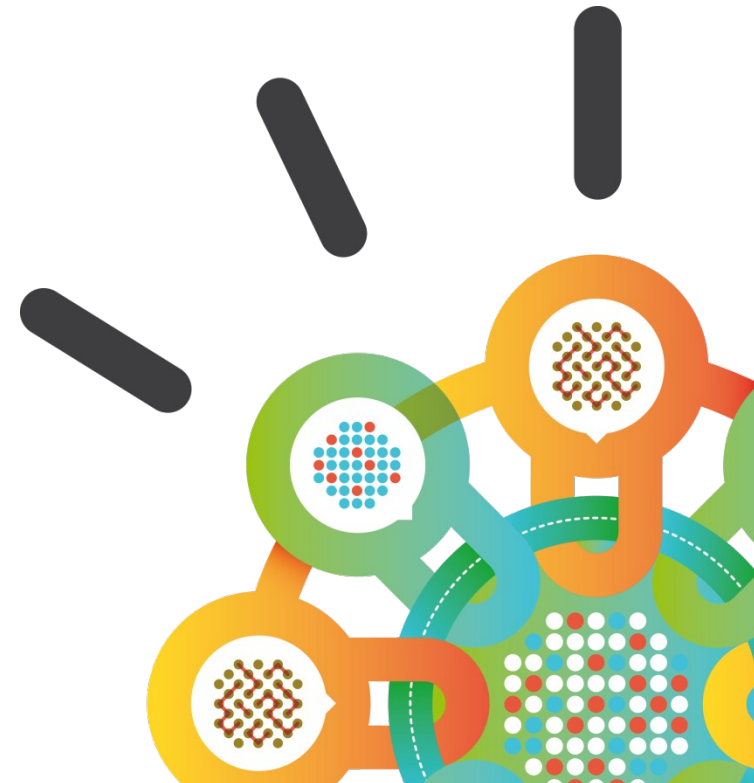




# AppScan 9.0 Security for Applications

Stefan.Mandl@de.ibm.com





# und im Jahr 2012, ...

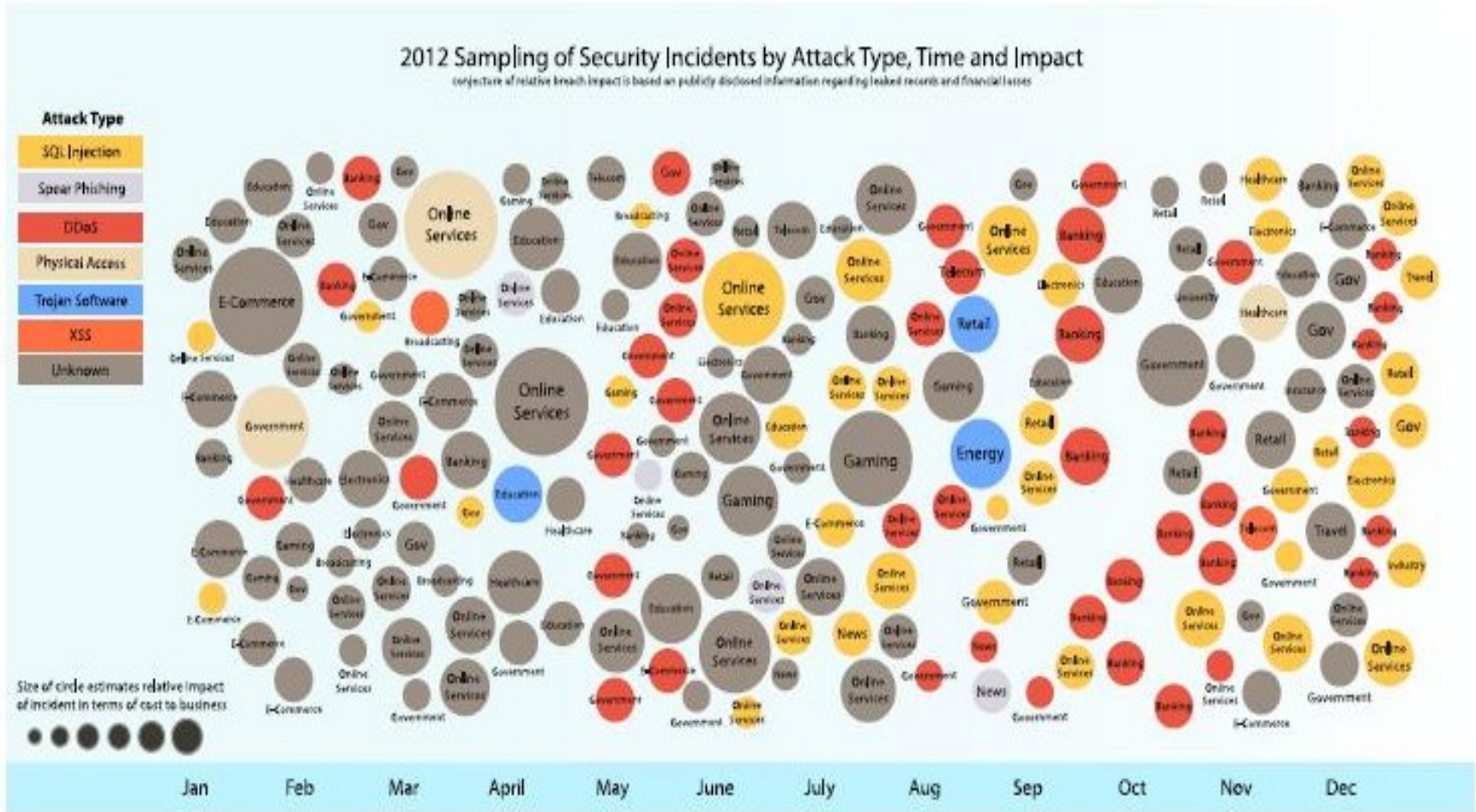


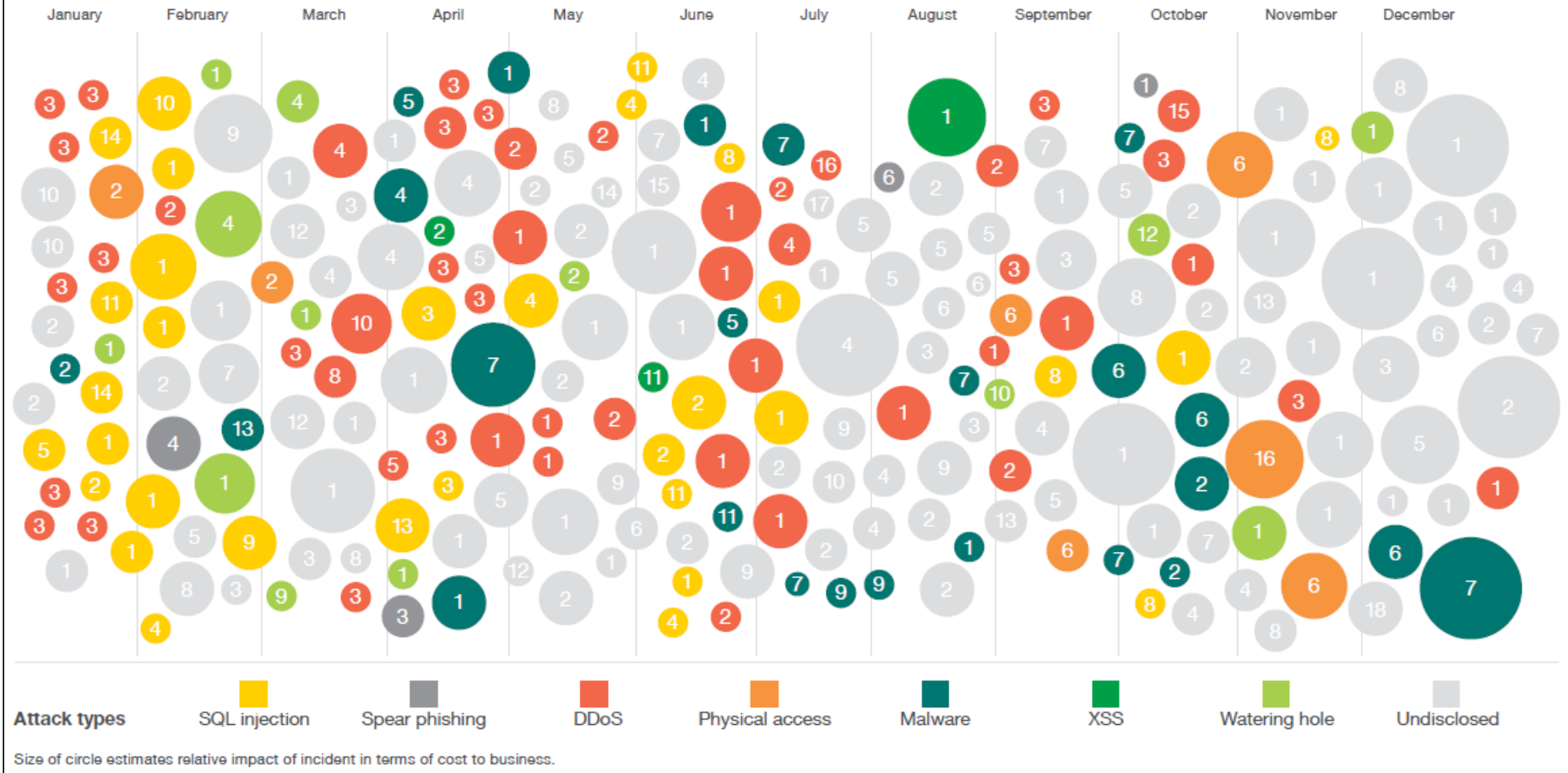
Figure 3: 2012 Sampling of Security Incidents by Attack Type, Time and Impact

Source: IBM X-Force® Research 2012 Trend and Risk Report

# und 2013, ...

## Sampling of 2013 security incidents by attack type, time and impact

conjecture of relative breach impact is based on publicly disclosed information regarding leaked records and financial losses



