



Tool-basierte Web Security

Matthias Rohr
20. März 2014



OWASP
The Open Web Application Security Project



OWASP

The Open Web Application Security Project

- Matthias Rohr
- CISSP, CSSLP, CISM, CCSK
- Application Security seit knapp 10 Jahren
- Autor sowie Berater und Geschäftsführer bei der Secodis GmbH in Hamburg
- Schwerpunkte:
 - Secure Development Lifecycle (SDL)
 - Tool-basierte Software-Sicherheitsanalysen



OWASP

The Open Web Application Security Project



Zunächst ein wenig FUD* ...

* = Fear, Uncertainty & Doubt

„Cyber-Angriffe“



OWASP

The Open Web Application Security Project

“The cyber threat is one of the most serious economic and national security challenges we face as a nation”

Obama, 29. Mai 2009



In einer Studie gaben 93% der größeren und 76% der kleinere Unternehmen in UK an, in 2011 von einem Cyber-Angriff betroffen gewesen zu sein

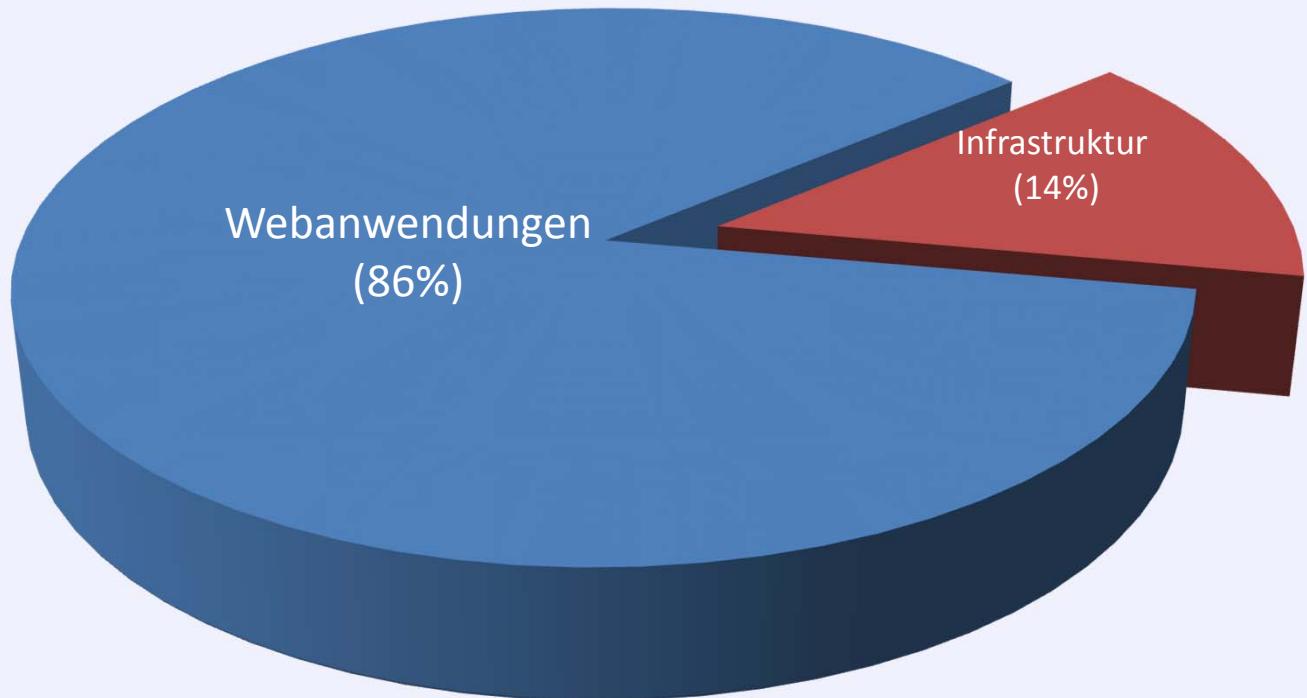
Information security breaches survey –
Technical report, April 2012, PwC

„Cyber-Angriffe“ = Angriffe auf Webanwendungen



OWASP

The Open Web Application Security Project



Quelle: UK Security Breach Investigations Report 2010

OWASP Top 10



OWASP

The Open Web Application Security Project

OWASP Top 10 – 2013

A1 – Injection

A2 – Broken Authentication and Session Management

A3 – Cross-Site Scripting (XSS)

A4 – Insecure Direct Object References

A5 – Security Misconfiguration

A6 – Sensitive Data Exposure

A7 – Missing Function Level Access Control

A8 – Cross-Site Request Forgery (CSRF)

A9 – Using Known Vulnerable Components

A10 – Unvalidated Redirects and Forwards

Merged with 2010-A7 into new 2013-A6



OWASP
The Open Web Application Security Project

OWASP Top 10 - 2013

The Ten Most Critical Web Application Security Risks

release



Creative Commons (CC) Attribution Share-Alike
Free version at <https://www.owasp.org>

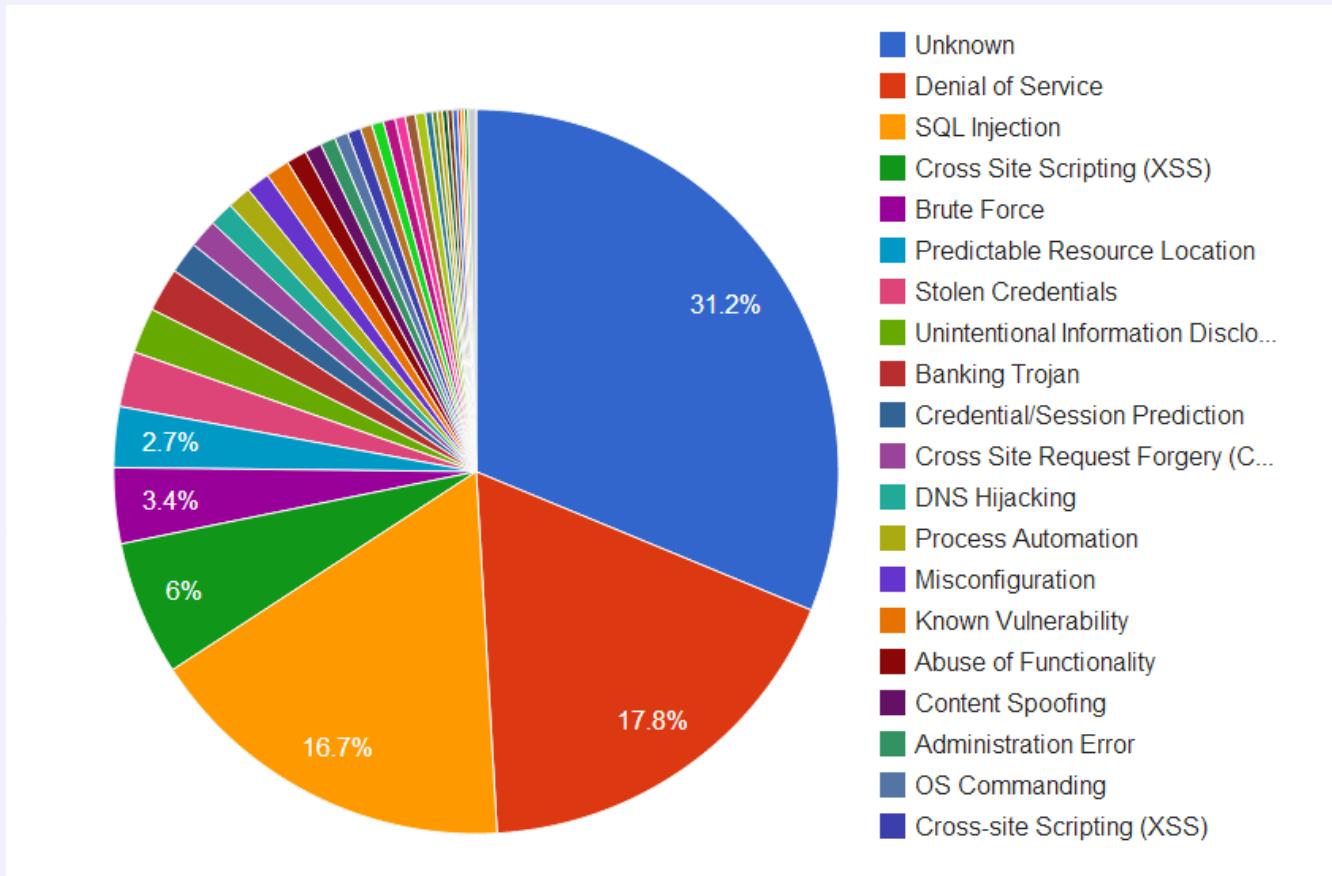
https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Angriffe



OWASP

The Open Web Application Security Project



1999-2011

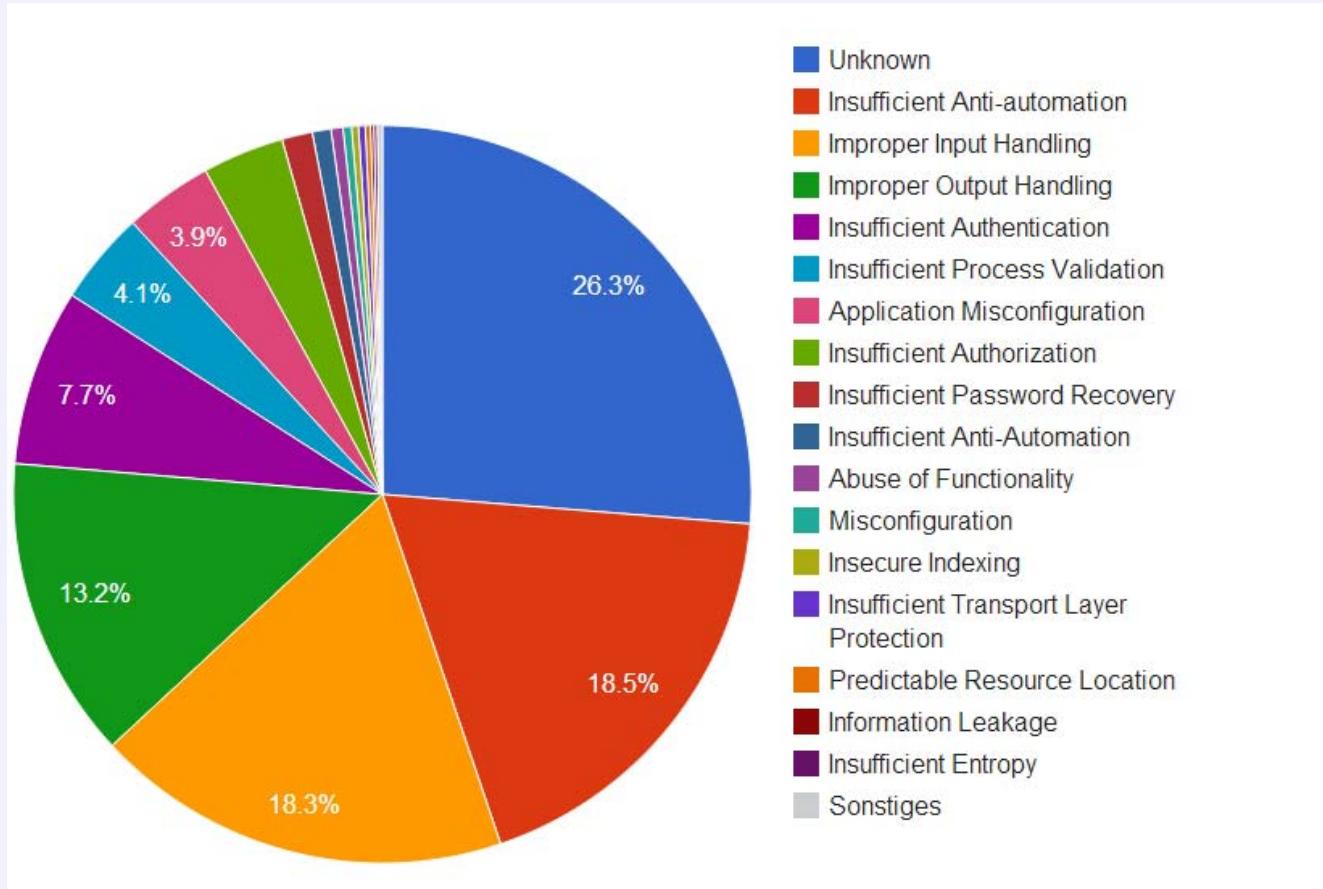
<http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database>

