Tool-basierte Web Security

Matthias Rohr

20. März 2014

OWASP

The Open Web Application Security Project

About Me

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

secodis
APPLICATION SECURITY SERVICES

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 oft he most serious economic and national security challenges we face as a nation“

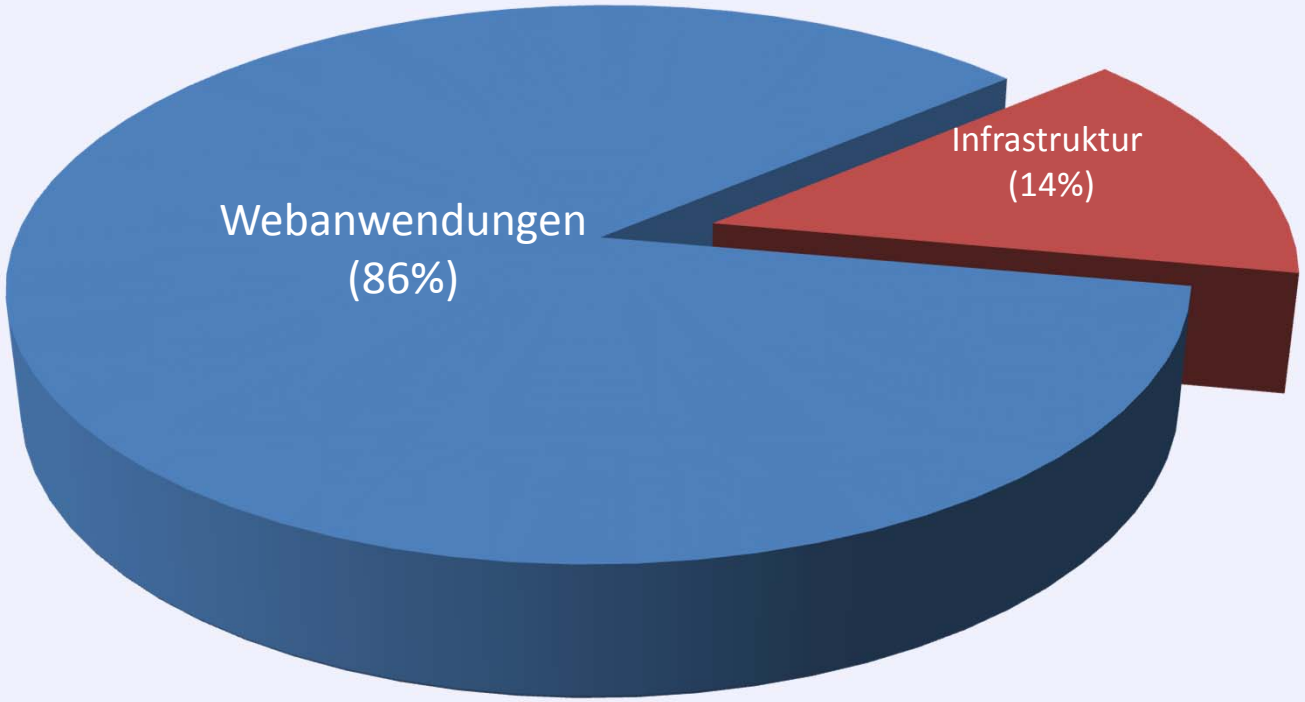
Obama, 29. Mai 2009

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

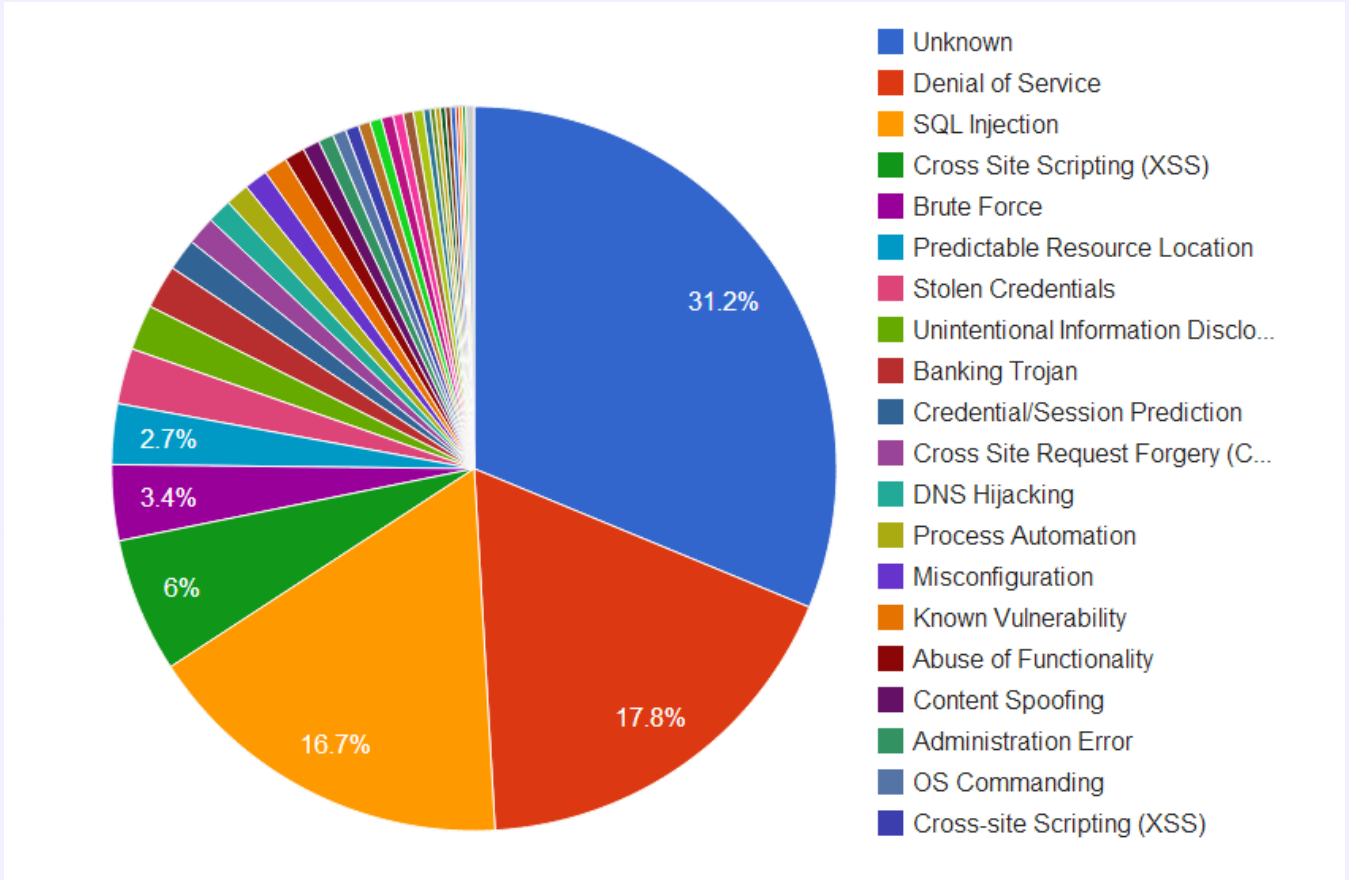
OWASP
The Open Web Application Security Project

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

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

Angriffe

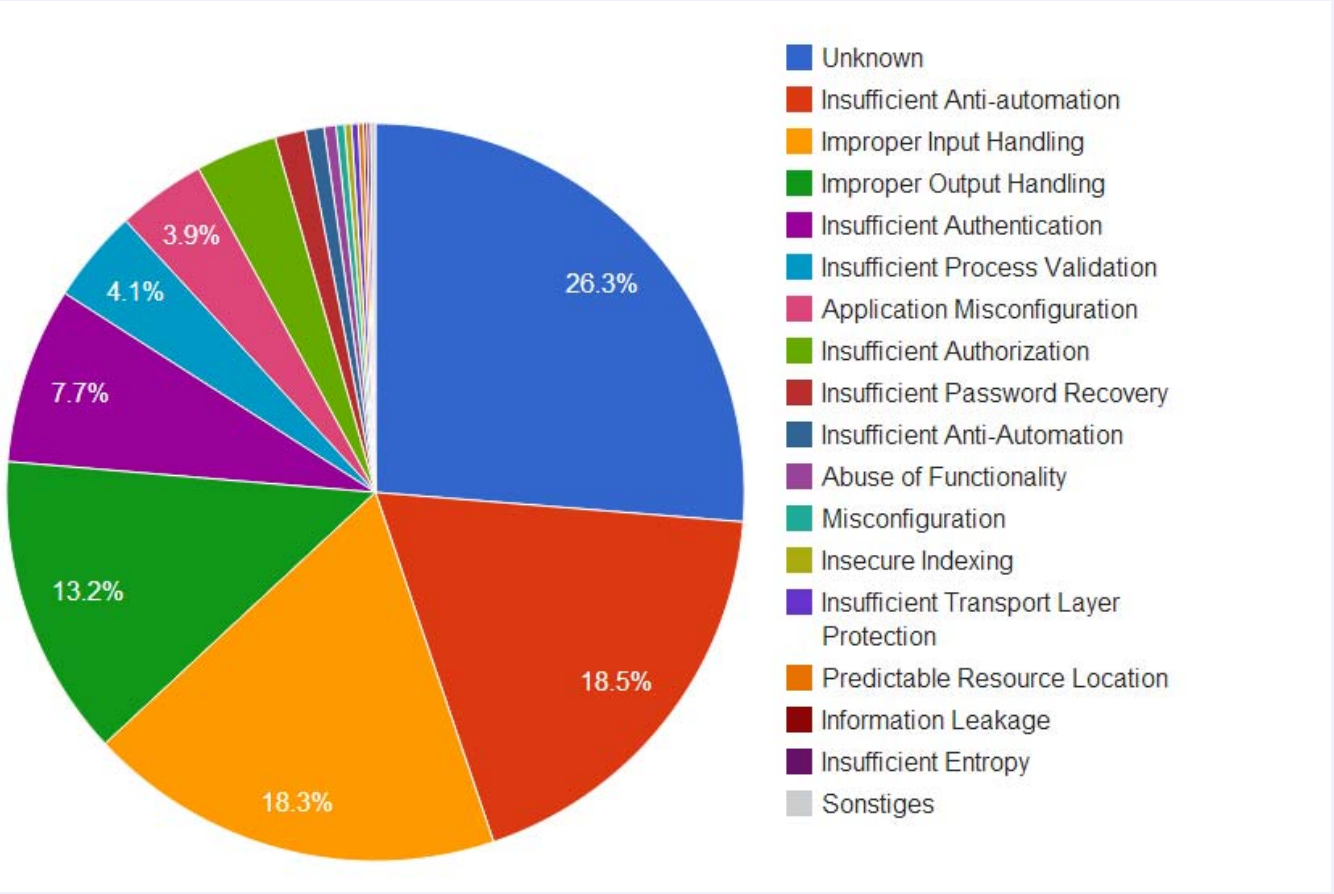
OWASP
The Open Web Application Security Project