Source: IBM X-Force® Research 2013 Trend and Risk Report

## Der Mythos: “Wir sind sicher”

**Wir haben  
Firewalls**

**Wir lassen jedes Jahr  
ein Audit durchführen**

**Wir haben SSL  
Verschlüsselung**

**Wir haben Netzwerk-  
Scanner**



## ... und die Security Lücken

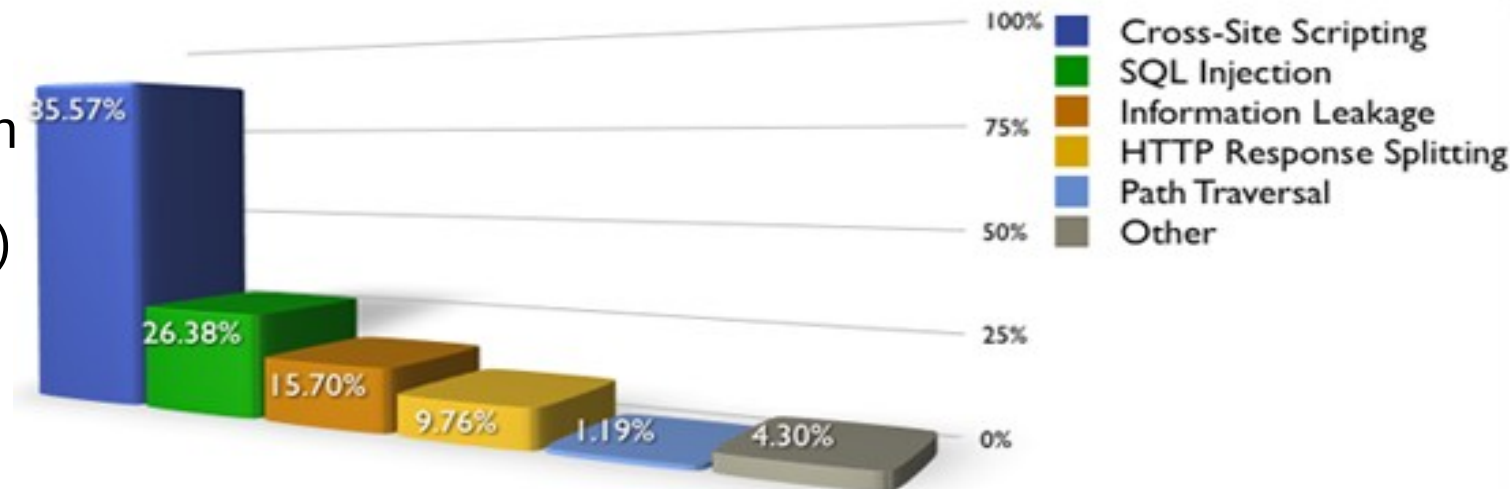
1. "Cross-Site Scripting" (XSS)
2. "SQL Injection"
3. Verstecktes Ausführen einer Datei
4. Unsichere "Direct Object Reference"
5. Verfälschung eines "Cross-Site Requests"

## Top Ten des "Open Web Application Security Project" (OWASP)

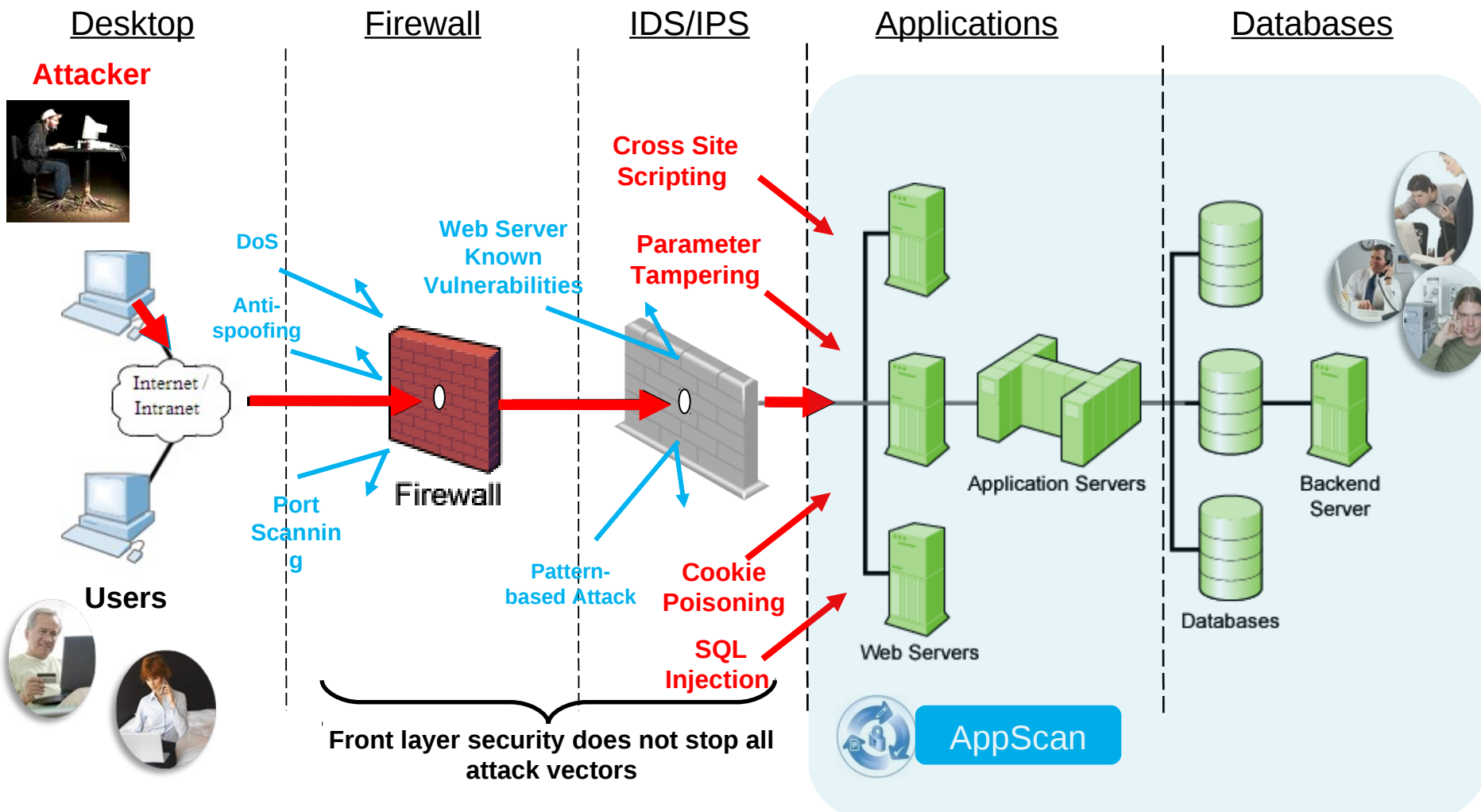
6. Informationsverlust und unsaubere Fehlerbehandlung
7. Broken Authentication & Session Management
8. Unsichere Kryptografie Speicherung
9. Unsichere Kommunikation
10. Fehlerhafte Abwehr von URL Zugriffen