Schwachstellen



OWASP

The Open Web Application Security Project



1999-2011

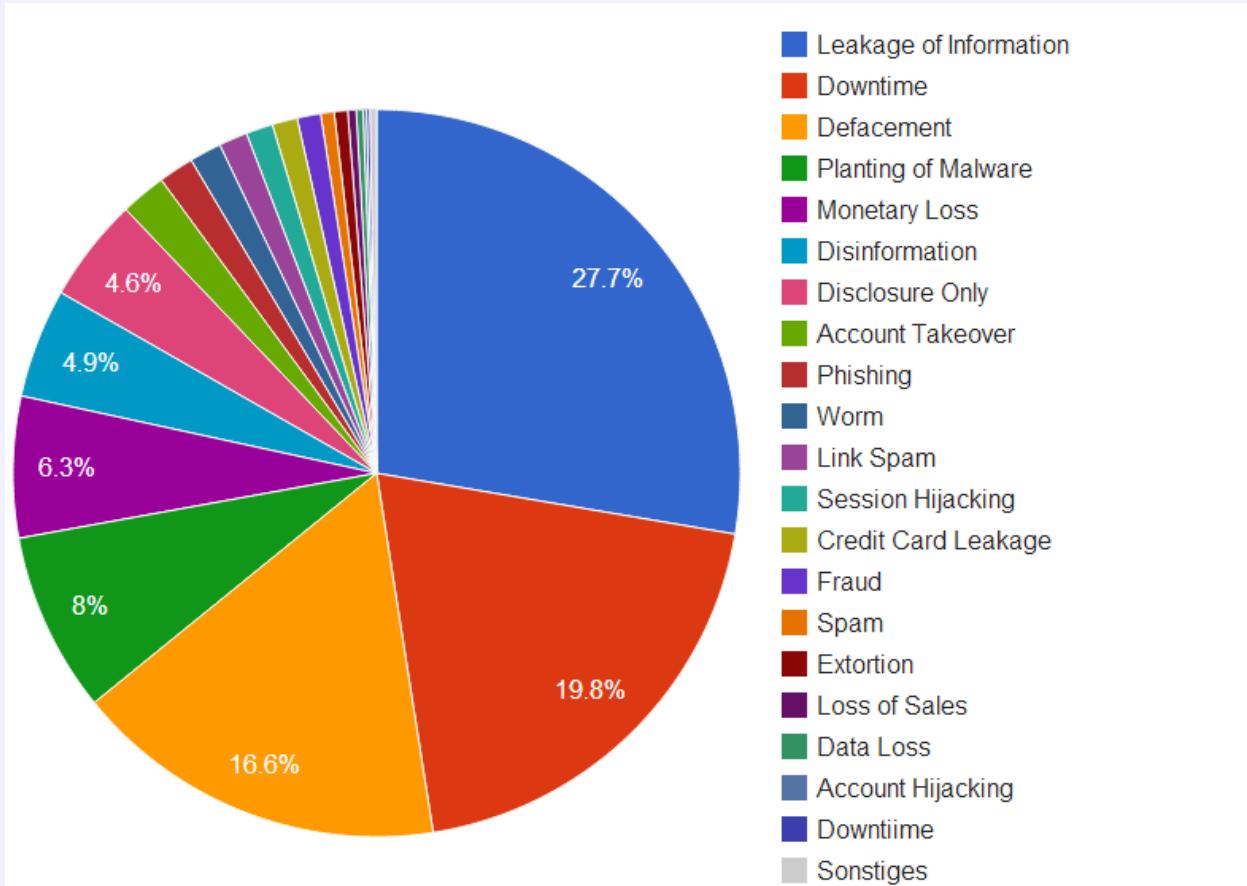
<http://projects.webappsec.org/w/page/13246995/Web-Hacking-incident-Database>

Schäden durch Angriffe auf Webanwendungen



OWASP

The Open Web Application Security Project



1999-2011

<http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database>

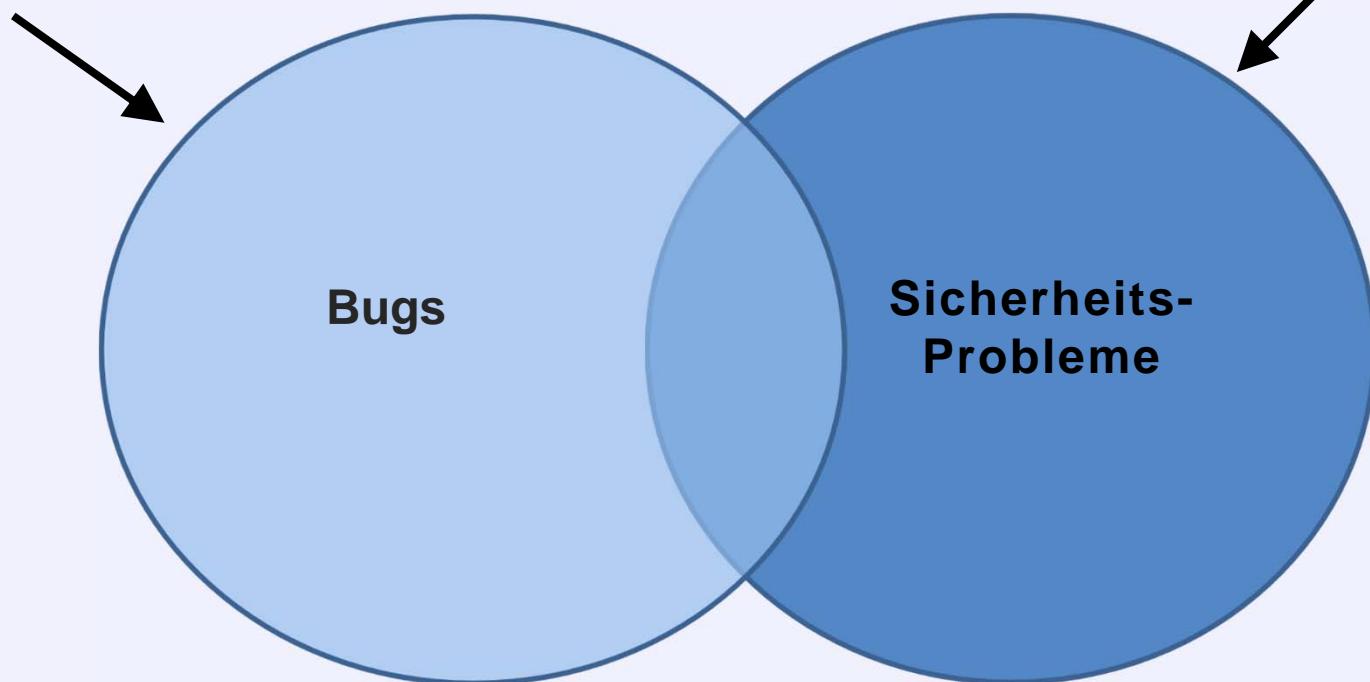
Bugs vs. Sicherheitsprobleme



OWASP

The Open Web Application Security Project

Anforderungen



Quelle: Joe Jarzombek, Software-Assurance: Enabling Enterprise Resilience through Security Automation and Software Supply Chaining Risk Management



OWASP

The Open Web Application Security Project

Tools ...

Code vs. Application Layer



OWASP

The Open Web Application Security Project

Schwachstellen in der laufenden Anwendung

Manuell: Pentest

Autom: Dynamic App. Security Testing (DAST)

```
<div class="lesson_title_box"><strong>Welcome back!</strong><span class="lesson_text_ap"><webSession.getUserNameInLesson()></span> - Edit Profile Page</div>
<div class="lesson_text">
    <form id="form1" name="form1" method="post" action="<=webSession.getCurrentLesson
    ()>.getFormAction()>">
        <Table border="0" cellpadding="0" cellspacing="0">
            <TR><TD width="110">
                First Name:
            </TD>
            <TD width="193">
                <input class="lesson_text_db" name="<=>CoastHillsFinancial.FIRST_NAME*>" type="text" value="<=>employee.getFirstName()>"/>
            </TD>
            <TD width="110">
                Last Name:
            </TD>
            <TD width="196">
                <input class="lesson_text_db" name="<=>CoastHillsFinancial.LAST_NAME*>" type="text" value="<=>employee.getLastName()>"/>
            </TD>
        </TR>
    </Table>
    Utilities
    Forward request
    Reply to request
    Discard
    Transfer
</div>
```

Session Mgmt /
CSRF,
Unsichere
Einbindung,
Logikfehler,
Anti-
Automatisierung,
...

Race Conditions
Buffer Overflows
Backdoors,
Unsichere APIs,
Anbindung
Backend,

XSS,
SQL Injection,
Datenv. allg.
Authentifizierung,
Zugriffsschutz,
Kryptographie,
Information
Leakage,
Fehlerbehandlung,
Konfiguration,
...