1999-2011
http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

Schwachstellen

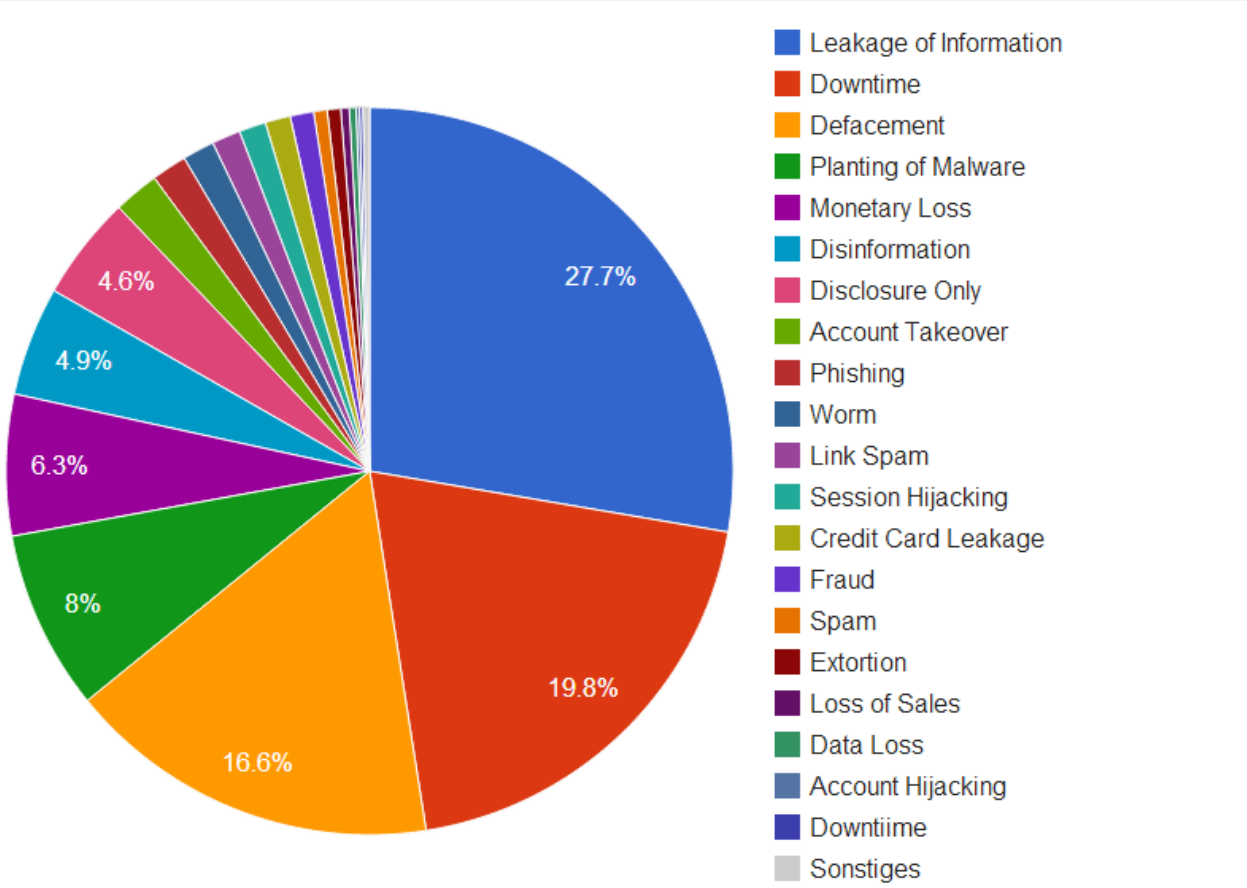
OWASP
The Open Web Application Security Project



1999-2011
http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

Schäden durch Angriffe auf Webanwendungen

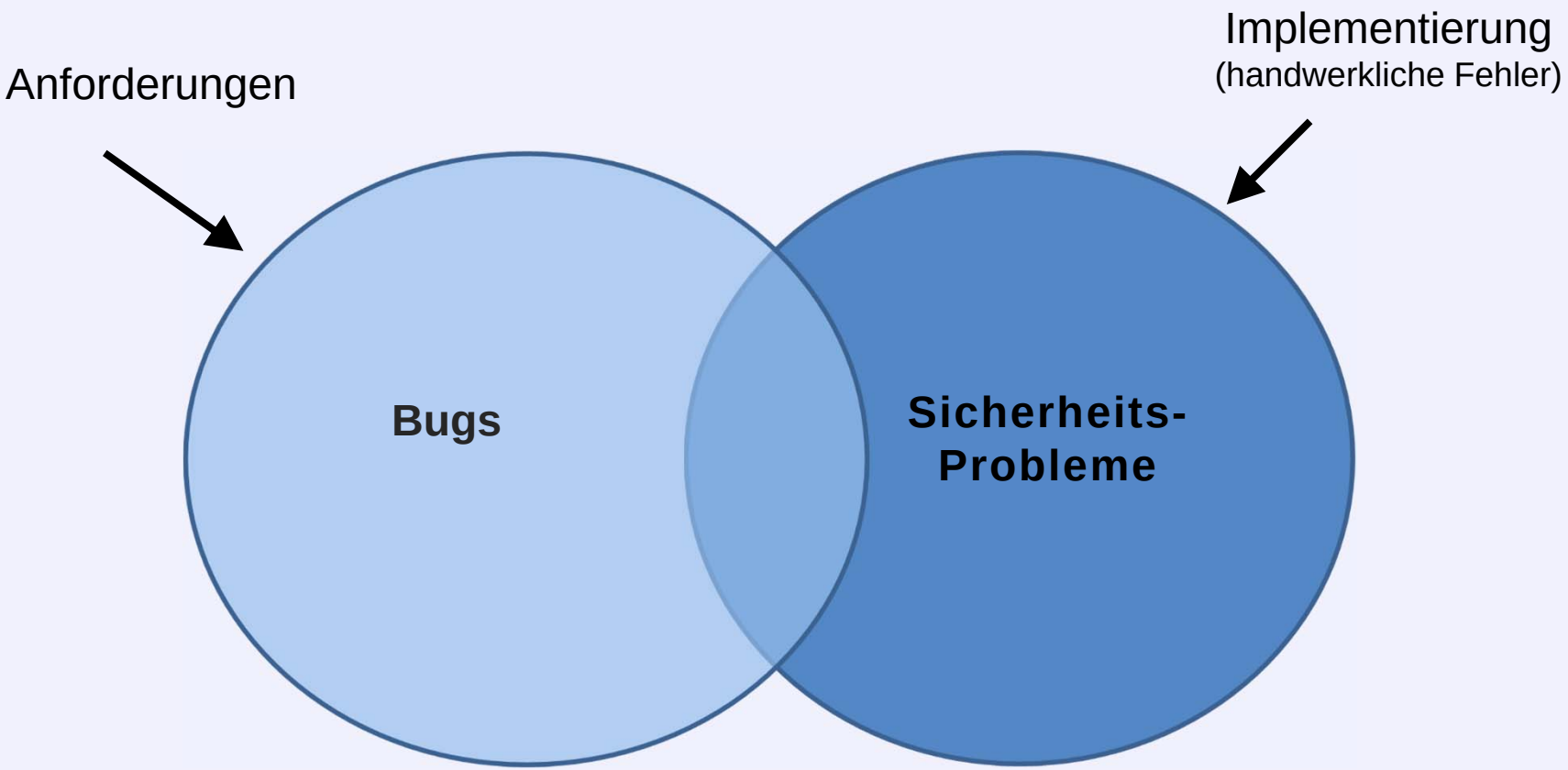
OWASP
The Open Web Application Security Project



1999-2011
http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

Bugs vs. Sicherheitsprobleme

OWASP
The Open Web Application Security Project



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 ...

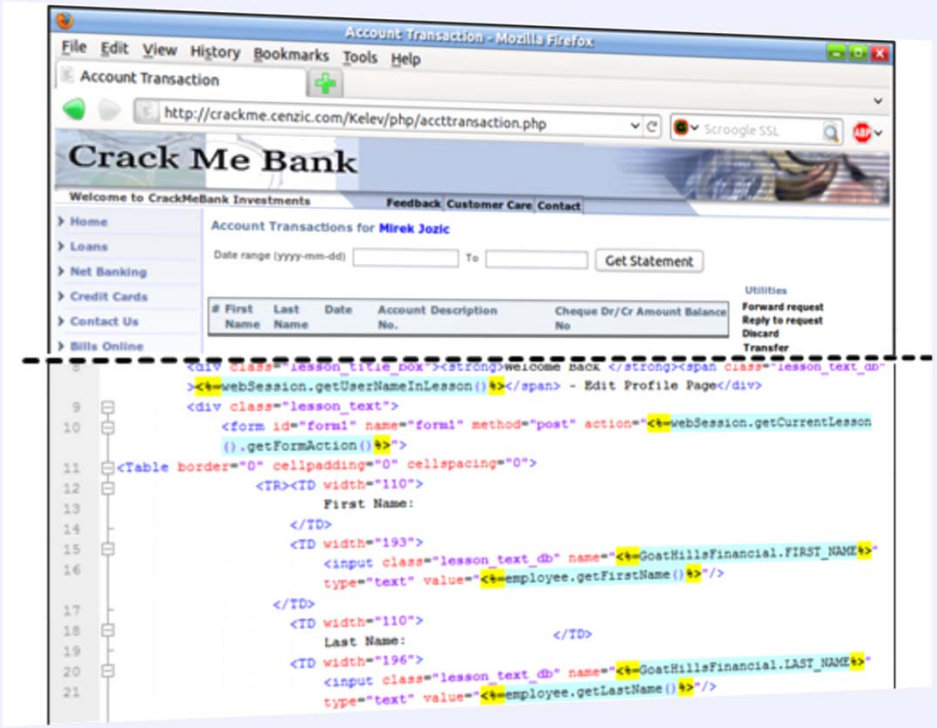
Code vs. Application Layer

OWASP
The Open Web Application Security Project

Schwachstellen in der laufenden Anwendung

Manuell: Pentest

Autom: Dynamic App. Security Testing (DAST)



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

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

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

Schwachstellen auf Codeebene

Manuell: Code Review

Autom: Static App. Security Testing (SAST)