Schwachstellen Statistik  
(31.373 Seiten)

Percentage of websites vulnerable by class (Top 5)



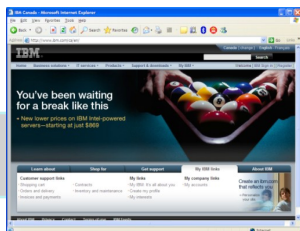
# Wie "lösen" Unternehmen das Problem?



# Analyse der Applikation

## Dynamische Analyse = Blackbox

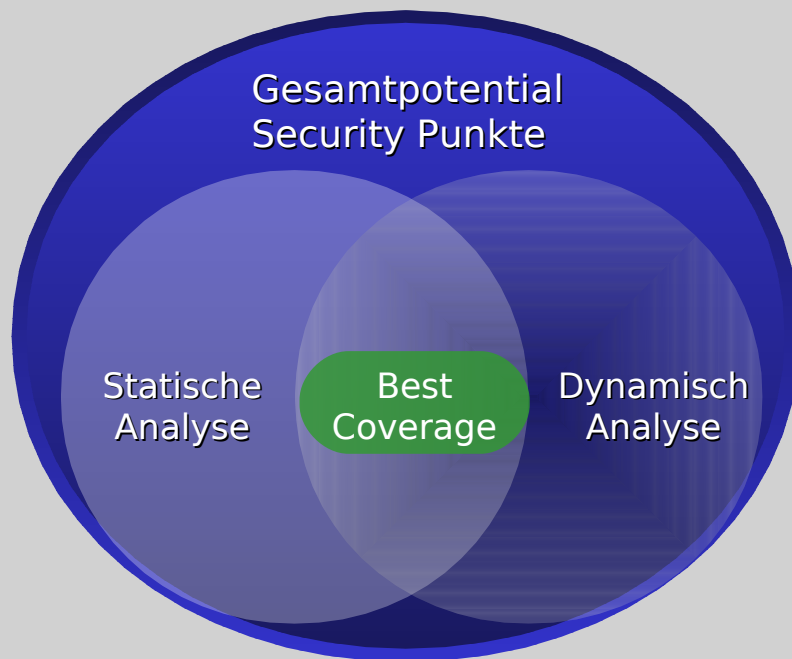
Scanning nach Security Schwachstellen auf Basis der compilierten Applikation



## Statische Code Analyse = Whitebox

Scanning nach Security Schwachstellen auf Source Code Basis

```
188 {  
189     ..... TnxCSSFontStyle .....  
190 }  
191 constructor TnxCSSFontStyle.Create(aFontStyle: TnxCSSFontStyleEnum);  
192 begin  
193     inherited Create(aFontStyle);  
194     FFontStyle := aFontStyle;  
195 end;  
196  
197 function TnxCSSFontStyle.GetStyleValue: string;  
198 begin  
199     Result := nxCSSFontStyleStrings[FontStyle];  
200 end;  
201  
202 procedure TnxCSSFontStyle.SetFontStyle(Value: TnxCSSFontStyleEnum);  
203 begin  
204     if FFontStyle <> Value then  
205         begin
```





# AppScan Standard Edition



The screenshot displays the IBM Rational AppScan interface. The main window shows a list of security issues for 'My Application', sorted by severity in descending order. The issues include Blind SQL Injection (2), Cross-Site Scripting (8), Database Error Pattern Found (11), DOM Based Cross-Site Scripting (2), HTTP PUT Method Site Defacement (4), Inadequate Account Lockout (1), Permanent Cookie Contains Sensitive Session Information (1), and Predictable Login Credentials (1). The 'Cross-Site Scripting' issue is highlighted, showing a high severity (CVSS 7.5) and a fix recommendation. The dashboard at the bottom left shows a total of 126 issues, with a bar chart indicating the distribution of issues by severity: 53 High, 32 Medium, 24 Low, and 17 Informational. The status bar at the bottom indicates 126 Security Issues, 53 High, 32 Medium, 24 Low, and 17 Informational.