Schwachstellen auf Codeebene

Manuell: Code Review

Autom: Static App. Security Testing (SAST)

Application Security Testing Tools



OWASP

The Open Web Application Security Project

- Dynamic App. Security Testing (DAST)
- Static App. Security Testing (SAST)
- Hybride Ansätze



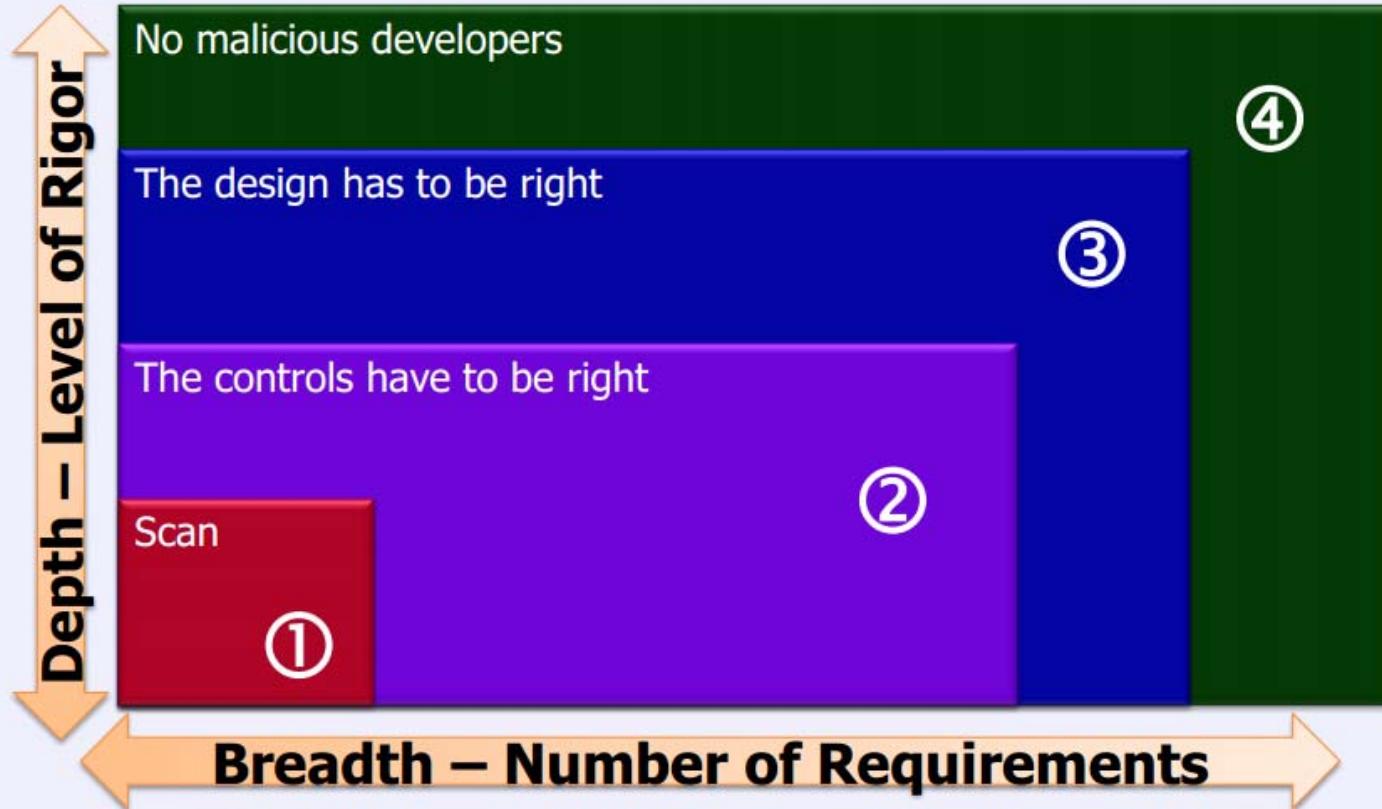
Tools vs. Manuelle Verifikation



OWASP

The Open Web Application Security Project

Quelle: OWASP ASVS 2009



Tools liefern keine Schwachstellen, nur Findings. Eine manuelle Verifikation und Bewertung ist immer erforderlich!

Häufige Vorbehalte



OWASP

The Open Web Application Security Project

- Zu teuer
- Viele False Positives (Schwachstellen die keine sind)
- Viele False Negatives (Schwachstellen die nicht gefunden werden)*
- Unzureichende Abdeckung
- Niemand der sie bedienen kann
- Gefährdet die Stabilität meiner Anwendung

* Vadim Okun. Aurelien Delaitre, Paul E. Black , The Second Static Analysis Tool Exposition (SATE), Special Publication 500-287, Juni 2010, http://samate.nist.gov/docs/NIST_Special_Publication_500-287.pdf



OWASP

The Open Web Application Security Project

10 Gründe,
warum Tools
trotzdem wichtig sind ...



OWASP

The Open Web Application Security Project

1. Anwendungen werden immer größer

- Große Webanwendungen > 100 KLOC
- Ein Security Reviewer schafft im Schnitt 1 KLOC pro Tag

2. Anwendungen ändern sich laufend

- Agile Entwicklung
- Schnelle Änderungen durch Änderungen im Deployment
- Laut Whitehat, wurden in 2012 im Schnitt 56 kritische Schwachstellen in Webanwendung eingebracht.*
- Änderungen von Anwendungen und deren Einbindung im Betrieb (Web 2.0)

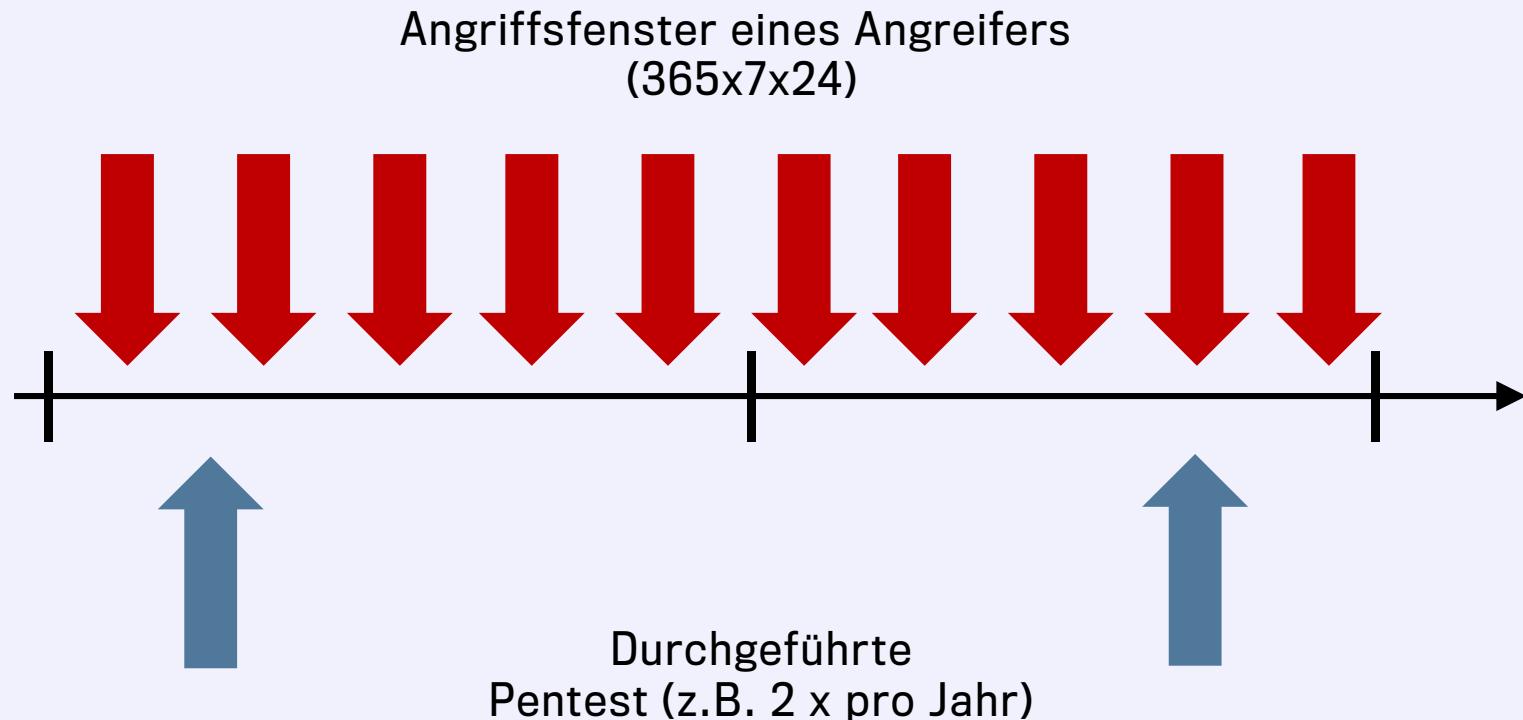
3. Security-Experten sind teuer und rar



* <http://de.slideshare.net/duncant75/whitehat-security-website-security-statistics-report-may-2013>



4. Tools arbeiten schnell und können kontinuierlich ausgeführt werden



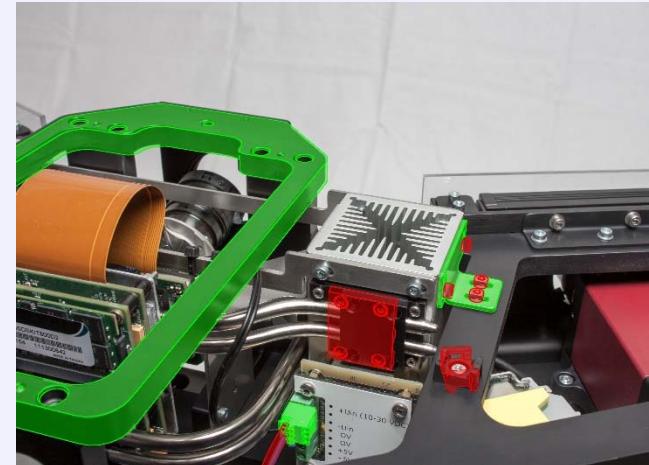
Warum Tools? (3)



OWASP

The Open Web Application Security Project

5. Tools können manuelle Analysen deutlich wirkungsvoller (bzw. überhaupt erst durchführbar) machen.
6. Mittels Tools lässt sich die Einhaltung spezifischer Policy's / Vorgaben laufend prüfen.
7. Tools ermöglichen nachvollziehbare, objektive und standardisierte Analysen.



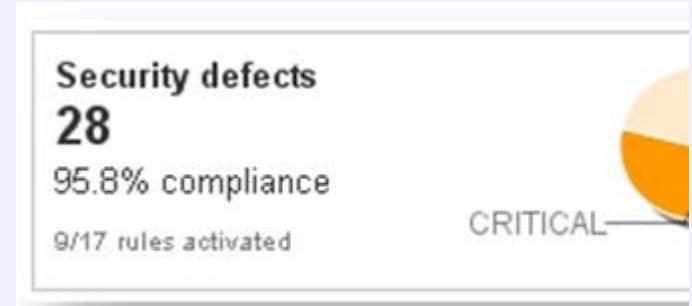
www.iff.fraunhofer.de



OWASP

The Open Web Application Security Project

8. Tools ermöglichen Nachhaltigkeit und kontinuierliche Verbesserung.
9. Tools ermöglichen die Ermittlung von Sicherheitskennzahlen (Metriken, KPIs).
10. Tools lassen sich in andere Tools integrieren (=> QA)



Limitationen von Tools müssen verstanden,
aber auch Chancen durch deren Einsatz
genutzt werden!



OWASP

The Open Web Application Security Project

Dynamic Application Security Testing (DAST)

= Tools, die eine laufende Anwendung auf potentielle Sicherheitsprobleme hin untersuchen

Sicherheitsprobleme, die DAST-Tools finden können...



OWASP

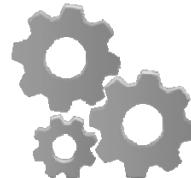
The Open Web Application Security Project

Implementierungs-Fehler



SQL Injection*
XSS*
Known Vuln.
etc.

Konfigurationsfehler



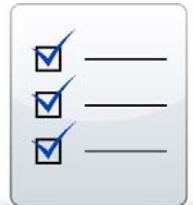
Error
Handling*
TLS-Config
etc.

Ungültige TLS-Zerts



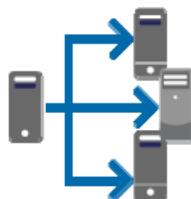
Abgelaufene
oder invalide
TLS/SSL-
Zertifikate

Sicherheitsanforderungen



Loginversuche
Passwortlänge
Access
Controls
etc.