Application Security Testing Tools

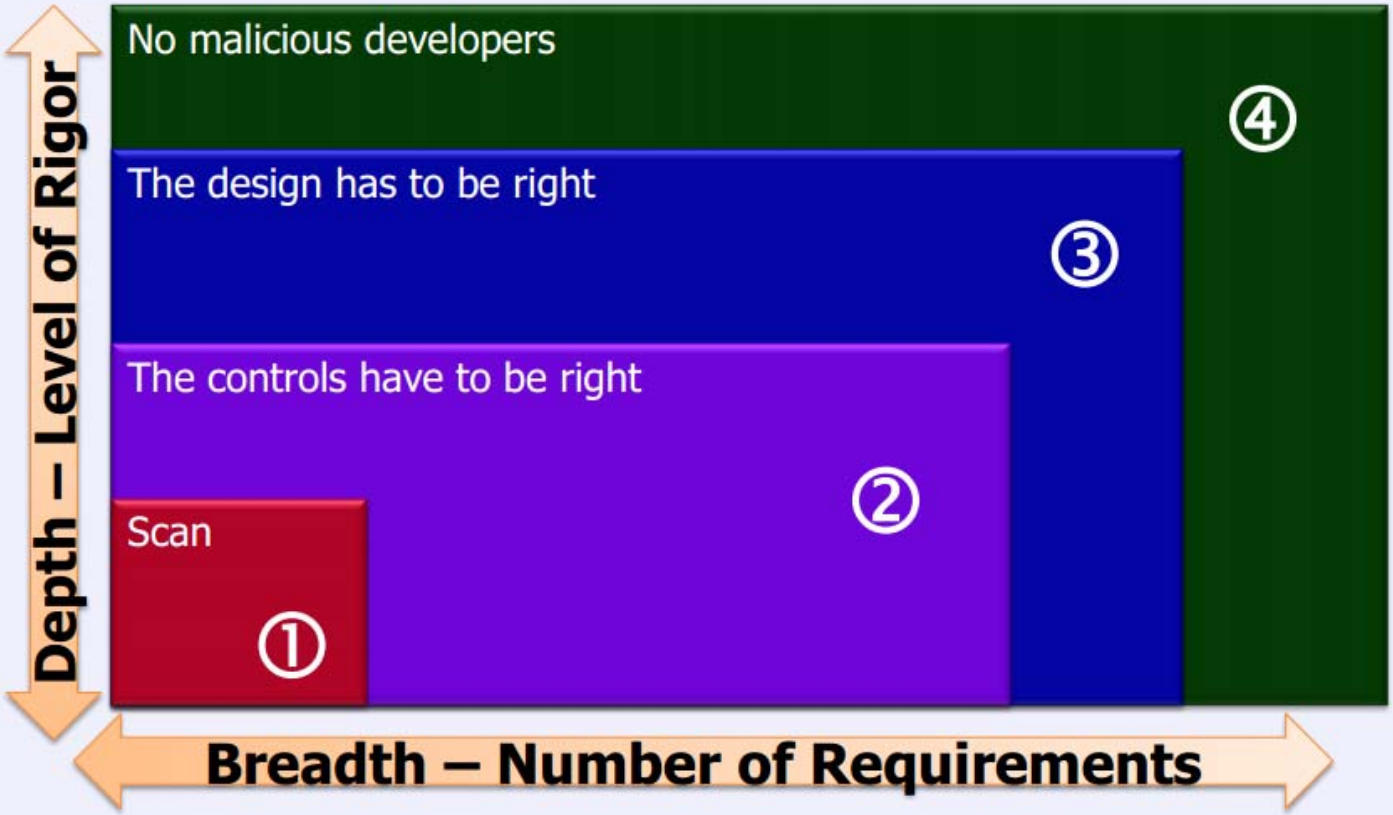
OWASP
The Open Web Application Security Project

- 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 …

Warum Tools? (1)

OWASP
The Open Web Application Security Project

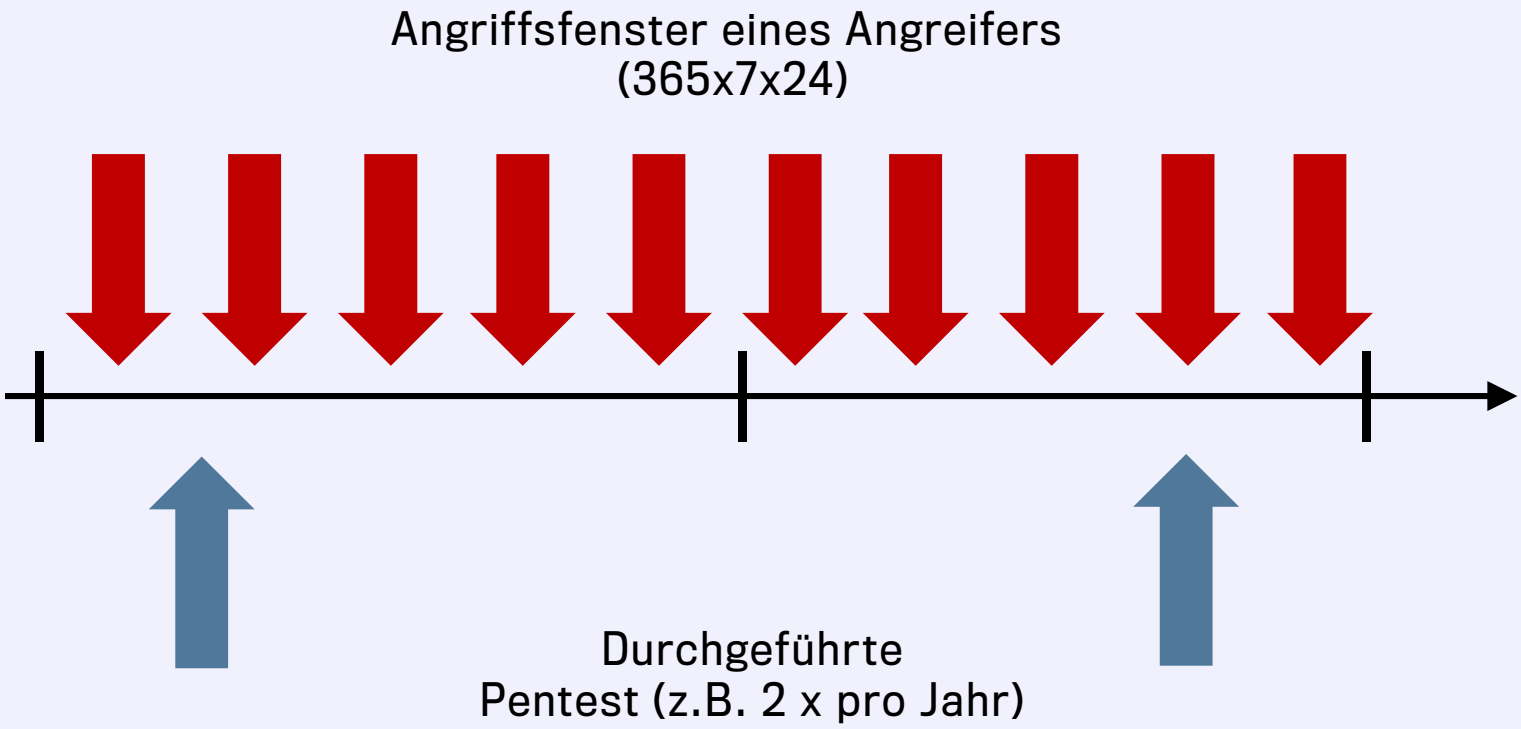
1. Anwendungen werden immer größer
   - Große Webanwendungen > 100 KLOC
   - Ein Security Reviewer schafft im Schnitt 1 KLOC pro Tag
2. Anwendungen ändern sich laufend
   - Agile Entwicklung
   - Schnelle Änderungen durch Änderungen im Deployment
   - Laut Whitehat, wurden in 2012 im Schnitt 56 kritische 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

Warum Tools? (2)

OWASP
The Open Web Application Security Project

4. Tools arbeiten schnell und können kontinuierlich ausgeführt werden

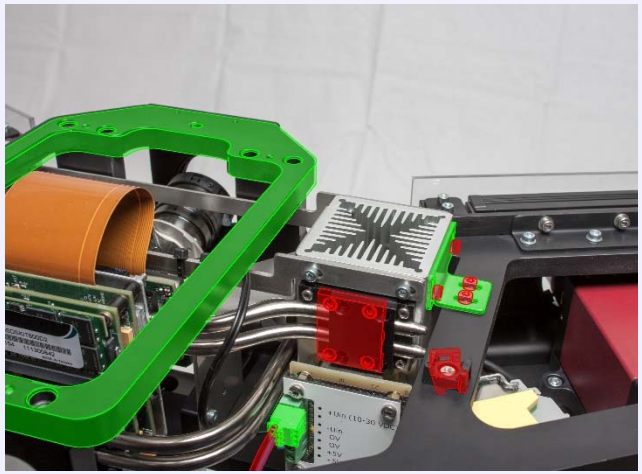


Quelle: OWASP

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 Policys / Vorgaben laufend prüfen.
7. Tools ermöglichen nachvollziehbare, objektive und standardisierte Analysen.

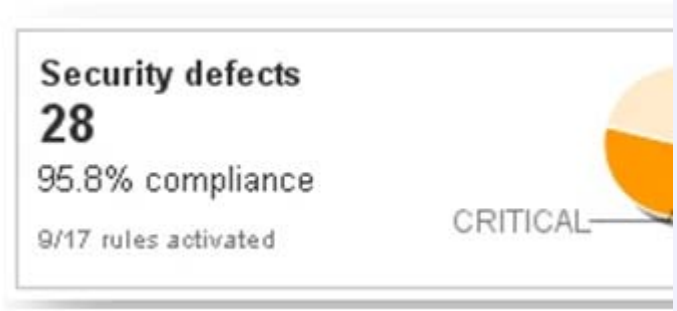


www.iff.fraunhofer.de

Warum Tools? (3)

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

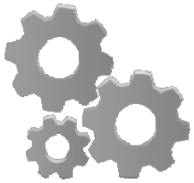
Sicherheitsprobleme, die DAST-Tools finden können…

OWASP
The Open Web Application Security Project

Implementierungs-Fehler

SQL Injection*
XSS*
Known Vuln.
etc.

Konfigurationsfehler



Error Handling*
TLS-Config
etc.

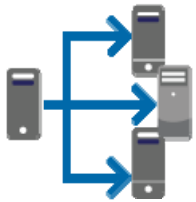
Ungültige TLS-Zerts

Abgelaufene oder invalide TLS/SSL-Zertifikate

Sicherheitsanforderungen

Loginversuche
Passwortlänge
Access Controls
etc.

Deploymentfehler



Testzugänge
SVN-Dateien
etc.