**Issue Information**

**Cross-Site Scripting**

**High**  
CVSS Metrics... (7.5)

Base   
Temporal   
Environmental

**URL:** http://www.altoromutual.com/search.aspx  
**Entity:** txtSearch  
**Security Risk:** It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user

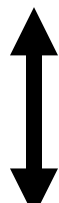
The script AppScan injected seems to be included in the response. If the screen shot below shows a simulation of the pop-up that the injected script produced, this is proof that the application is vulnerable to Cross-Site Scripting. If not, to verify this vulnerability: 1) Open the Request/Response tab and click Show in Browser, and see if a pop-up appears. Note that

Visited URLs 107/107 | Completed Tests 22106/22106 | 126 Security Issues | 53 High | 32 Medium | 24 Low | 17 Informational



Deployed Application

Dynamic Analysis



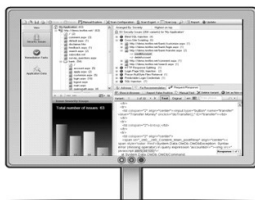
AppScan Standard ( Enterprise)

(DAST)

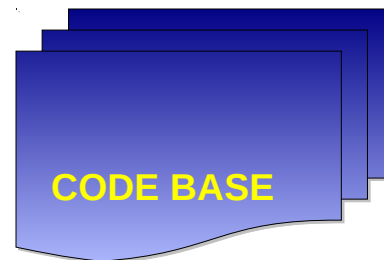


Deployed Application

Dynamic Analysis



AppScan Standard (DAST)



CODE BASE

Static Analysis



| Severity | Vulnerabilities | Exceptions |        | Total |
|----------|-----------------|------------|--------|-------|
|          |                 | Type       | Type # |       |
| High     | 16              | 16         | 327    | 433   |
| Medium   | 5               | 12         | 31     | 46    |
| Low      | 3               | 0          | 307    | 310   |

| Severity | Type             | API/Source               |
|----------|------------------|--------------------------|
| High     | Injection SQL    | api/jsp/sgl/Statement.do |
| High     | Validation Req   | rs.jsp/sgl/Servlet.do    |
| High     | Injection SQL    | api/jsp/sgl/Statement.do |
| High     | Injection SQL    | api/jsp/sgl/Statement.do |
| High     | Validation Error | rs.jsp/sgl/Servlet.do    |
| High     | Injection SQL    | api/jsp/sgl/Statement.do |
| High     | Validation Error | rs.jsp/sgl/Servlet.do    |

AppScan Source (SAST)



# AppScan Source Edition

**IBM Rational AppScan Source Edition for Security**

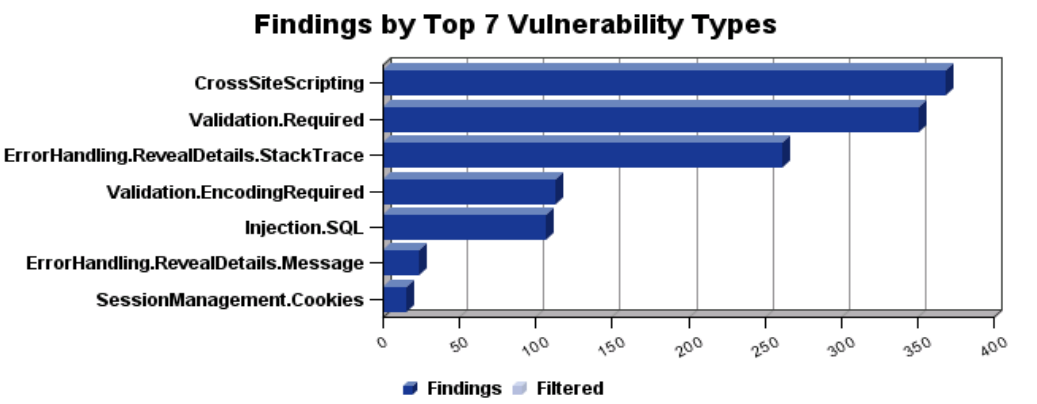
File Edit Scan Tools Admin View Perspective Help

Configuration Triage Analysis Configuration: default

Assessment Summary Filter Editor

Chart Style: Bar Chart Property: Vulnerability Type Show All Findings

### Findings by Top 7 Vulnerability Types



| Reset         | Vulnerability | Exceptions |            | Totals      |
|---------------|---------------|------------|------------|-------------|
|               |               | Type I     | Type II    |             |
| High          | 18            | 375        | 198        | 591         |
| Medium        | 21            | 82         | 109        | 212         |
| Low           | 265           | 113        | 116        | 494         |
| <b>Totals</b> | <b>304</b>    | <b>570</b> | <b>423</b> | <b>1297</b> |

Findings Filtered

Findings SmartAudit Console OWASP

Showing All Findings

- Findings (4,723)
  - A1 - Unvalidated Input (395)
  - A2 - Broken Access Control (11)
  - A3 - Broken Authentication and Sess
  - A4 - Cross Site Scripting (XSS) Flaws
  - A5 - Buffer Overflow (6)
  - A6 - Injection Flaws (354)
  - A7 - Improper Error Handling (641)
  - A8 - Insecure Storage (73)
  - A9 - Denial of Service (20)
  - A10 - Insecure Configuration Manag
  - Other (1,914)

| Severity | Classification | Vulnerability Type | API          |
|----------|----------------|--------------------|--------------|
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Vulnerability  | CrossSiteScripting | javax.servle |
| High     | Type I         | CrossSiteScripting | javax.servle |
| High     | Type I         | CrossSiteScripting | javax.servle |
| High     | Type I         | CrossSiteScripting | javax.servle |