Deploymentfehler



Testzugänge
SVN-
Dateien
etc.

Malware



Schadcode auf
den eigenen
oder in
dynamisch
eingebundenen
Seiten
(Werbebanner)

* Low Hanging Fruits

Pentesting Tools - Burp



OWASP

The Open Web Application Security Project

The screenshot shows two instances of the Burp Suite interface side-by-side.

Left Window (Repeater Tab):

- Toolbar:** Burp, Intruder, Repeater, Window, Help.
- Sub-Toolbar:** Target, Proxy, Spider, Scanner.
- Buttons:** Intruder, Repeater, Sequencer, Decoder, Comparer, Options, Alerts.
- Status Bar:** 1 x, 2 x, 3 x, 4 x, ...
- Bottom Buttons:** Target, Positions, Payloads, Options.

Content Area:

- Section:** **Payload Positions**
- Description:** Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions – see help for full details.
- Attack type:** Sniper.
- Request Preview:** GET /bookDetails.html?id=\$735\$&userId=\$341\$
- Buttons:** Add §, Clear §, Auto §, Refresh.
- Search:** Type a search term, 0 matches, Clear.
- Metrics:** 3 payload positions, Length: 409.

Right Window (Proxy Tab):

- Toolbar:** Burp, Intruder, Repeater, Window, Help.
- Sub-Toolbar:** Repeater, Sequencer, Decoder, Comparer, Options, Alerts.
- Buttons:** Target, Proxy, Spider, Scanner, Intruder.
- Status Bar:** 1 x, 2 x, 3 x, 4 x, ...
- Bottom Buttons:** Target, Positions, Payloads, Options.

Content Area:

- Configuration:** Payload set: 1, Payload count: 512, Payload type: Simple list, Request count: 1.536.
- Section:** **Payload Options [Simple list]**
- Description:** This payload type lets you configure a simple list of strings that are used as payloads.
- Buttons:** Paste, Load ..., Remove, Clear.
- Payload List:** !@#\$%^&*%#@#\$%\$@#\$%^*((), !@#0%#0##018387@#0AA**(), -, "or "a"="a", "or "x"="x", "or 0=0 #", "or 0=0 --", "or 1=1 or ""="".

Außerdem: OWASP ZAP, Fiddler, (OWASP Webscarab), diverse Browser-Plugins

Web Security Scanner GUIs (kommerziell)



OWASP

The Open Web Application Security Project

The screenshot shows the HP WebInspect interface. The top menu includes File, Edit, View, Tools, Scan, AMP, Reports, and Help. The toolbar has buttons for New, Open, Compliance Manager, Policy Manager, Report, Schedule, SmartUpdate, Start / Resume, Pause, Stop, Audit, and Rescan. The main window displays a 'Scan Dashboard' with 'Crawl 56 of 56' and 'Audit 248 of 248'. A 'Vulnerabilities' chart shows counts for Critical (22), High (3), Medium (6), Low (3), Info (34), and Best Practices (11). Below the chart is a table of 'Attack Type' vs 'Attacks' with rows for Manipulation, Exploratory, and Other. On the left, a 'Site' tree view shows various URLs under categories like (Post) and (Query). A 'Scan Info' sidebar lists crawl, audit, session, and activity details. A 'Duplicates' table at the bottom lists multiple instances of Cross-Site Scripting vulnerabilities. Navigation tabs include Site, Sequence, Search, Step Mode, Vulnerabilities, Information, Best Practices, Scan Log, and Server Information.

- HP WebInspect
- IBM AppScan
- NTSpider
- Accunetix
- Burp

Web Security Scanner (kostenfrei)



OWASP

The Open Web Application Security Project

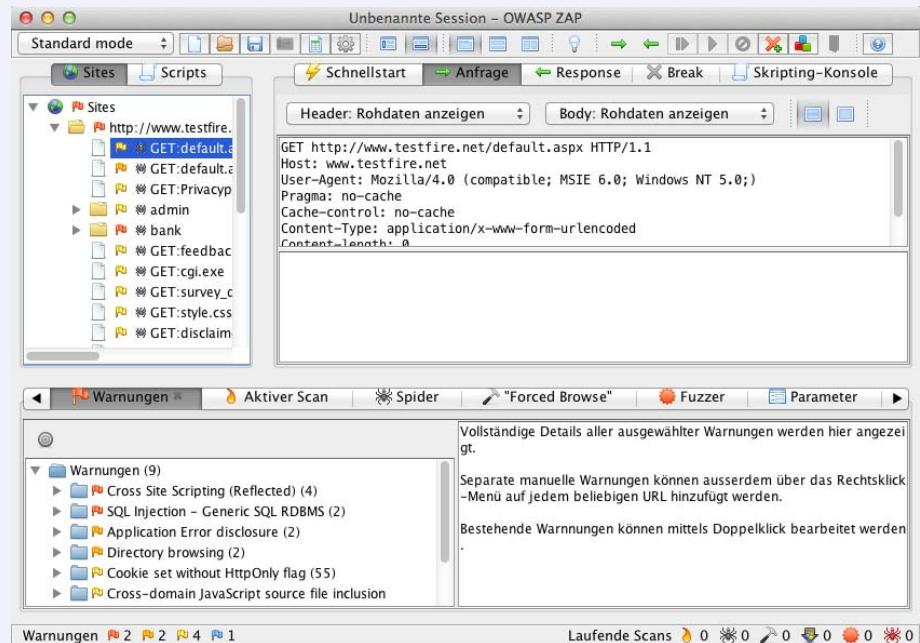
The screenshot shows the OWASP ZAP interface. The top bar indicates "Unbenannte Session - OWASP ZAP". The left sidebar lists "Sites" and "Scripts". Under "Sites", there is a tree view of a testfire.net site with various URLs like default.aspx, Privacy, admin, bank, feedbac, cgi.exe, survey_c, style.css, and disclaim. The main pane displays an "Anfrage" (Request) and "Response" tab. The request tab shows a GET request to http://www.testfire.net/default.aspx with headers including Host, User-Agent, Pragma, Cache-control, Content-Type, and Content-length. The response tab shows the raw HTML content of the page. Below the main pane is a "Warnungen" (Warnings) tab, which lists 9 findings: Cross Site Scripting (Reflected) (4), SQL Injection – Generic SQL RDBMS (2), Application Error disclosure (2), Directory browsing (2), Cookie set without HttpOnly flag (55), and Cross-domain JavaScript source file inclusion.

- OWASP ZAP
- w3af
- skipfish
- Nikto / Wikto
- SSLScan / SSL Labs

Mittels ThreadFix
lassen sich aus
Scanner-
Ergebnissen
Virtual-Patching-
Regeln für
ModSecurity
generieren.



- Aktive und Passive Scans
- Sehr aktive Community (laufende Weiterentwicklung)
- UI und Daemon Mode
- Offenes Design:
 - API
 - Eigene Add-Ons
 - Eigne Skripte
 - Programmaufrufe



Vorsicht vor dem Einsatz auf produktiven Systemen!

DAST as a Service: Veracode



OWASP

The Open Web Application Security Project

Veracode Platform X

<https://analysiscenter.veracode.com/auth/index.jsp#ReviewResultsDynamicFlaws:29305:88935:186005:171177:186966::>

Analytics Applications Services Policies eLearning Directory Matthias Rohr: Secodis GmbH Admin Ops Your Account Help Logout

Application: test

Scan: Static | Dynamic: 16 Dec 2013 Dynamic (Links Report)

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') (CWE ID: 89)

Request Readout Show: Fix First Analyzer Flaw Details None Download Flaw Details

Flaw URL: <http://www.defensiveweb.org/13154692451/DVWA/vulnerabilities/brute/?username=vcode%27&R4FJxzZz=Veracode&Login=Login>

Method: GET
Protocol: Http

Details Request/Response

Request:

```
GET /13154692451/DVWA/vulnerabilities/brute/?username=vcode%27&R4FJxzZz=Veracode&Login=Login HTTP/1.1
Host: www.defensiveweb.org
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:21.0) Gecko/20100101 Firefox/21.0
Cookie: PHPSESSID=2db3f97582b12768c7ad0ee8ea3994f; security=low
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://www.defensiveweb.org/13154692451/DVWA/vulnerabilities/brute/
Connection: keep-alive
```

Vulnerable Parameter: username
Injected Value: vcode'
Original Value: vcode

Response:

```
HTTP/1.1 200 OK
Date: Mon, 16 Dec 2013 22:52:03 GMT
Server: Apache
X-Powered-By: PHP/5.4.23
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 190
Keep-Alive: timeout=2, max=188
```

Font re-size: A A A

Flaws: 34 of 34 Rows/Page: 50

All 34 Flaws	0 Flaws	Search: Id	GO	CLEAR	Take Selected Action	GO	?
CWE ID & Name							
ID	Sev	Vuln Parameter	Path	Status	Mitigation		
4	4	username	/13154692451/DVWA/vulnerabilities/brute/?username=vcode%27&R4FJxzZz=Veracode&Login=Login	Open	none		
13	4	id	/13154692451/DVWA/vulnerabilities/sql?id=1%27&Submit=Submit	Open	none		
20	4	id	/13154692451/DVWA/vulnerabilities/sql_bind/?id=1%27%3d%28select%20%29from%28select%28sleep%2815%29%29%29as0x41%29%3d%27&Submit=Submit	Open	none		
25	4	-	/13154692451/DVWA/	Open	none		
1	3	-	/13154692451/DVWA/login.php	Open	none		
402	2	id	/13154692451/DVWA/vulnerabilities/view_source.php?id=%22%3c%2fscript%3e%3cscript%3epholidCallback%287548494245%29%3c%2fscript%3e&security=low	Open	none		
83	3	-	www.defensiveweb.org	Open	none		
326	3	page	/13154692451/DVWA/vulnerabilities/f/	Open	none		
5	3	page	?page=%2f.%2f..%2f..%2f..%2f..%2f..%2fetc%2fpasswd	Open	none		
6	3	security	/13154692451/DVWA/vulnerabilities/xss_r/	Open	none		
8	3	mtxMessage	/13154692451/DVWA/vulnerabilities/xss_s/	Open	none		
22	2	page	/13154692451/DVWA/vulnerabilities/	Open	none		

© VERACODE, INC. 2006 - 2013 | USAGE GUIDELINES | RESPONSIBLE DISCLOSURE POLICY | VERACODE SUPPORT
Last account activity on 12/16/13 3:24 PM EST from IP: 92.224.64.80 For use under U.S. Pat. Nos. 8,363,155 and 8,854,924, and patents pending.