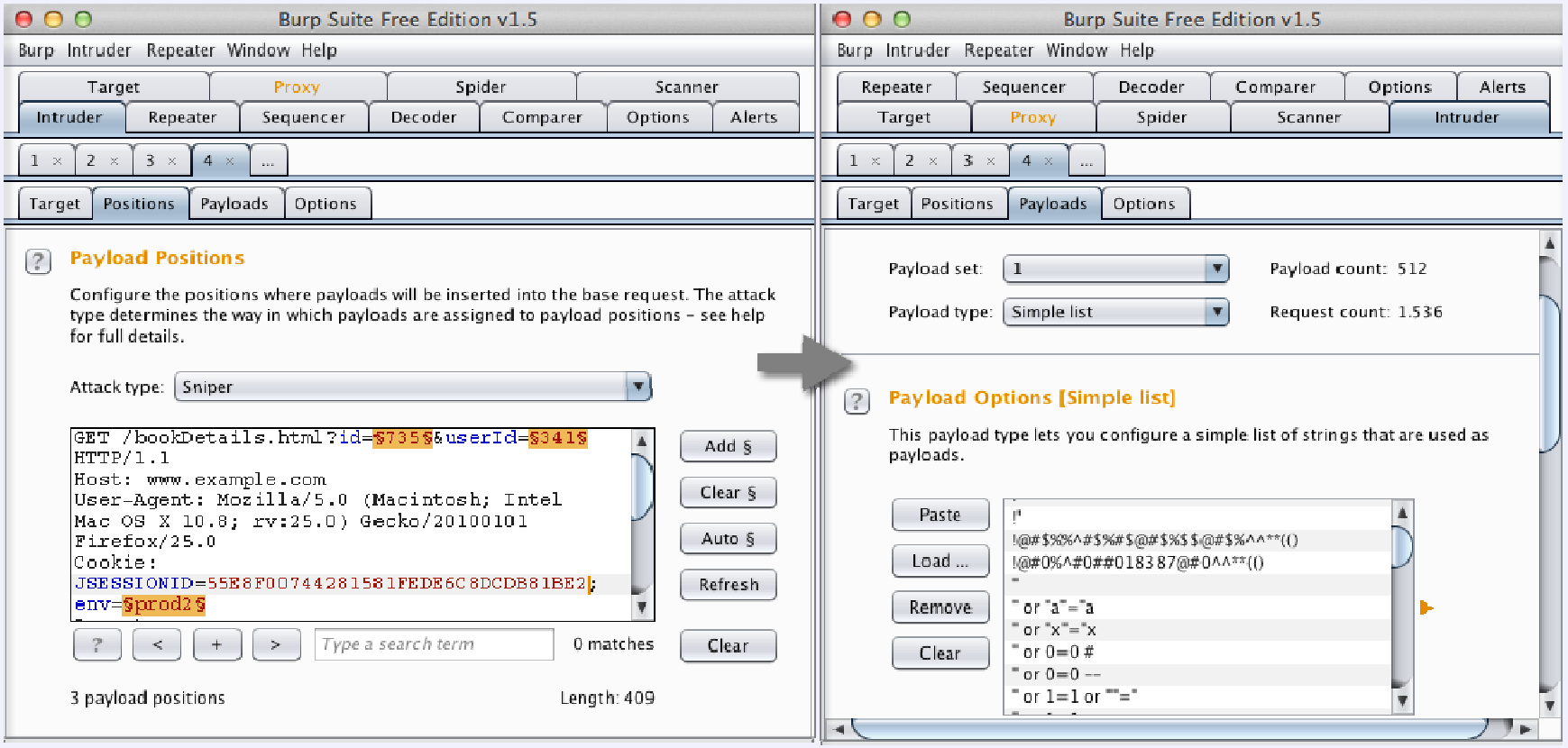
Malware

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

* Low Hanging Fruits

Pentesting Tools - Burp

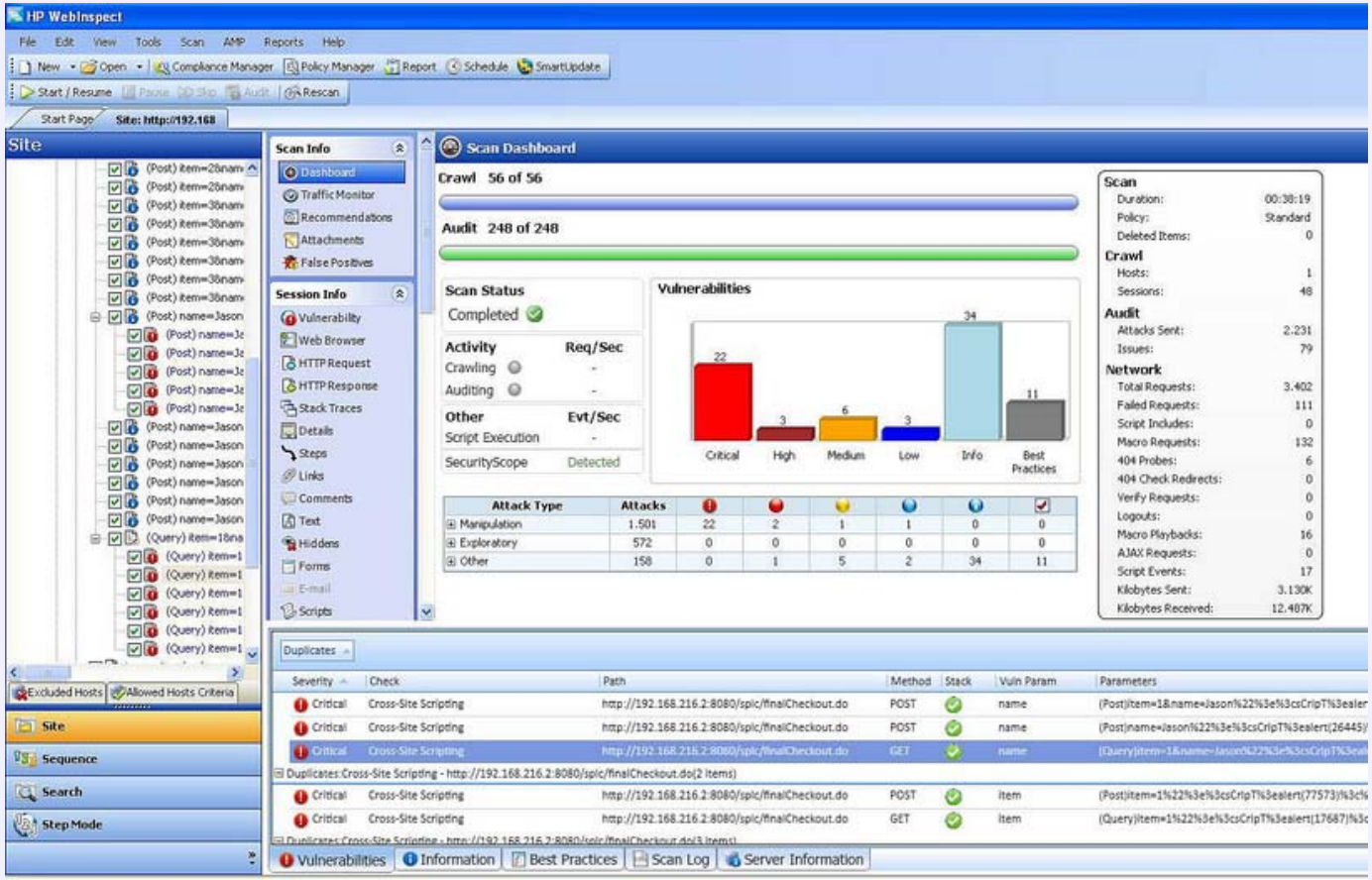
OWASP
The Open Web Application Security Project



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

Web Security Scanner GUIs (kommerziell)

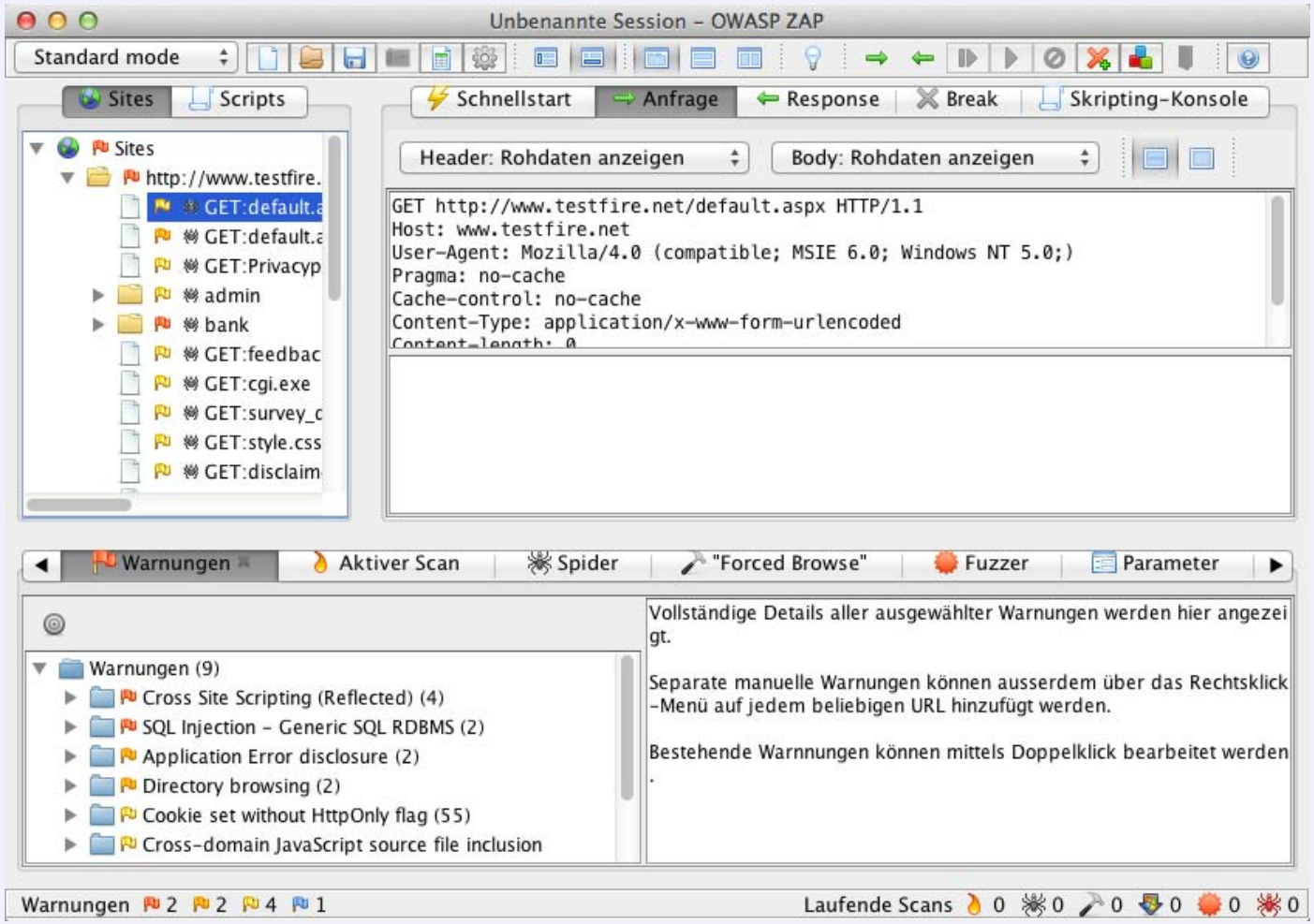
OWASP
The Open Web Application Security Project



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

Web Security Scanner (kostenfrei)

OWASP
The Open Web Application Security Project



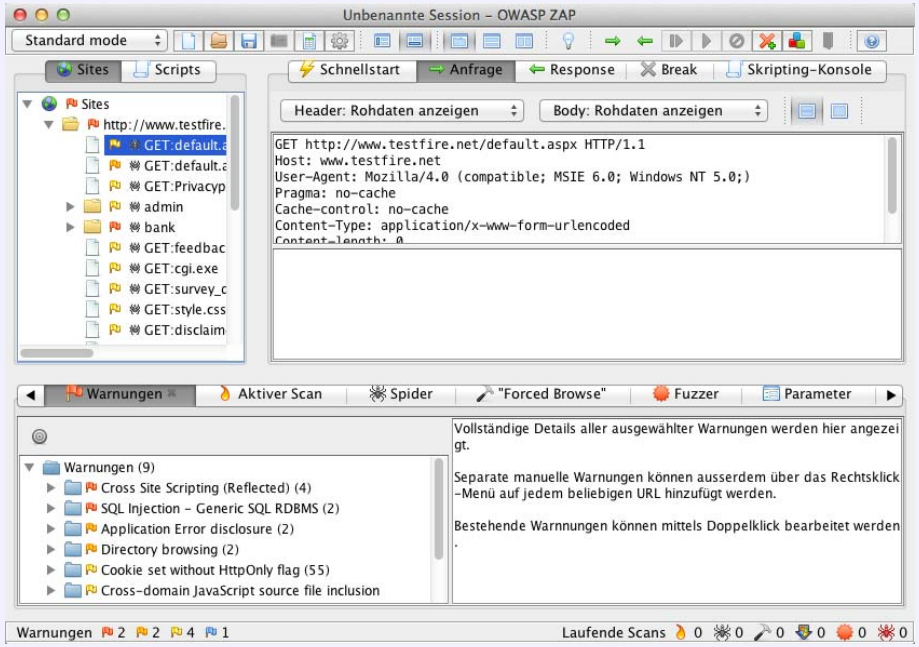
- OWASP ZAP
- w3af
- skipfish
- Nikto / Wikto
- SSLScan / SSL Labs