Finding Detail Bundles

**Details**

Context: [out . javax.servlet.is](#)

Classification:

Vulnerability Type: CrossSiteScripting

Severity: High

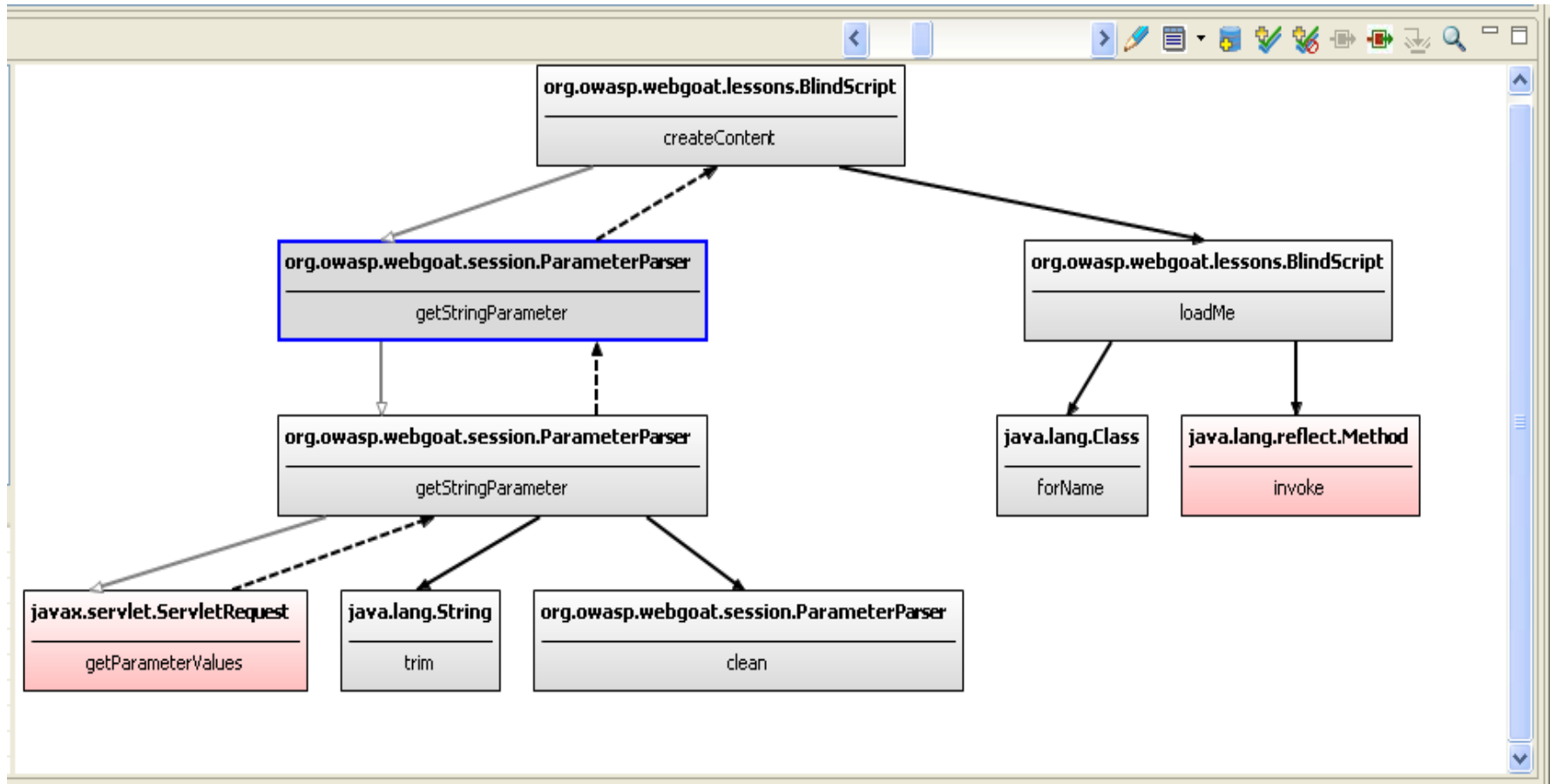
Bundle: CrossSiteScripting

**Reporting**

Lines Before: 5

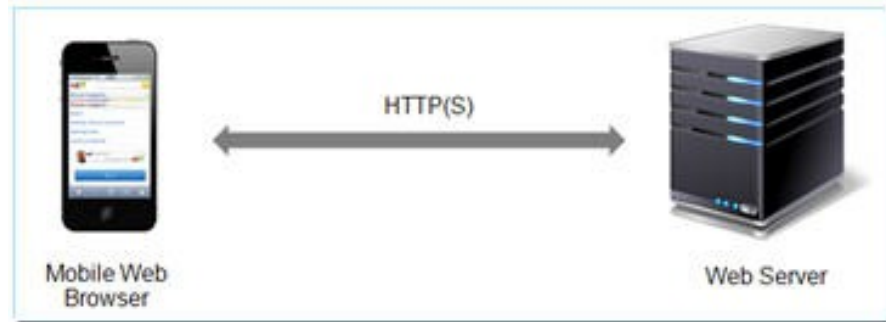
Lines After: 5

# Datenflussmodell in AppScan Source Edition



# Security Risk a Function of Mobile Application Type

Mobile **Web** applications