Weiterhin: Whitehat Sentinel, NTOSpider On-Demand

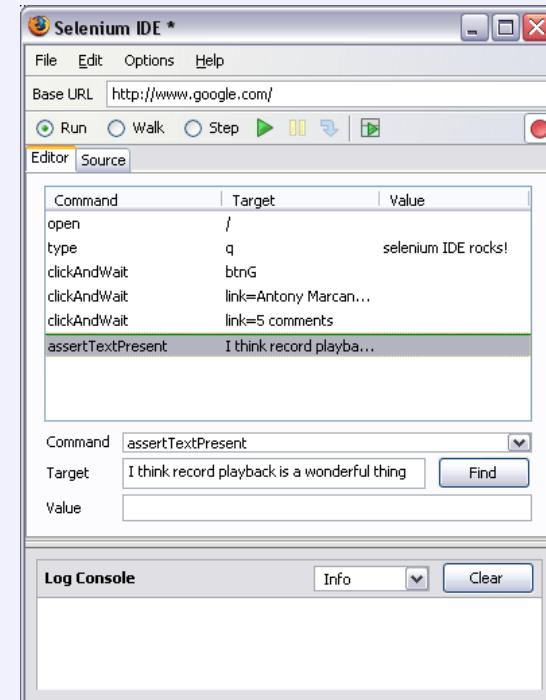
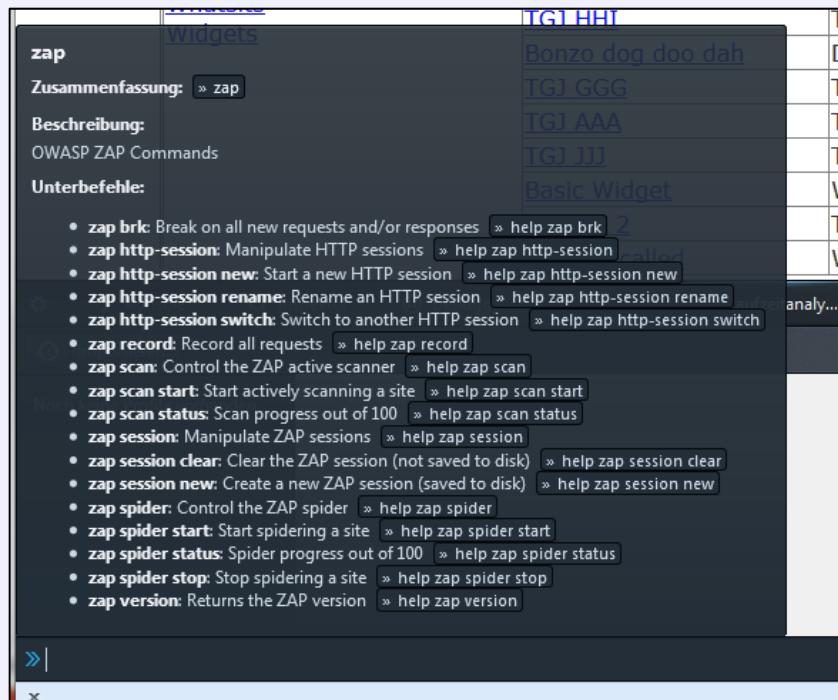
Comming Next ... Bessere Tool Integration



OWASP

The Open Web Application Security Project

- SAST – DAST - Integration
- QA-Tool-Integration (z.B. Selenium, Unit Tests, QuickTest Pro)
- Browser-Integration (Plug-n-Hack, FF Firebug, Vendor Plugins)





OWASP

The Open Web Application Security Project

Static Application Security Testing (SAST)

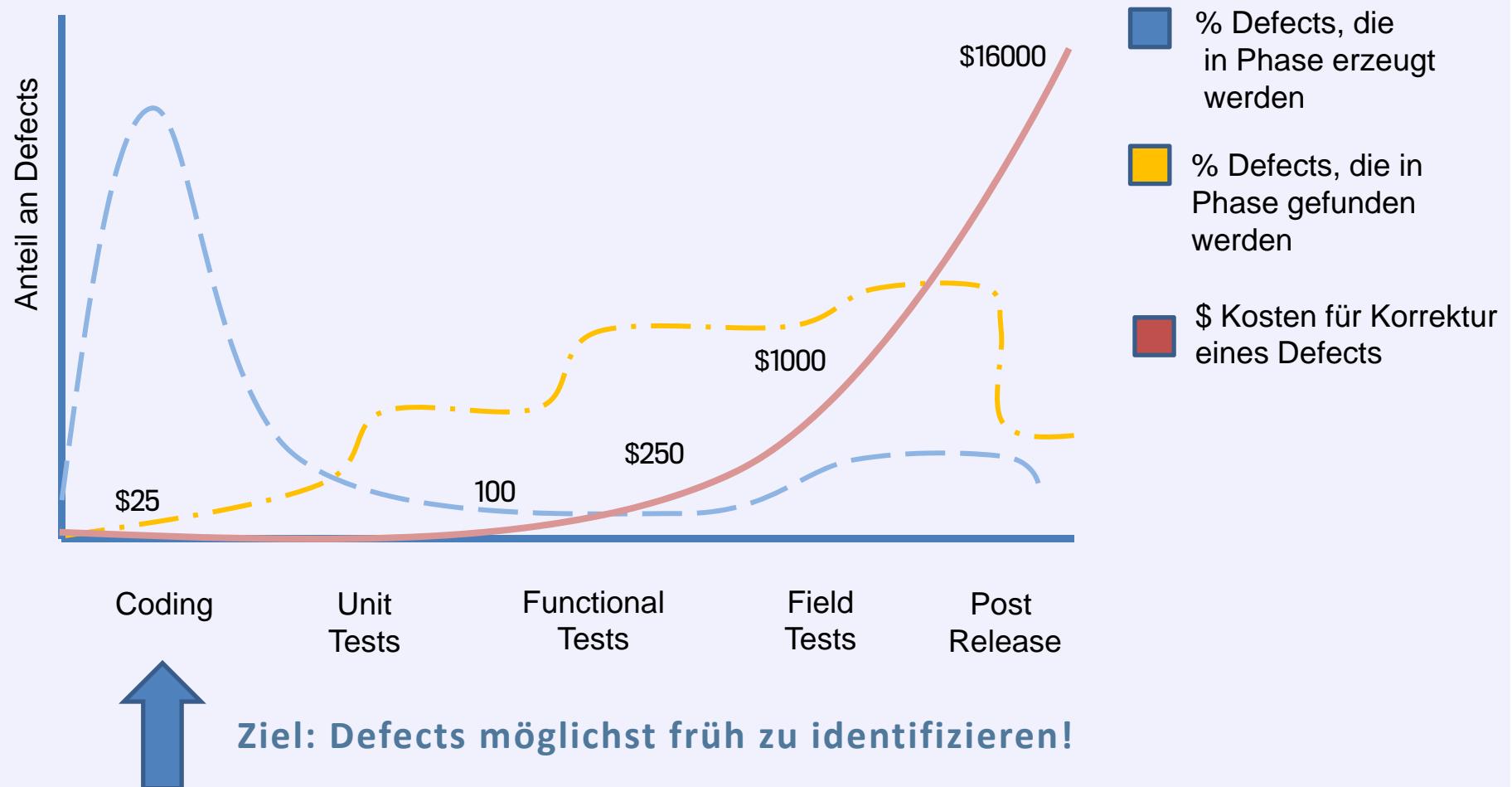
- = Tools, die entweder den Sourcecode, Binärkode oder Bytecode auf potentielle Sicherheitsmängel hin untersuchen

Sicherheit im SDLC



OWASP

The Open Web Application Security Project



Sicherheitsprobleme, die SAST-Tools finden können...



OWASP

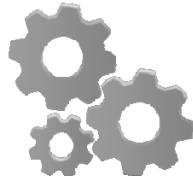
The Open Web Application Security Project

Implementierungs-Fehler



- SQL Injection / XSS
- Buffer Overflows / Race Conditions
- Unsichere APIs / Aufrufe
- etc.

Konfigurationsfehler



- Fehlerbehandlung
- Validierung
- Hartkodierte Passw.
- etc.

Security Anforderungen



- Secure Coding Guidelines / Policies
- Architektur-Constraints
- Etc.

Sensible Änderungen



Unautorisierte
Änderung an
sensiblen
Dateien (z.B.
Bibliotheken)



Lexikalische Analyse

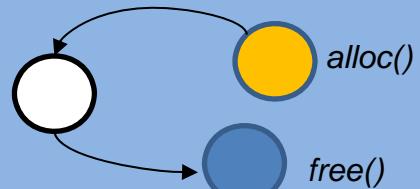
`grep -i -r "exec()" *`

Identifizierung verdächtiger Zeichenketten im Code
(Reguläre Ausdrücke)

Beispiele

- Hartkod. Passwörter
- Entwicklerkommentare
- Kommentierter Code

Kontrollflussanalyse

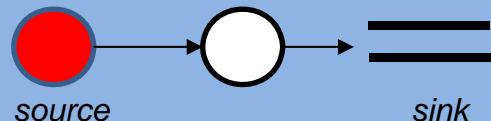


Identifiziert fehlerhaft freigegebenen Speicher, unsichere Sequenz von Funktionsaufrufen, etc.

Beispiele

- Buffer Overflows
- Race Conditions

Datenflussanalyse



Dient dem Auffinden von Fehlern in der Datenvalidierung

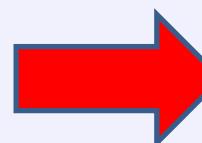
Beispiele

- Cross-site Scripting
- SQL Injection

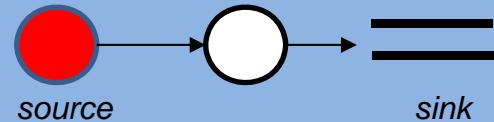
Weiterhin: Semantische Analyse, Strukturanalyse, Konfigurationsanalysen



- Die Datenflussanalyse ist das wichtigste Analyse-Verfahren eines SAST-Tools!
- Die meisten kostenfreien Tools (PMD, Findbugs, etc.) bieten keine (bzw. nur eine eingeschränkte) Datenflussanalyse.