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

OWASP ZAP

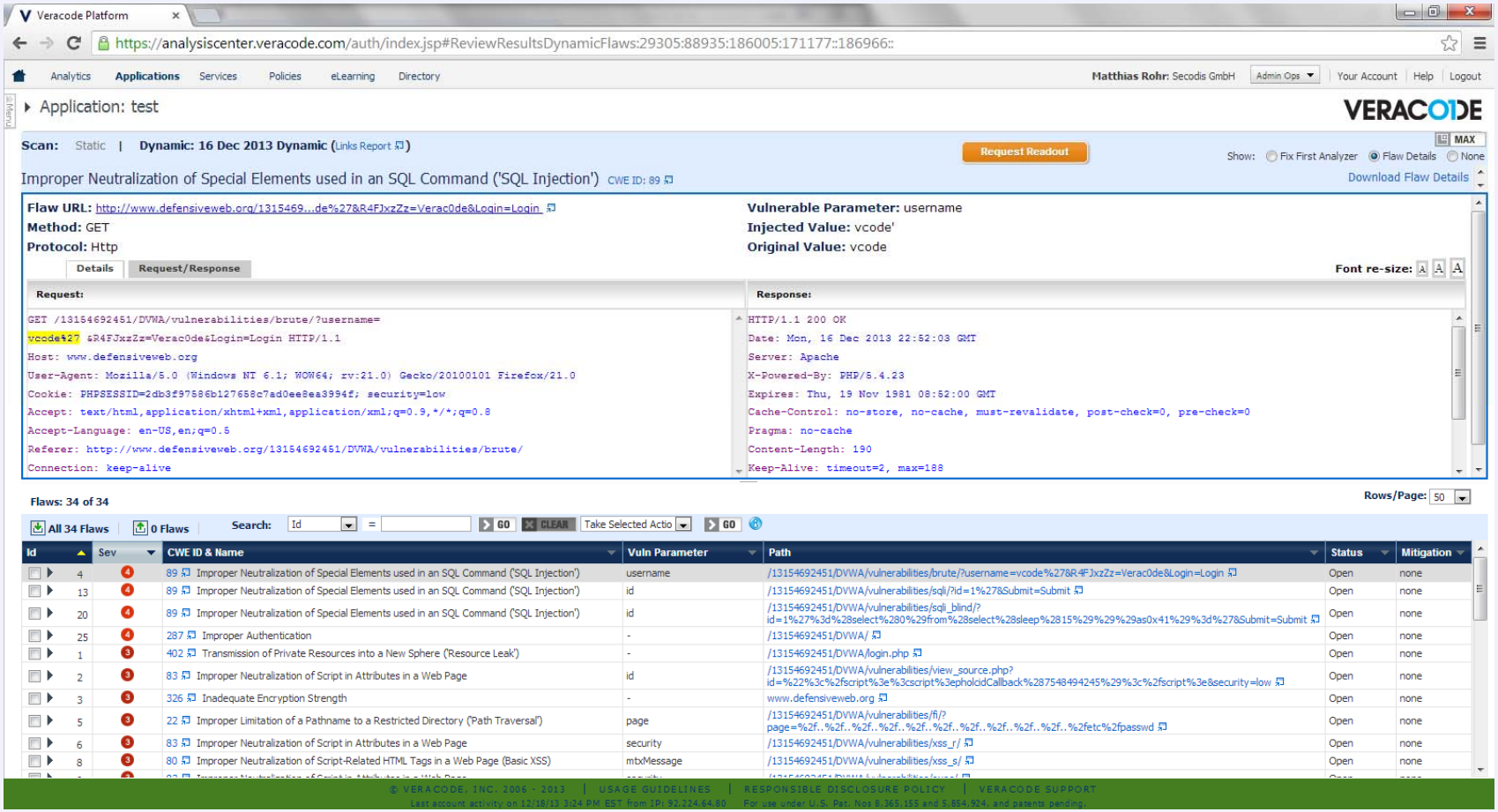
OWASP
The Open Web Application Security Project

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



Vorsicht vor dem Einsatz auf produktiven Systemen!

DAST as a Service: Veracode

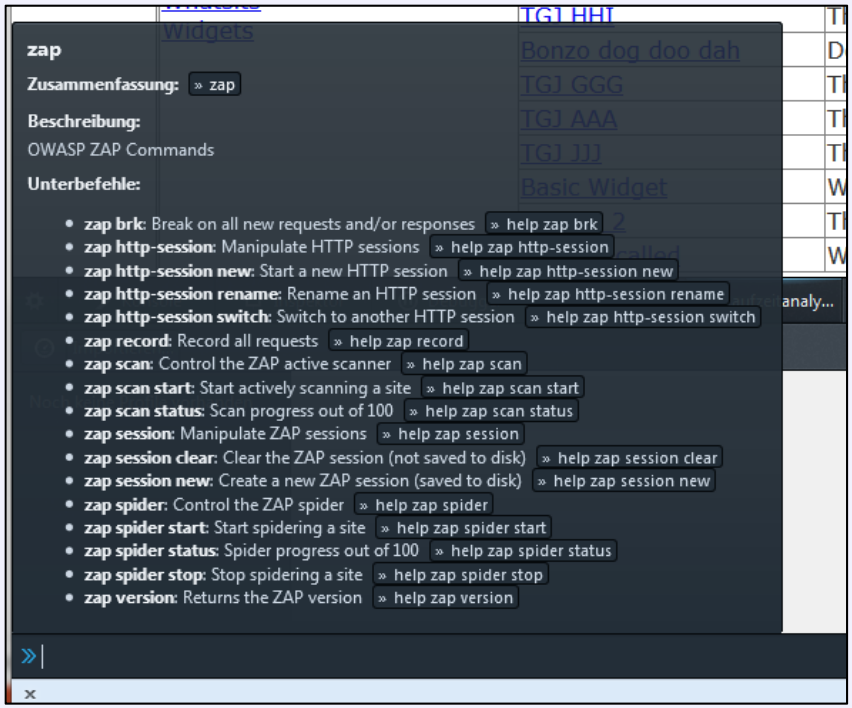


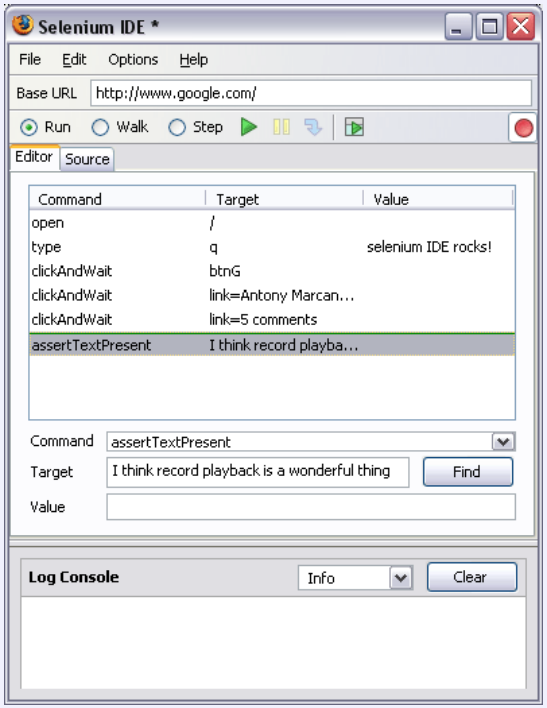
Weiterhin: Whitehat Sentinal, NTOSpider On-Demand

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)





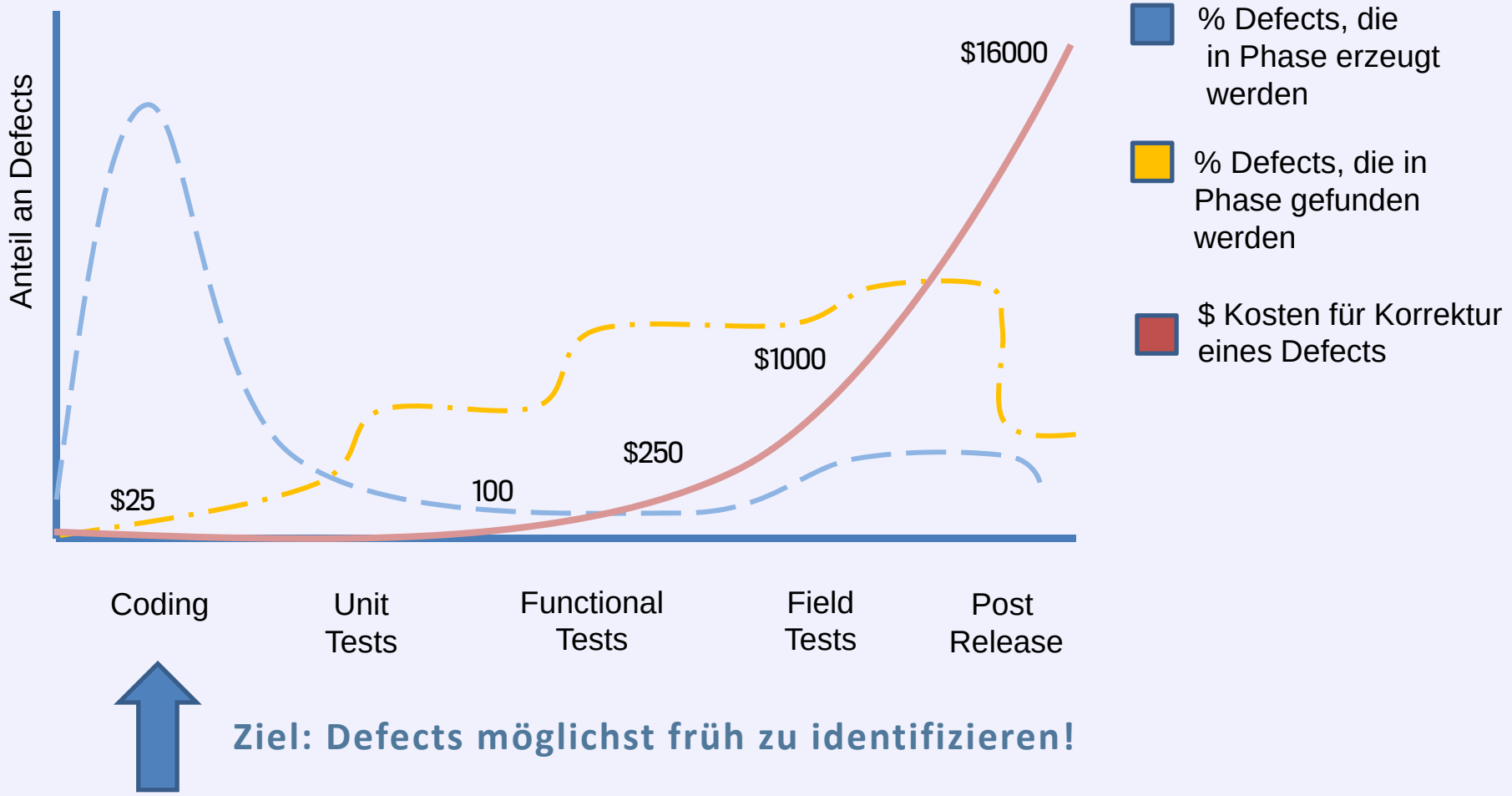
OWASP
The Open Web Application Security Project

Static Application Security Testing (SAST)

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

Sicherheit im SDLC

OWASP
The Open Web Application Security Project



Caspers Jones, Applied Software Measurement: Global Analysis of Productivity and Quality, 1996

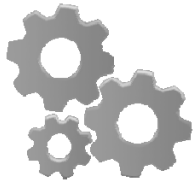
Sicherheitsprobleme, die SAST-Tools finden können…

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-Contraints
- Etc.

