security standards across all areas

• Security requirements
  – are taken into account and
  – are implemented in all phases of product development
• Developer
  – fixes software security issues
• Security Expert
  – review scan results, decides on fixes
• Build Master
  – scans the source code, manages results
• Scrum Master
  – requests scan, assigns vulnerabilities to developers
For passing D2P Q-gate, evidence has to be provided that the source code has been scanned and exploitables have been fixed.

P2D: Planning to Development. / D2P: Development to Production. / P2R: Production to Ramp-up (gradual roll-out to customers).
• Third party code
  – Open Source libraries and frameworks
  – Freeware
  – other third party components
• Different approaches
  – SAST analysis by SAP
  – Certificate from vendor
  – SLA with vendor
Why is SAP using Static Code Analysis?
Secure Development Lifecycle at SAP
Static Code Analysis at SAP
Challenges and Outlook
• Over 2000 developers are using SAST tools
• Over 500 MLOC scanned
<table>
<thead>
<tr>
<th>Language</th>
<th>Scan Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP</td>
<td>SAP</td>
</tr>
<tr>
<td>C/C++</td>
<td>Coverity</td>
</tr>
<tr>
<td>Others</td>
<td>HP/Fortify</td>
</tr>
</tbody>
</table>
• SAP on Corporate Security Requirements
  • SAP Applications shall be free of backdoors
  • SQL injection vulnerabilities shall be avoided
  • Cross-Site Scripting vulnerabilities shall be prevented
  • Directory traversal vulnerabilities shall be prevented
  • The system shall be protected against buffer overflow vulnerabilities

• OWASP Top 10
• CWE/SANS Top 25 2011
• CVE
• Collect feedback from the
  • Product Security Response Team
  • Development Teams
• Develop rules/models to improve the scans
• Continuously improve the infrastructure
• Continuously improve the rollout process
Input to Improve Code Scans

OWASP
The Open Web Application Security Project

• Further input channels: Development teams, internal research, scan reviews, code reviews
Lessons Learned

• Scans have to be obligatory but not introduced ‘brute force’

• Establish Secure Development Life Cycle make scans a natural part of development

• Plan carefully
  – Do not start with scans right before Dev. Close
  – Do it regularly (nightly)

• Do not introduce changes during development
Why is SAP using Static Code Analysis?
Secure Development Lifecycle at SAP
Static Code Analysis at SAP
Challenges and Outlook
C, C++, CVS, Perforce, ABAP Repository, Git, Ant, Maven, Make, Japro, Mapro, Hudson, Jenkins MS Foundation Server, Mobile, Android, iOS, Objective-C, Java, Local Build, Central Build, Different Development Cycles, Development on/for different OS like Linux Windows, Unicode, Non-Unicode, Developer IDE Support (Plugins for Eclipse, Visual Studio), Silverlight, BSP, Flash, TCL/TK, Erlang, Perl, JSP, TagLib, OSS, HTML5/JavaScript, Convenient Access to Central Scan Result Repositories, Provide a Secure Code Scan Infrastructure, Convince everybody that code scanning helps to improve software, Open Source, Third Party Software, Merge Threat Modeling with Code Scans, Q-Gate-Reporting, Establish a stable plan-scan-check-correct workflow, ...
• Assume the following index.html:

```
<TITLE>Welcome!</TITLE>
Hi

<SCRIPT>
    var pos=document.URL.indexOf("name=")+5;
    document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
Welcome to our system
```

and a call

index.html?name=<script>alert(document.cookie)</script>

• resulting in a DOM-based XSS attack
• DOM implementations are Browser specific
• A simple script statement
  
  ```javascript
  <script language="javascript">
    document.write("<script>src='other.js'></script>";
  </script>
  
  • Dynamically creating script tags
  
  ```javascript
  var oHead = document.getElementsByTagName('HEAD').item(0);
  var oScript = document.createElement("script");
  oScript.type = "text/javascript";
  oScript.src = "other.js";
  oHead.appendChild(oScript);
  ```
  
  • Or using eval() directly (not shown here)
Combining the complexity of two worlds

```javascript
var entry=JSON.parse(data);
query = "insert into "FOO(".NAME")";
var conn = $.db.getConnection();
conn.execute(query);
```
• SAST works very well for
  – “traditional” programming languages
  – Analyzing data paths within one technology

• Many new development uses JavaScript
  – HTML5/JavaScript UIs
  – Server-side JavaScript

• JavaScript
  – Untyped
  – Dynamic programming model
You cannot pay people well enough, to do proper code audits.

I tried.

(Yaron Minsky, Jane Street Capital)
Thank you!