Datenflussanalyse



Dient dem Auffinden von Fehlern in der Datenvalidierung

Beispiele

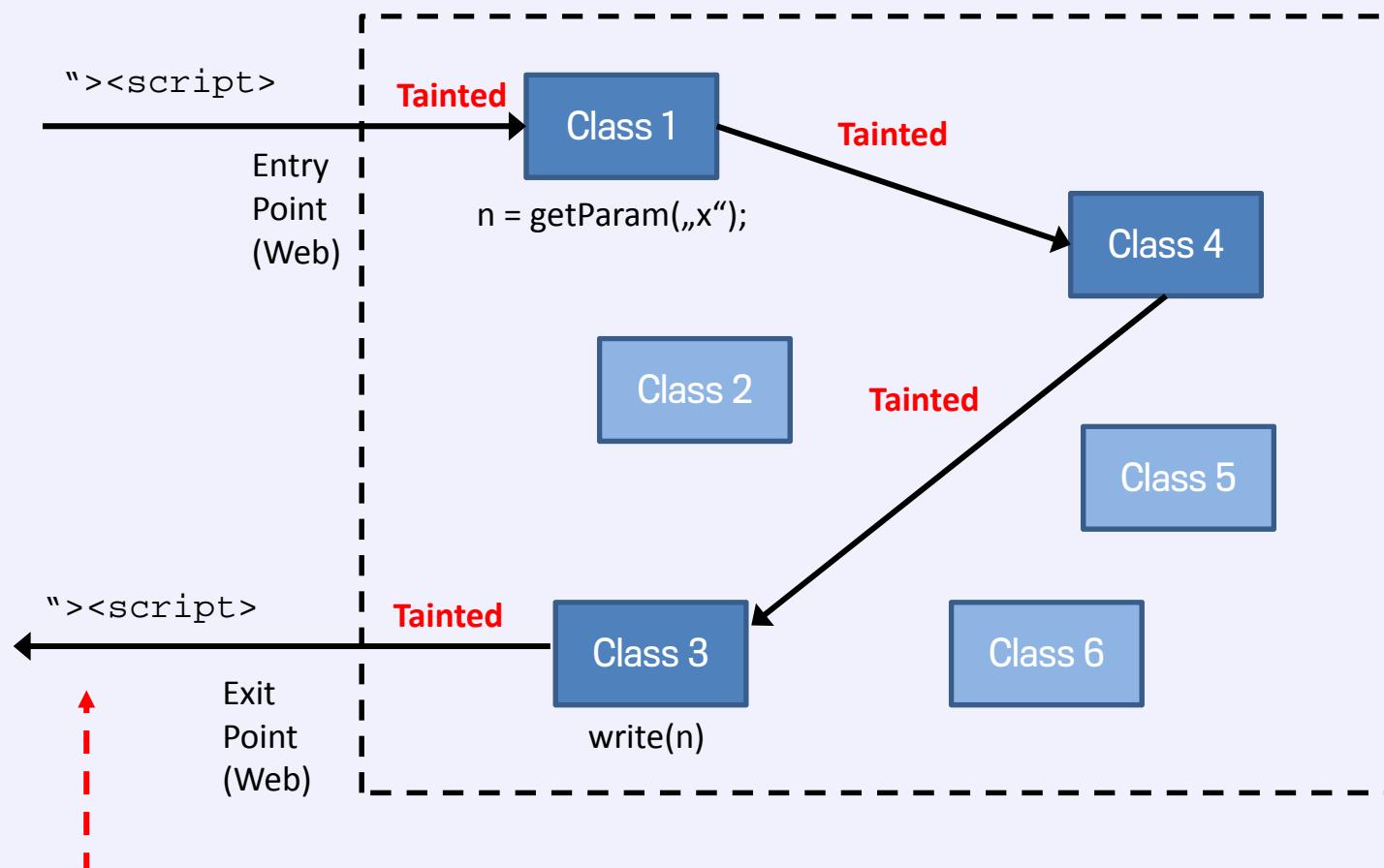
- Cross-site Scripting
- SQL Injection

Datenflussanalysen (aka Taint Analyse)



OWASP

The Open Web Application Security Project



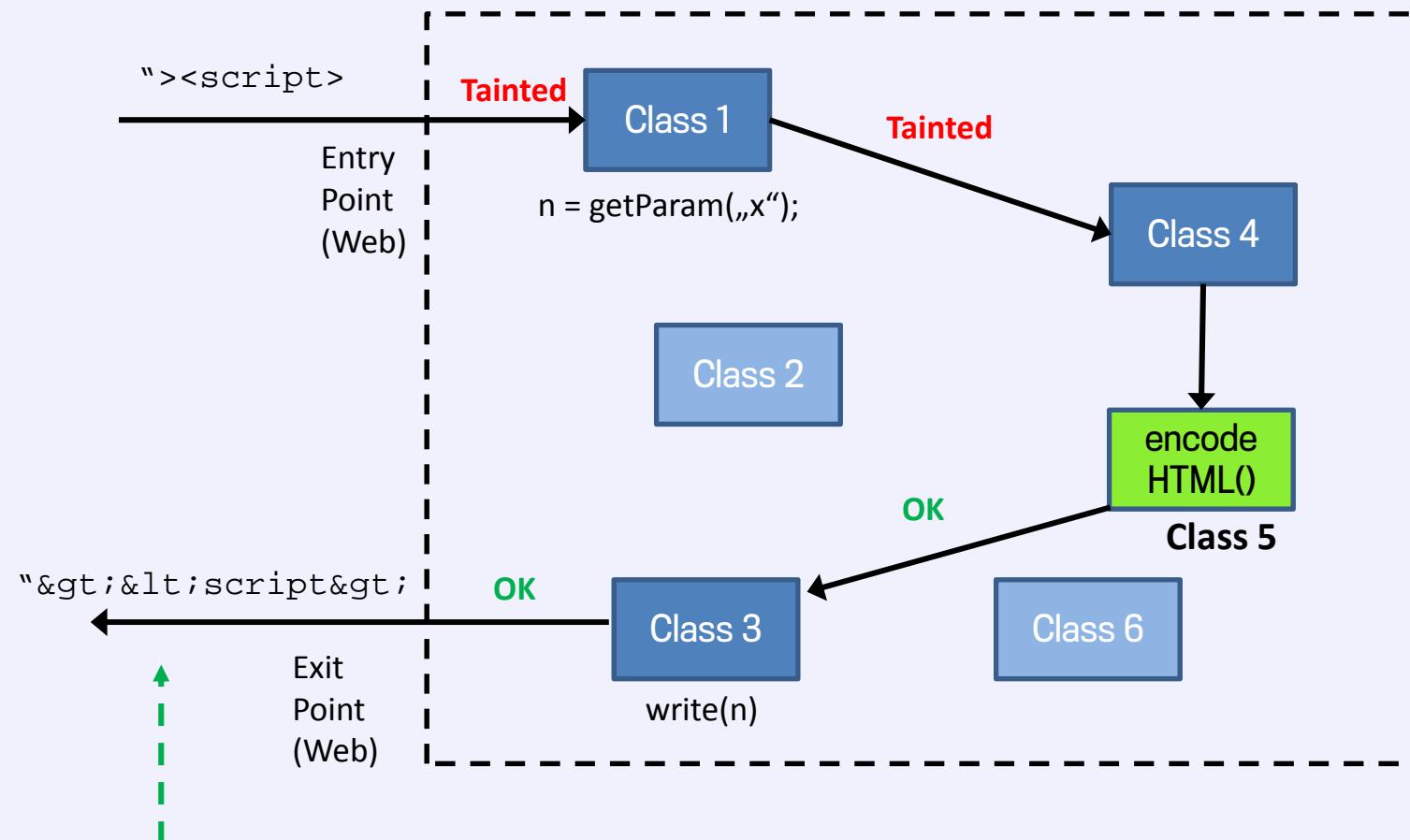
Cross-site Scripting!

Datenflussanalysen (aka Taint Analyse)



OWASP

The Open Web Application Security Project



kein Cross-site Scripting!

HP Fortify - Auditworkbench



OWASP

The Open Web Application Security Project

WebGoat5.0 – JavaSource/org/owasp/webgoat/lessons/WeakSessionID.java – Audit Workbench

AUDIT WORKBENCH FORTIFY

Summary | Audit Guide | Scan | Reports

Filter Set: Security Auditor View | **My Issues**

Critical (463)

Group By: Category

Analysis Evidence

Rule ID: DBDBBF6-DE26-4FC5-8347-D48032B2BF5D
Taint Flags: WEB, XSS
Direct Function Call: org.apache.ecs.html.Input.Input()

Code View:

```
119 .addElement("Studio RTA - Laptop/Reading Cart with");
120 tr.addElement(new TDC().addElement("69.99").setAlign("right"));
121 tr.addElement(new TDC().addElement(
122     new Input(Input.TEXT, "QTY1", s.getParser()
123         .getStringParameter("QTY1", "1")))
124         .setAlign("right"));
125 quantity = s.getParser().getFloatParameter("QTY1", 1.0f);
126 total = quantity * 69.99f;
127 runningTotal += total;
128 tr.addElement(new TDC().addElement("$" + total));
129 t.addElement(tr);
130 tr = new TRC();
131 tr.addElement(new TDC()
132     .addElement("Dyne - Traditional Notebook Case"));
133 tr.addElement(new TDC().addElement("27.99").setAlign("right"));
134 tr.addElement(new TDC().addElement("1.00").setAlign("right"));
```

Functions

Show: All | Group by: package | Include external API | Legend...

Top-level functions

- /
- /lessons/ConfManagement
- /lessons/CrossSiteScripting
- /lessons/General
- /lessons/RoleBasedAccessControl
- /lessons/SQLInjection
- http://java.sun.com/JSP/Page
- java.io
- java.lang
- java.net
- java.nio
- java.nio.charset
- java.security
- java.sql
- java.text
- java.util
- java.util.regex
- javax.crypto
- javax.crypto.spec
- javax.servlet
- javax.servlet.http
- javax.servlet.jsp
- javax.xml.namespace
- org.apache.axis
- org.apache.axis.client
- org.apache.catalina
- org.apache.catalina.users
- org.apache.ecs
- org.apache.ecs.html

Diagram View:

ReflectedXSS.createContent → ParameterParser.getStringParameter

```
graph TD; A[ReflectedXSS.createContent] --> B[ParameterParser.getStringParameter]; B --> C[Input(2)]; C -- sink --> D[Return];
```

Summary | Details | Recommendations | History | Diagram | Correlated Issues | Screenshots | Security

Checkmarx - Datenflussanalysen



OWASP

The Open Web Application Security Project

Results Graph

Graph Type: Full Graph Key Nodes Ends Show related data flows Show not exploitable flows Result State ▾

Scan Results

- Java
 - High
 - Code_Injection (9 : Found)
 - Reflected_XSS_All_Clients (193 : Found)
 - Second_Order_SQL_Injection (37 : Found)
 - SQL_Injection (386 : Found)
 - Stored_XSS (607 : Found)
 - Medium
 - Low
 - Info

JavaScript

 - High
 - Medium
 - Low

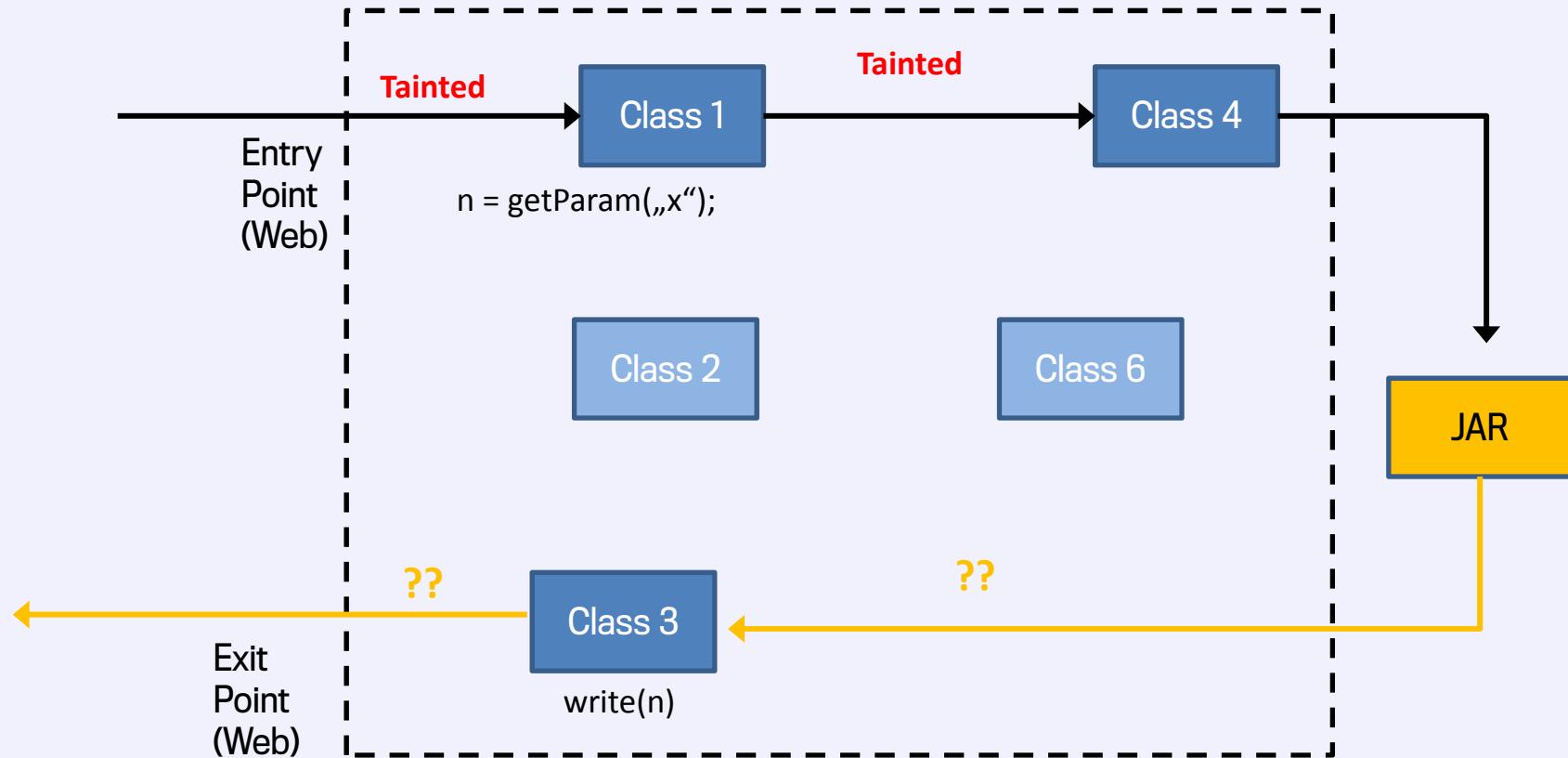
The data flow graph illustrates the flow of data from user input fields (e.g., Name, Email, Password) through various processing modules (e.g., Hashing, Encryption, Decryption, Logging) and databases (e.g., MySQL, Oracle, PostgreSQL, SQLite). High-risk vulnerabilities like Code Injection and SQL Injection are highlighted in red, showing their propagation through the system. The graph provides a visual representation of how data flows from the user interface into the database and back to the UI, identifying potential attack vectors.