Sensible Änderungen

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

SAST-Analyseverfahren

OWASP
The Open Web Application Security Project

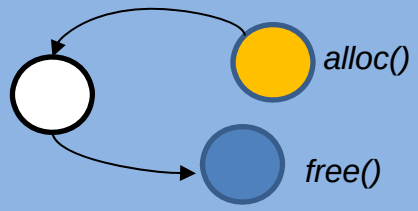
Lexikalische Analyse

```
grep -i -r “exec(” *
```

Identifizierung verdächtiger Zeichenketten im Code (Reguläre Ausdrücke)

Beispiele
- Hartkod. Passwörter
- Entwicklerkommentare
- Kommentierter Code

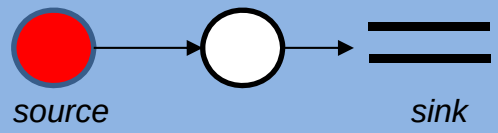
Kontrollflussanalyse



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

Beispiele
- Buffer Overflows
- Race Conditions

Datenflussanalyse



Dient dem Auffinden von Fehlern in der Datenvalidierung

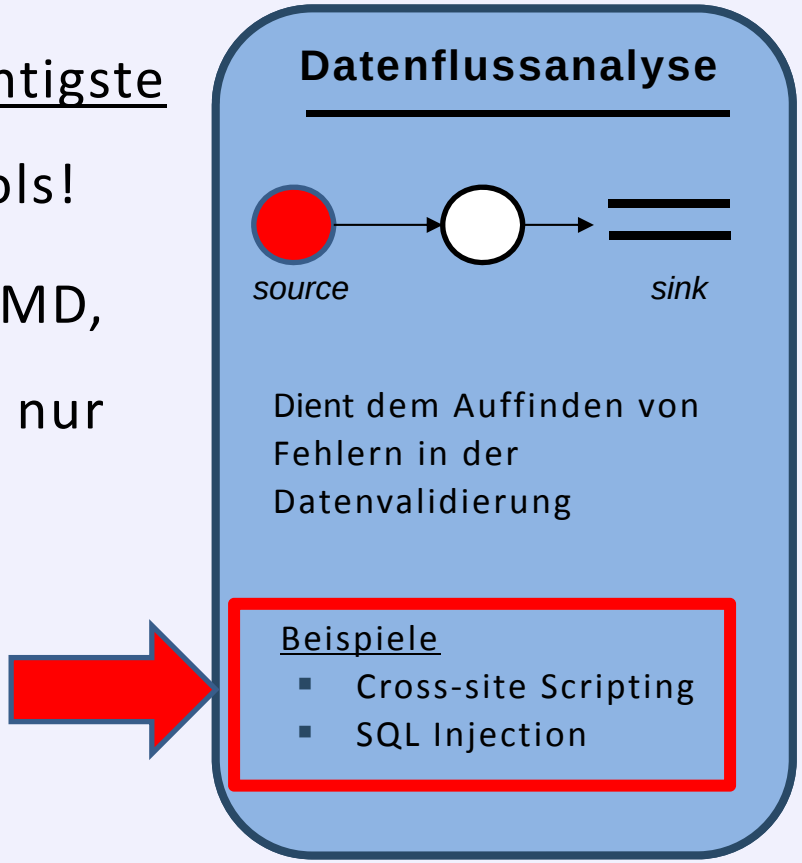
Beispiele
- Cross-site Scripting
- SQL Injection

Weiterhin: Semantische Analyse, Strukturanalyse, Konfigurationsanalysen

Datenflussanalysen

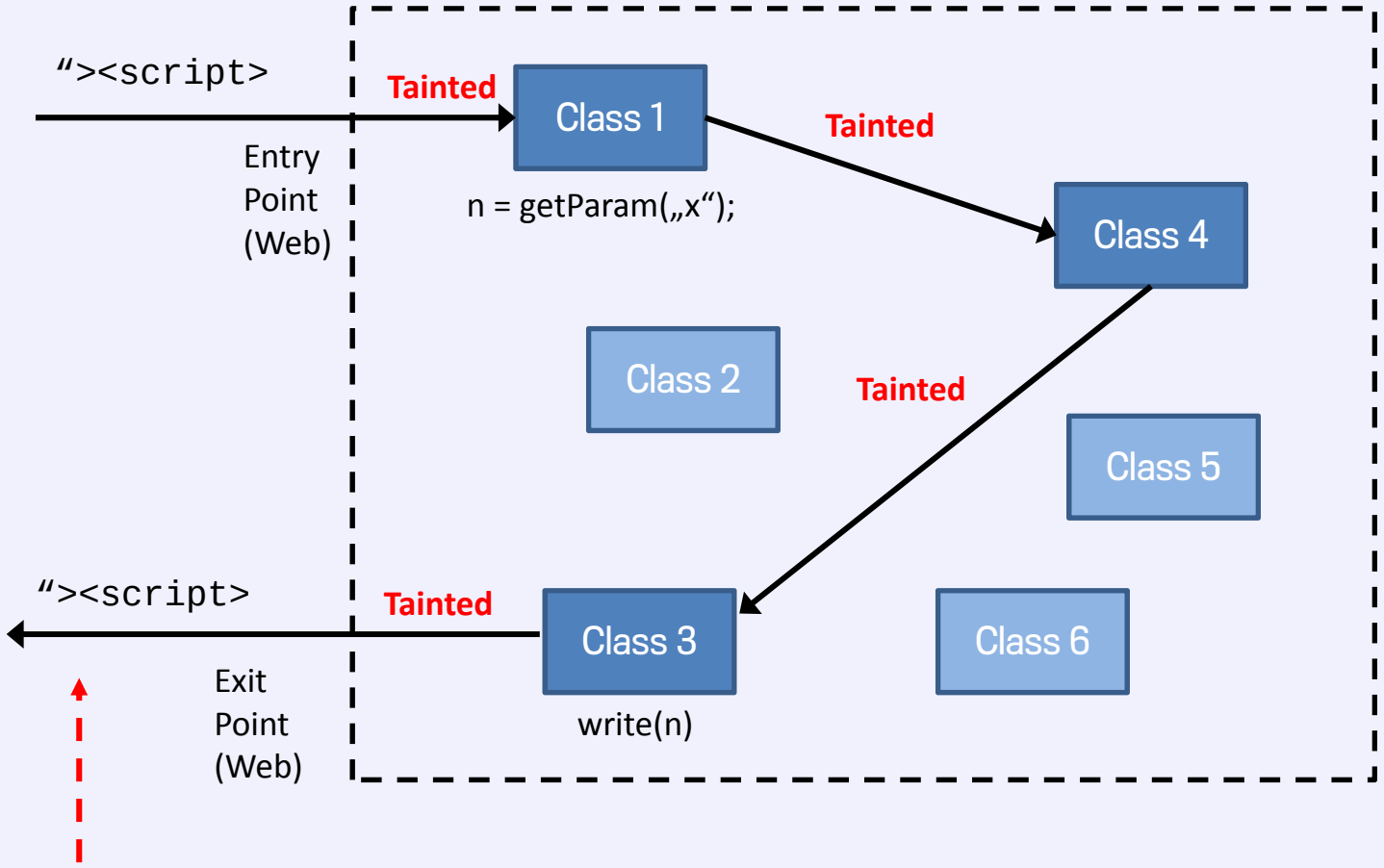
OWASP
The Open Web Application Security Project

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



Datenflussanalysen (aka Taint Analyse)

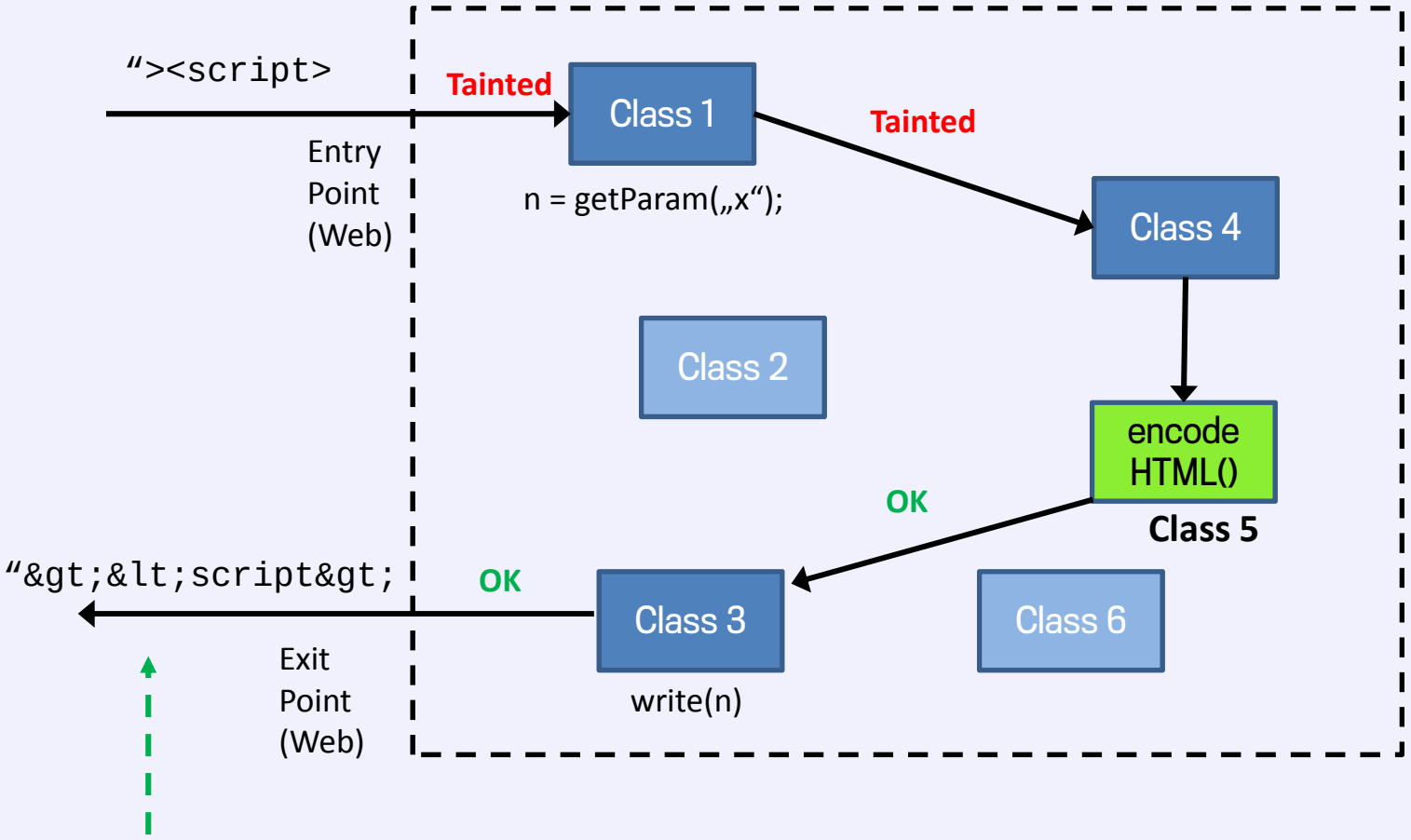
OWASP
The Open Web Application Security Project



Cross-site Scripting!