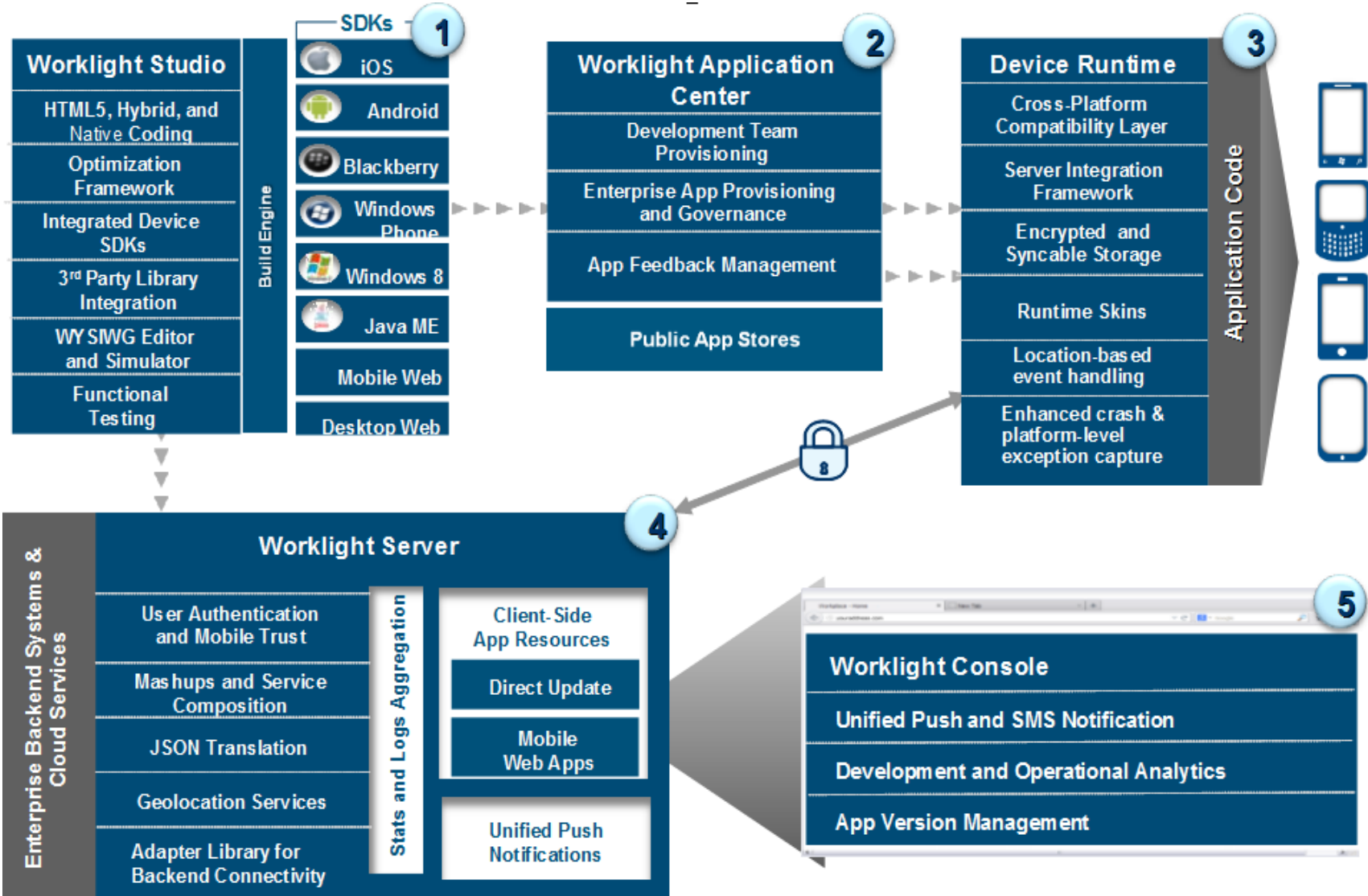
Mobile **Native** applications



Mobile **Hybrid** applications



# Was ist IBM Worklight ?





# AppScan Mobile Support: Server and Native



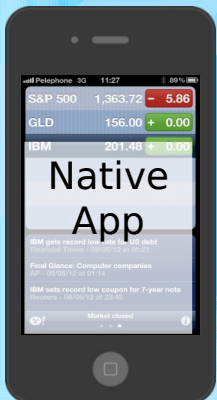
## Mobile Web Apps

JavaScript / HTML5 hybrid analysis

## Server Side Logic

SAST (source code)

DAST (web interfaces)