Problem von Source Code Analysen



OWASP

The Open Web Application Security Project



Weitere Probleme: Dependency Injection, komplexe Validierungsfunktionen
(z.B. via Regulären Ausdrücken), ... Ergebnis: viele False Positives

Bytecodeanalysen mit Veracode



OWASP

The Open Web Application Security Project

Application: A Goat on the Web

Scan: Static: 11 Nov 2013 Static | Dynamic: 2 Oct 2013 Dynamic

Source Code View (HTML5 Version): Exec.java

Request Readout

Show: Fix First Analyzer Source Code Viewer None

MAX

Lines 1-544 Load Different File Goto Line Source History... ▾

```
282     ByteArrayOutputStream errors = new ByteArrayOutputStream();
283     ExecResults results = new ExecResults(command, input, successCode,
284         timeout);
285     BitSet interrupted = new BitSet(1);
286     boolean lazyQuit = false;
287     ThreadWatcher watcher;
288
289     try
290     {
291         // start the command
292         child = Runtime.getRuntime().exec(command);
293
294         // get the streams in and out of the command
295         InputStream processIn = child.getInputStream();
296         InputStream processError = child.getErrorStream();
```

Flaws: 268 of 268

Rows/Page: 50 < 1 2 3 4 5 6 >

All 268 Flaws	0 Flaws	Search: Id =	GO	CLEAR	Take Selected Action	GO	Help	
ID	Sev	Exp	CWE ID & Name	Module	Source	Count	Status	Mitigation
41	5	V.Likely	78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	WebGoat-5.0-with-jsp.war	Exec.java: 103	1	Open	none
76	5	V.Likely	78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	WebGoat-5.0-with-jsp.war	Exec.java: 292	1	Open	none
24	4	V.Likely	89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	WebGoat-5.0-with-jsp.war	Login.java: 149	1	Open	none
25	4	V.Likely	89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	WebGoat-5.0-with-jsp.war	UpdateProfile.java: 176	1	Open	none
37	4	V.Likely	89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	WebGoat-5.0-with-jsp.war	ViewProfile.java: 178	1	Open	none
73	4	V.Likely	89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	WebGoat-5.0-with-jsp.war	SqlNumericInjection.java: 130	1	Open	none

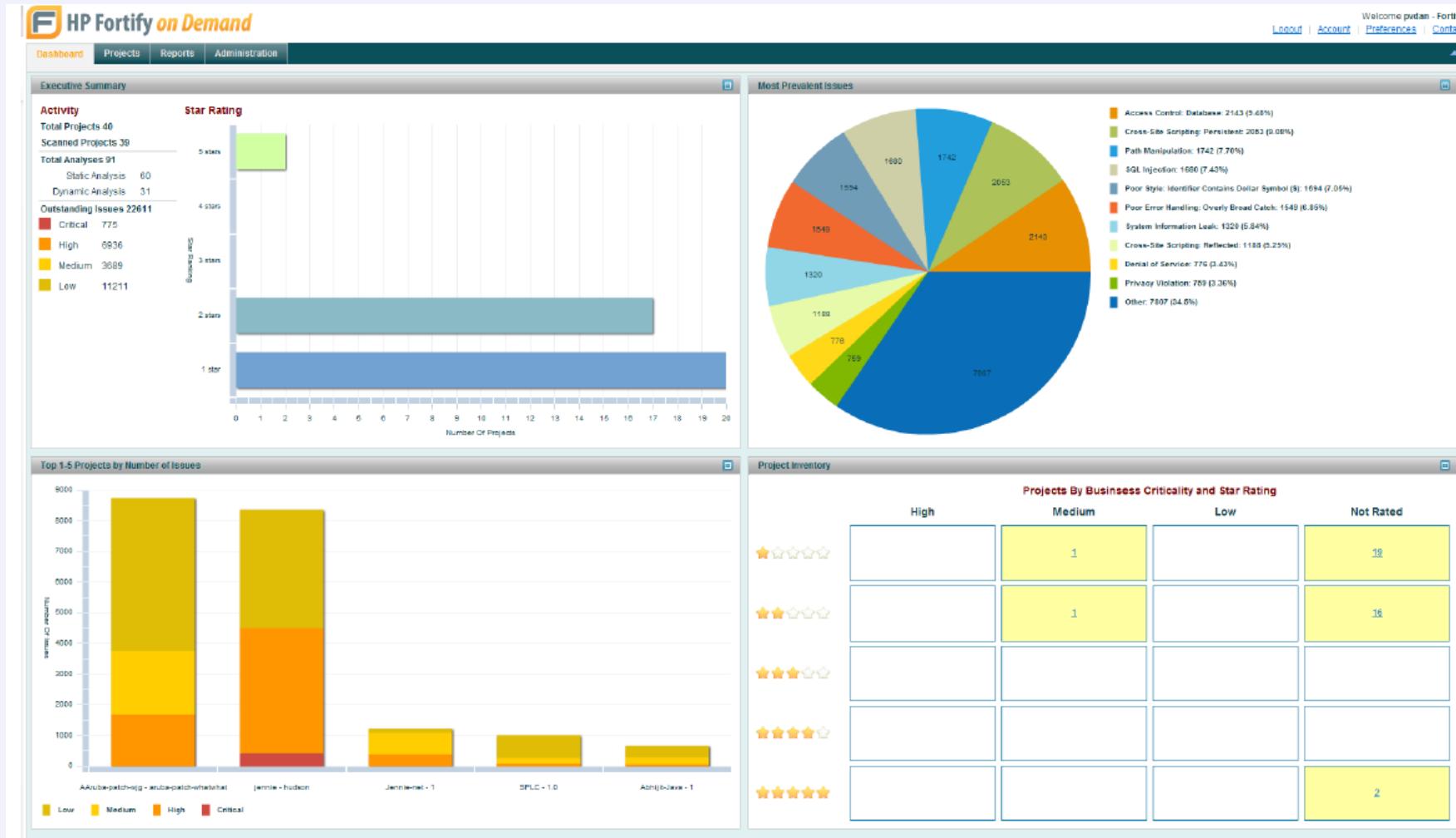
Flaws Call Stack

Fortify on Demand (FOD)