Datenflussanalysen (aka Taint Analyse)

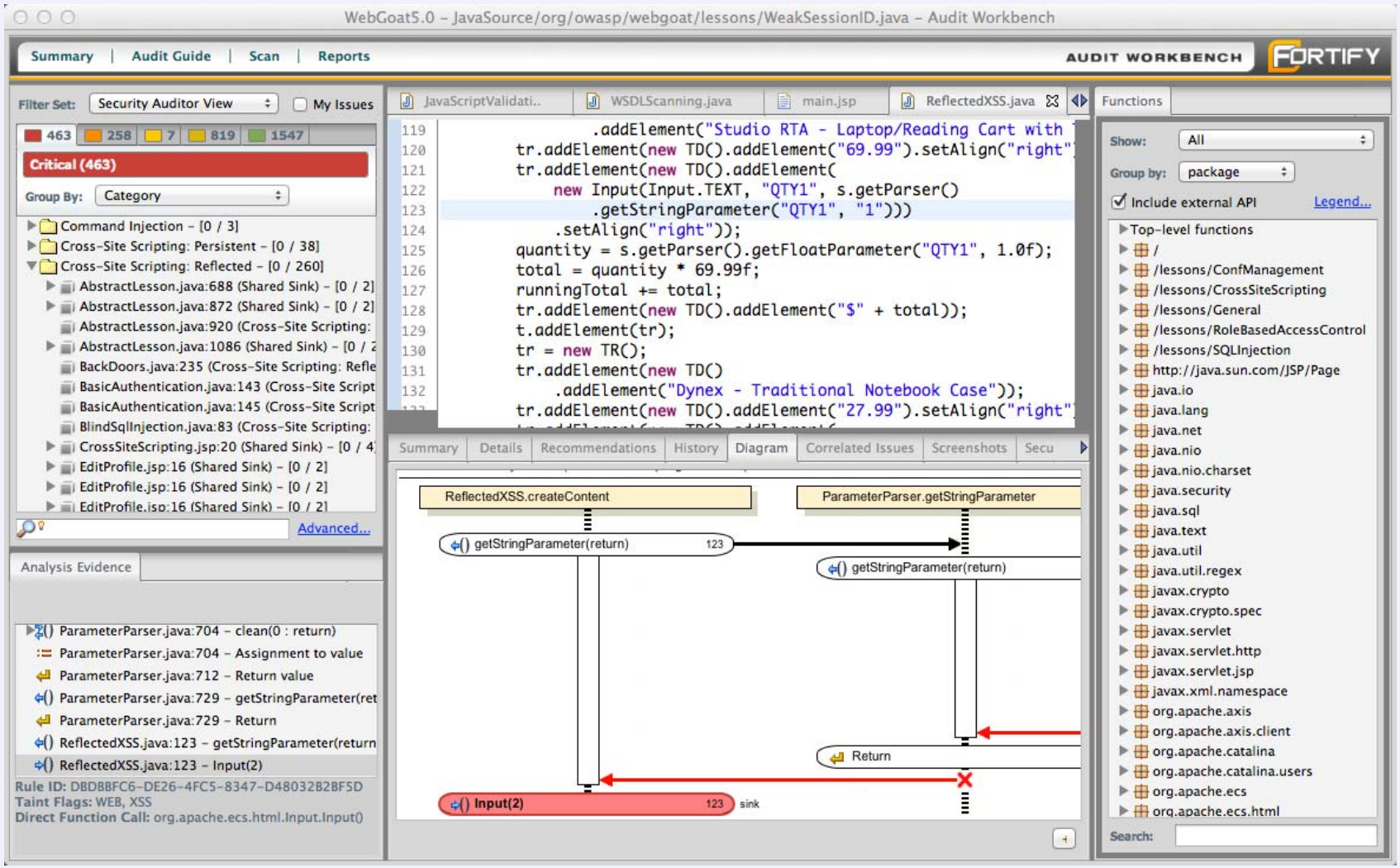
OWASP
The Open Web Application Security Project



kein Cross-site Scripting!