## Native Apps

Android applications  *Static Analysis*

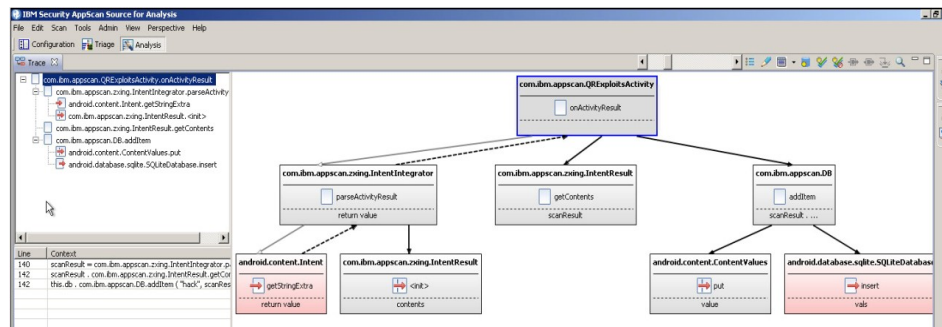
iOS applications  *Static Analysis*

JavaScript  *Static Analysis*



# IBM AppScan Source – Native SAST Support

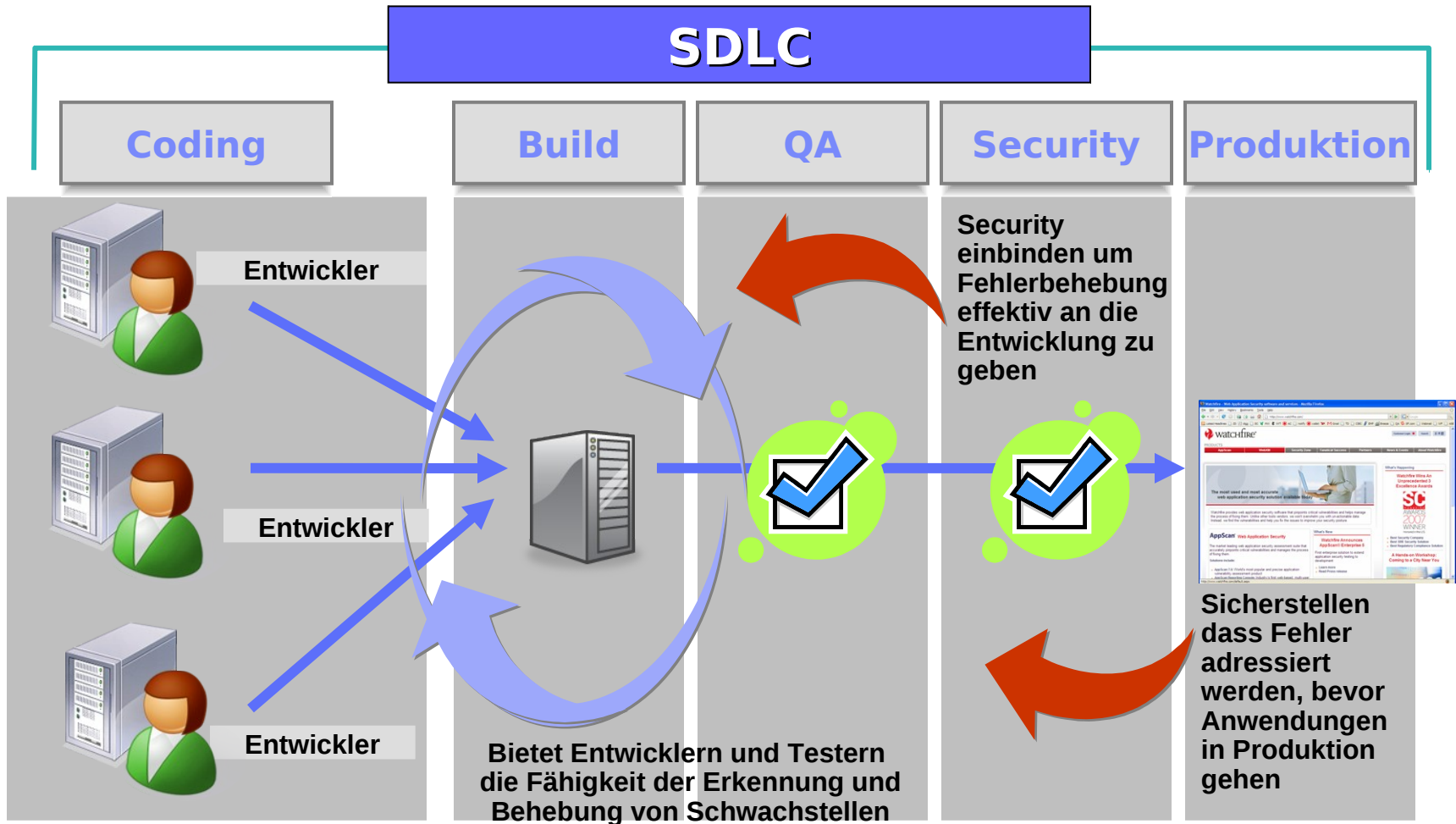
The most comprehensive mobile application security!



- Support for **Native Android** and **iOS** applications
- Security SDK research & risk assessment of over 40,000 APIs
- **Full** call and data flow analysis
  - ▶ Java
  - ▶ JavaScript
  - ▶ **Objective-C** (Mac OS X only)
- **Identify** where sensitive **data** is being **leaked**
- Ensures applications are not susceptible to malware



# Sicherheitstests bereits in den Entwicklungsprozess einbinden





[ibm.com/security](https://ibm.com/security)