OWASP

The Open Web Application Security Project



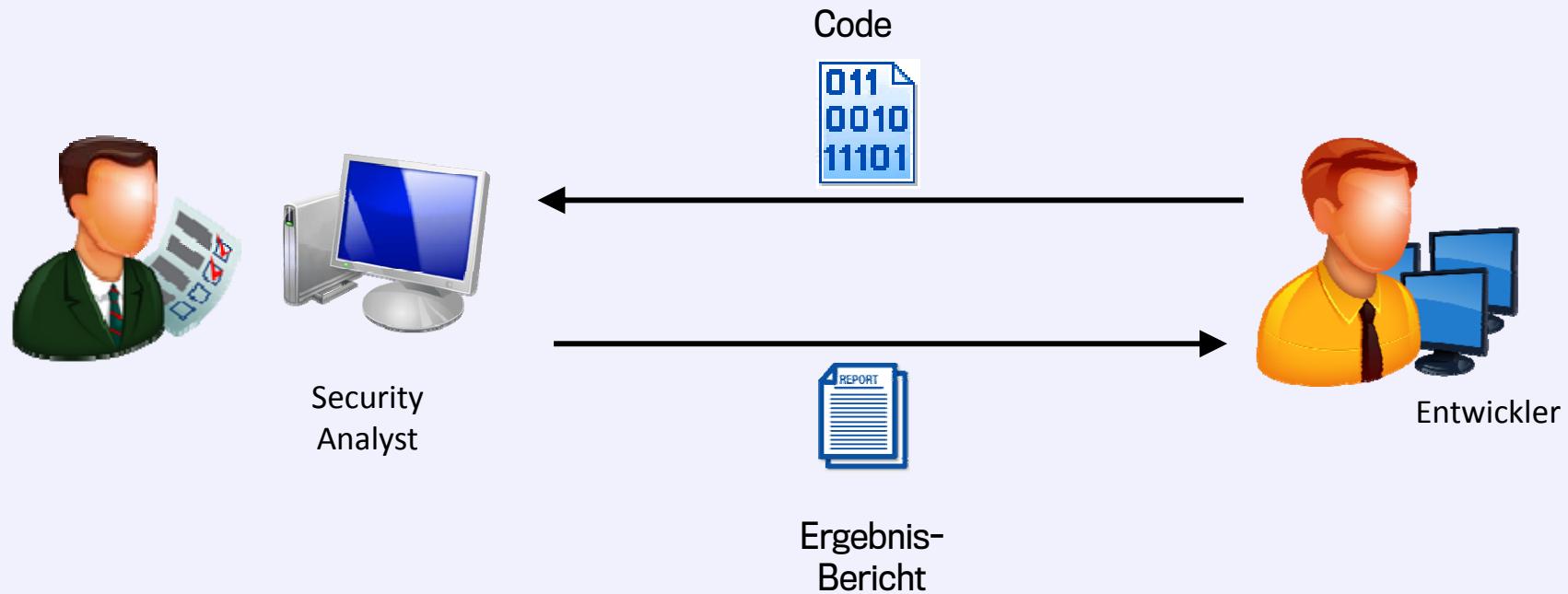
www.fortifymyapp.com

AST-Deployment: Tools der 1. Generation



OWASP

The Open Web Application Security Project

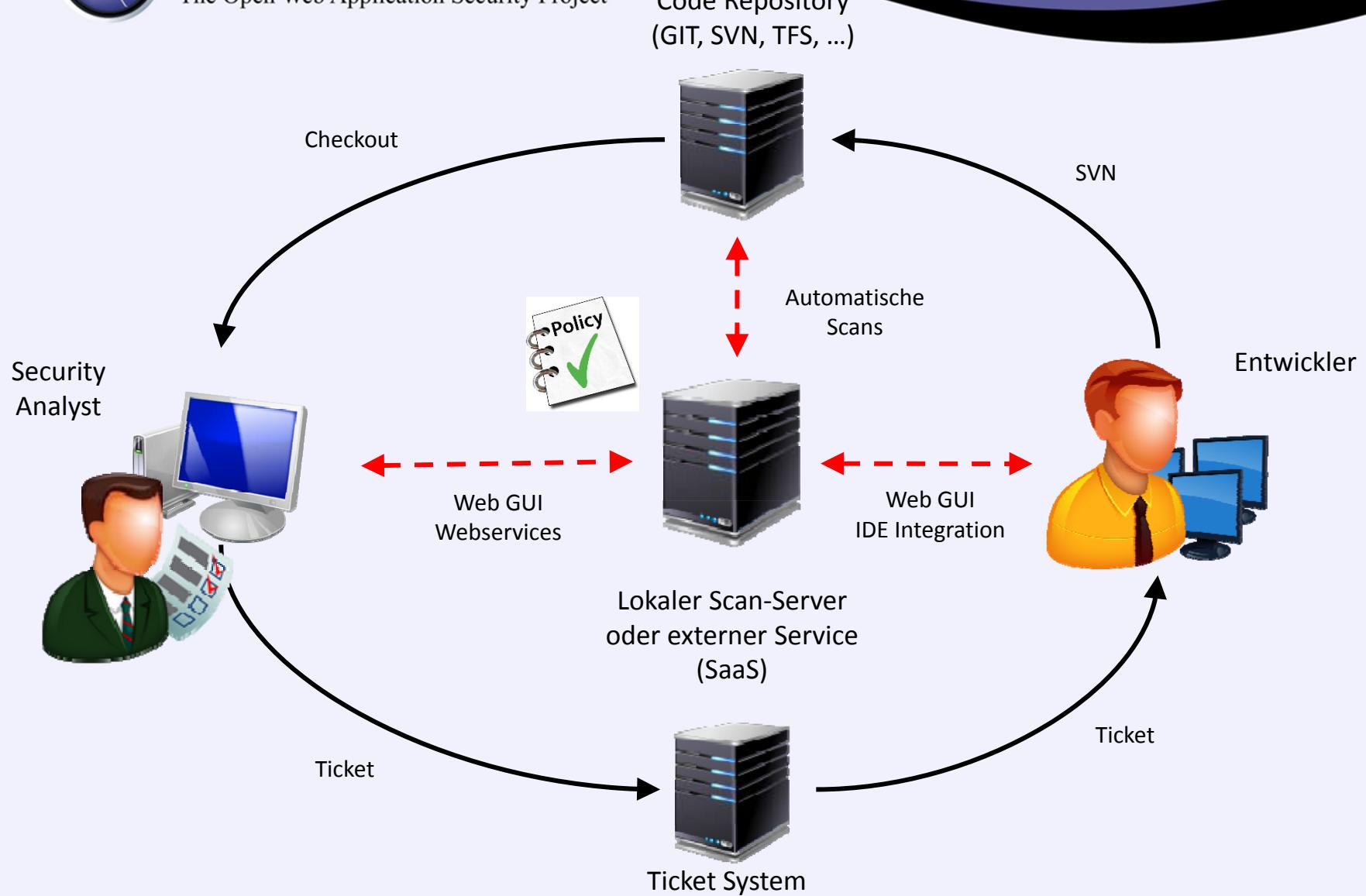


SAST Deployment: Heutige Integration



OWASP

The Open Web Application Security Project



On-Premise vs. Off-Premise



OWASP

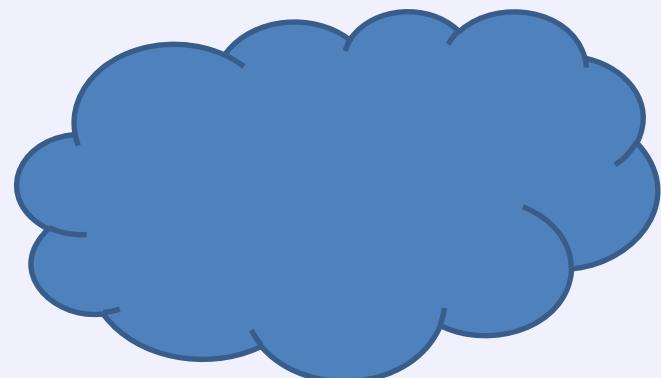
The Open Web Application Security Project

On-Premise
(Lokal installiert)



Vs.

Off-Premise
(Cloud-Based)



SAST Deployment: Heutige Integration (2)



OWASP

The Open Web Application Security Project

Build Server
(Jenkins, Hudson,
Bamboo, ...)



Code Repository



Automatischer
Scan

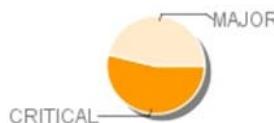


Scan Server /
SaaS



Entwickler

Security defects
28
95.8% compliance
9/17 rules activated



Review



QA

Custom Rules



OWASP

The Open Web Application Security Project

Custom Rules Editor

File Edit Help

Outline

Name: \Fortify\CRE-3.90\rules\custom-rule

Version: 1.0

Description: Description of RulePack.

Group by: Taint Flags/Rule Type

Showing 1 of 1 rules

<?xml version="1.0" encoding="UTF-8"?>
<RulePack xmlns="http://www.fortifysoftware.com/schema/rules">
 <RulePackID>950F4B5A-85D0-4954-97D5-35C3F0285057</RulePackID>
 <SKU>SKU-
 <Name>
 <![CDATA[
 Fortify\CRE-3.90\rules\custom-rule]]>
 </Name>
 <Version>1.0</Version>
 <Description>
 Description of RulePack
 </Description>
 <Rules version = "3.16">
 <RuleDefinitions>
 <CustomDescriptionRule formatVersion="3.15">
 <RuleID>D40B319C-F9D6-424F-9D62-BB1FA3B3C645</RuleID>
 <RuleMatch>
 <Category>
 <Value>SQL Injection</Value>

Query Results Comments Debug Messages

Queries

Find_Methods
Find_Outputs
Find_Parameters
Find_Password_Strings
Find_Passwords
Find_Personal_Info
Find_Read
Find_ReDoS
Find_Regex
Find_Remoting
Find_Replace
Find_Sanitize
Find_Session_Create

Query Source

```
1 result = Find_Passwords() +  
2   All.FindByShortName("Credit", false) +  
3   All.FindByShortName("Account", false) +  
4   All.FindByShortName("SSN", false) +  
5   All.FindByShortName("SocialSecurity", false);
```

Run Query
Properties...
Show Description
Override
Delete

Fortify

Checkmarx

Entwickler IDE Integration



OWASP

The Open Web Application Security Project

The screenshot shows a Microsoft Visual Studio interface with several toolbars and toolbars. The top toolbar includes File, Bearbeiten, Ansicht, Projekt, Erstellen, Debuggen, Team, Daten, Extras, Test, Analyse, and a Debug dropdown. A secondary toolbar on the right contains HP Fortify, Fenster, and Hilfe buttons. The main window displays a solution named 'BlogEngine.NET 1.3' with two projects: 'BlogEngine.Core' and 'BlogEngine.Web'. The code editor shows a C# file with methods like Page.cs and contact.aspx.cs. A context menu is open over the code, listing options such as 'Build With Veracode Settings', 'Rebuild With Veracode Settings', 'Precompile Web Projects', 'Upload Build...', 'Download Results...', 'View Results', 'Import Results...', 'Options...', 'Help', 'About', and 'Check For Updates...'. Another context menu is visible on the right side of the screen, listing 'Analyze Source Code of Solution', 'Analyze Component Project', 'Open Collaborative Audit ...', 'Open Audit Project ...', 'Options ...', 'Connect to SSC Server', and 'Remediation Options...'. The bottom status bar indicates 'Ready'.

Fortify Visual Studio Plugin

Veracode Visual Studio Plugin



- Qualität der Findings / Scan-Engine (False-Negatives / -Positives)?
- On-Site oder Off-Site („Cloud-Based“)?
- Sourcecode oder Bytecode?
- Integration: Kann das Tool in meine QA / Entwicklung integrieren?
- Unterstützte Sprachen: Werden alle relevanten Technologien (Sprachen, Frameworks) unterstützt, wenn ja wie vollständig?
- Möglichkeit eigener Scan Policys / Custom Regeln?
- Bedienbarkeit: Wer bedient das Tool (QA, Security Experten, Entwickler), wie gut lassen damit Analysen durchführen?
- Customizing: Wie viel ist wünschenswert, wieviel handelbar?
- Kosten!
-

**Tools sind unterschiedlich für bestimmte Organisationen /
Anforderungen geeignet und sollten daher immer vor einem
Kauf evaluieren!**



OWASP

The Open Web Application Security Project

Fazit



1. Der Einsatz von Tools ist häufig wichtig (bzw. erforderlich) um die Sicherheit einer (Web-)Anwendung zu gewährleisten.
2. Jedes Tool hat eine individuelle Eignung und Limitationen, die Verstanden werden müssen.
3. Ohne entsprechende Prozesse, Verantwortlichkeiten und Mitarbeiter die es bedienen können, ist ein (intern eingesetztes) Tool wertlos.

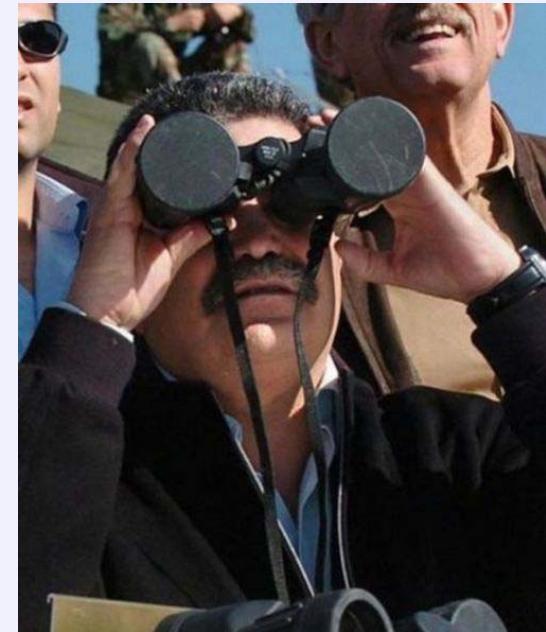




OWASP

The Open Web Application Security Project

4. Jedes Tool ist nur so gut, wie der, der es bedient.
5. Jedes Tool kann stets nur eine Ergänzung zu manueller Verifikation darstellen, keinen Ersatz!
6. Insbesondere Enterprise SAST Tools erfordern ein Auswahl- und Einführungsverfahren (Pilot, On Boarding, etc.).





OWASP

The Open Web Application Security Project

Danke!

Fragen?