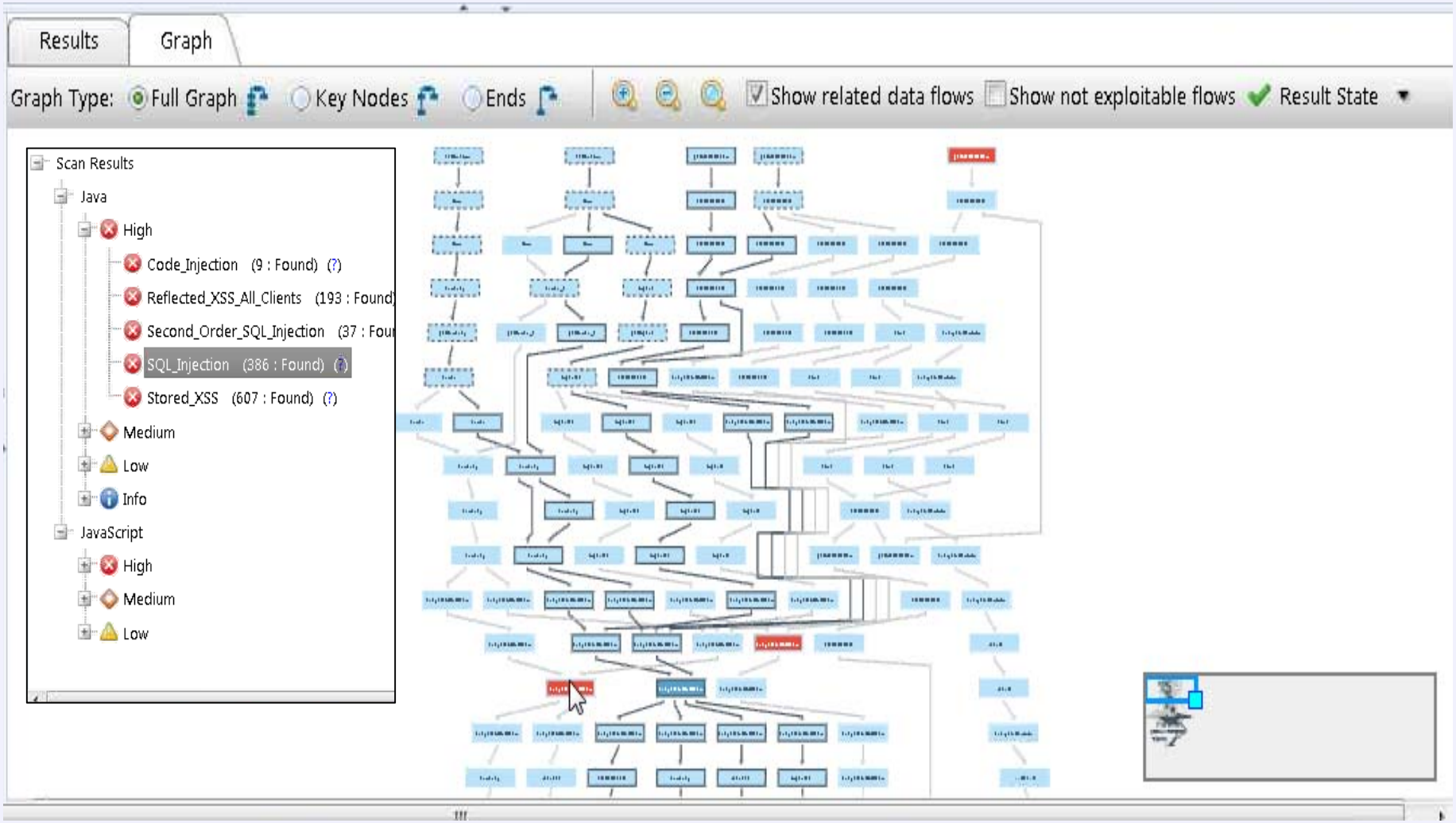
HP Fortify - Auditworkbench

OWASP
The Open Web Application Security Project



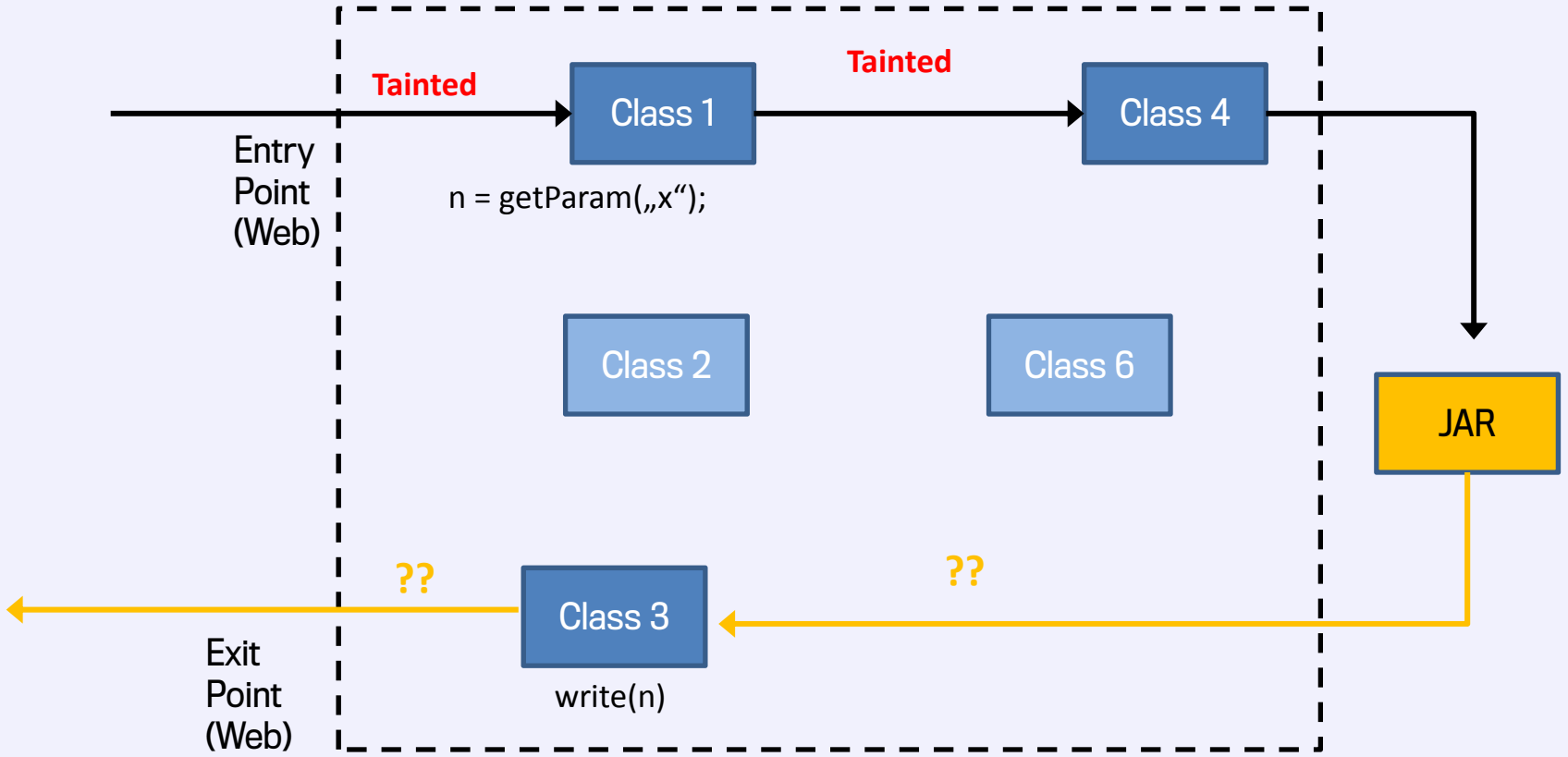
Checkmarx - Datenflussanalysen

OWASP
The Open Web Application Security Project



Problem von Source Code Analysen

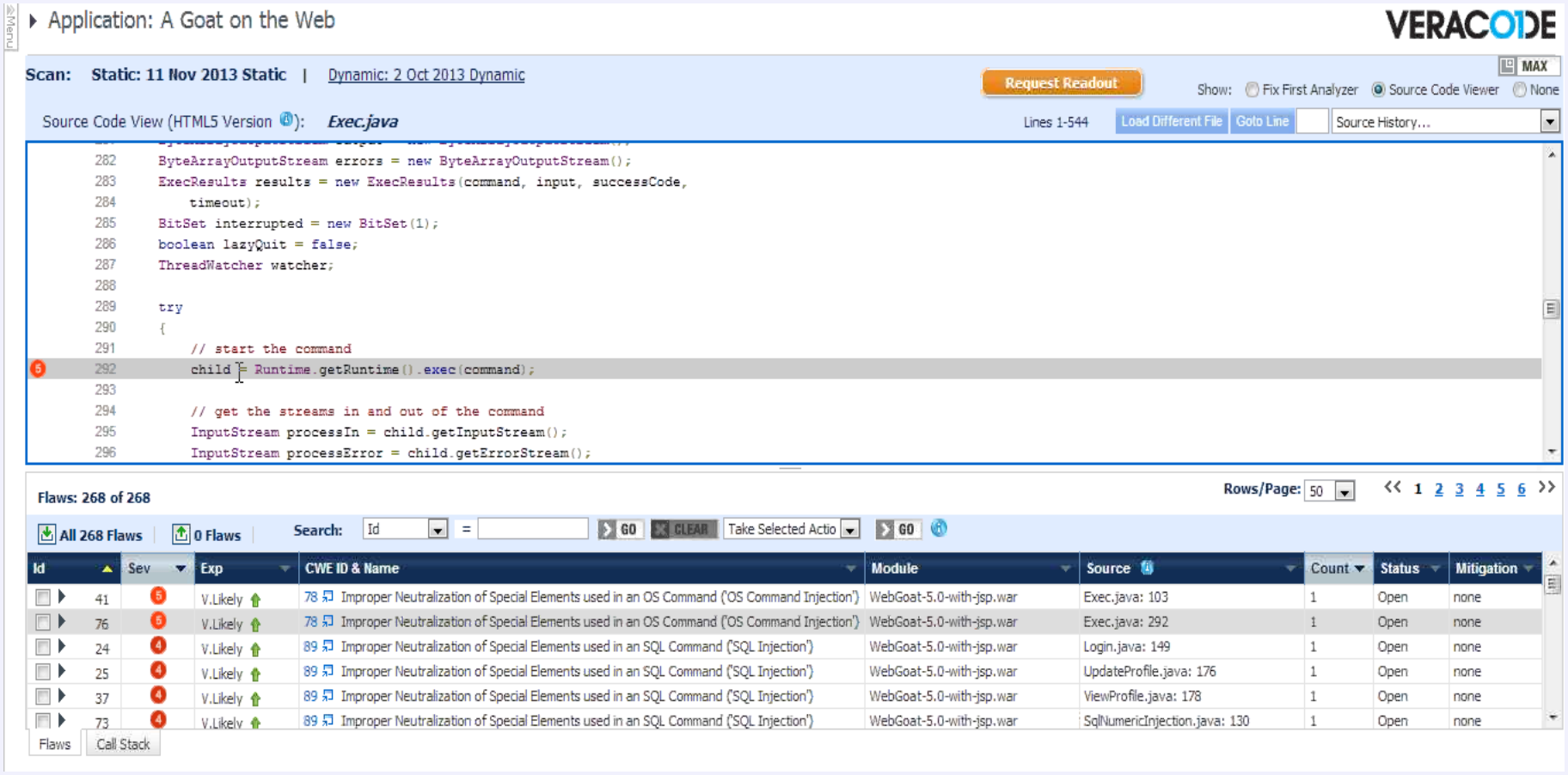
OWASP
The Open Web Application Security Project



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

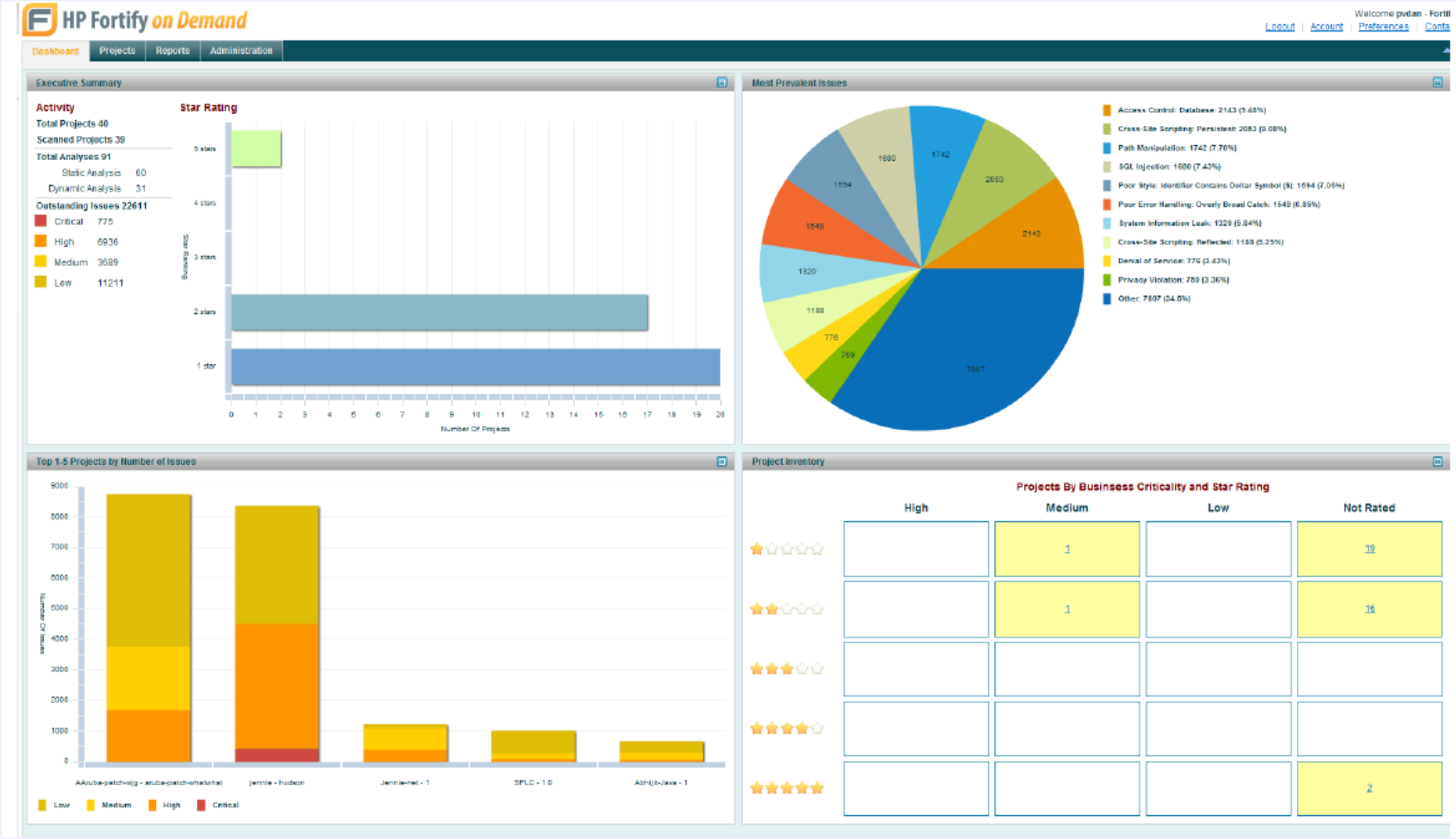
Bytecodeanalysen mit Veracode

OWASP
The Open Web Application Security Project



Fortify on Demand (FOD)

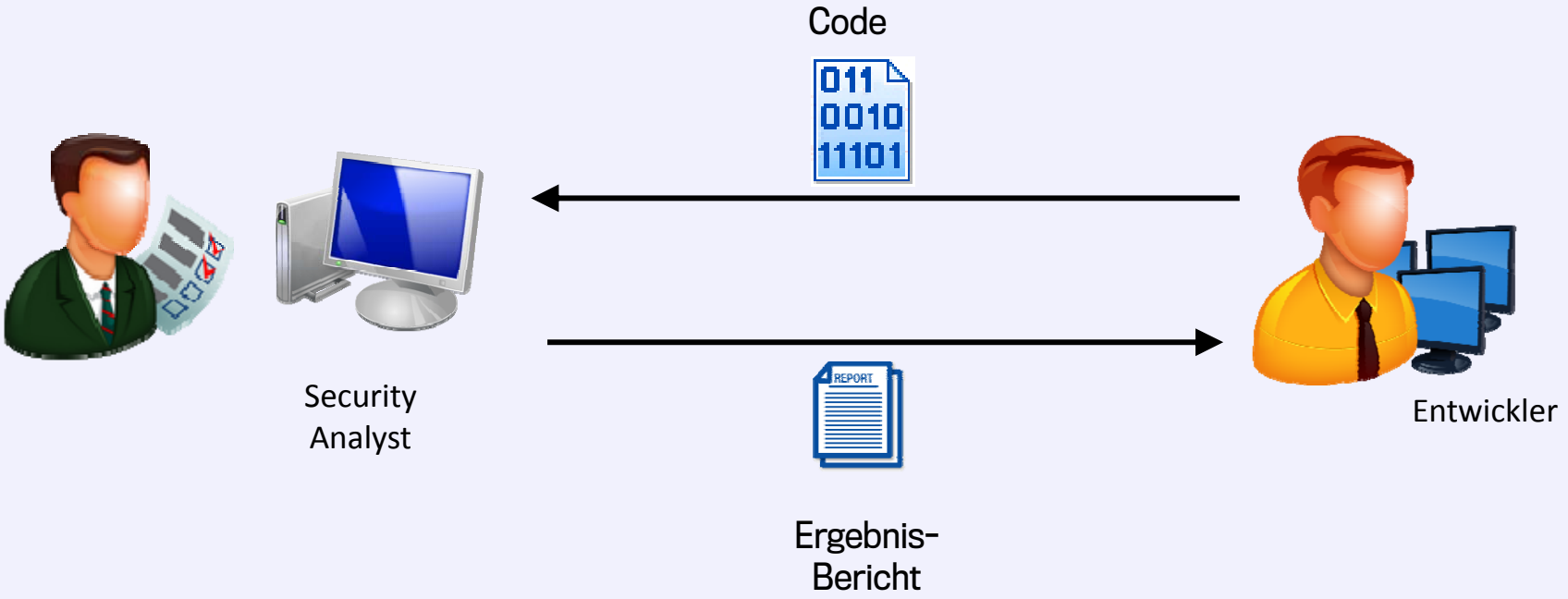
OWASP
The Open Web Application Security Project



www.fortifymyapp.com

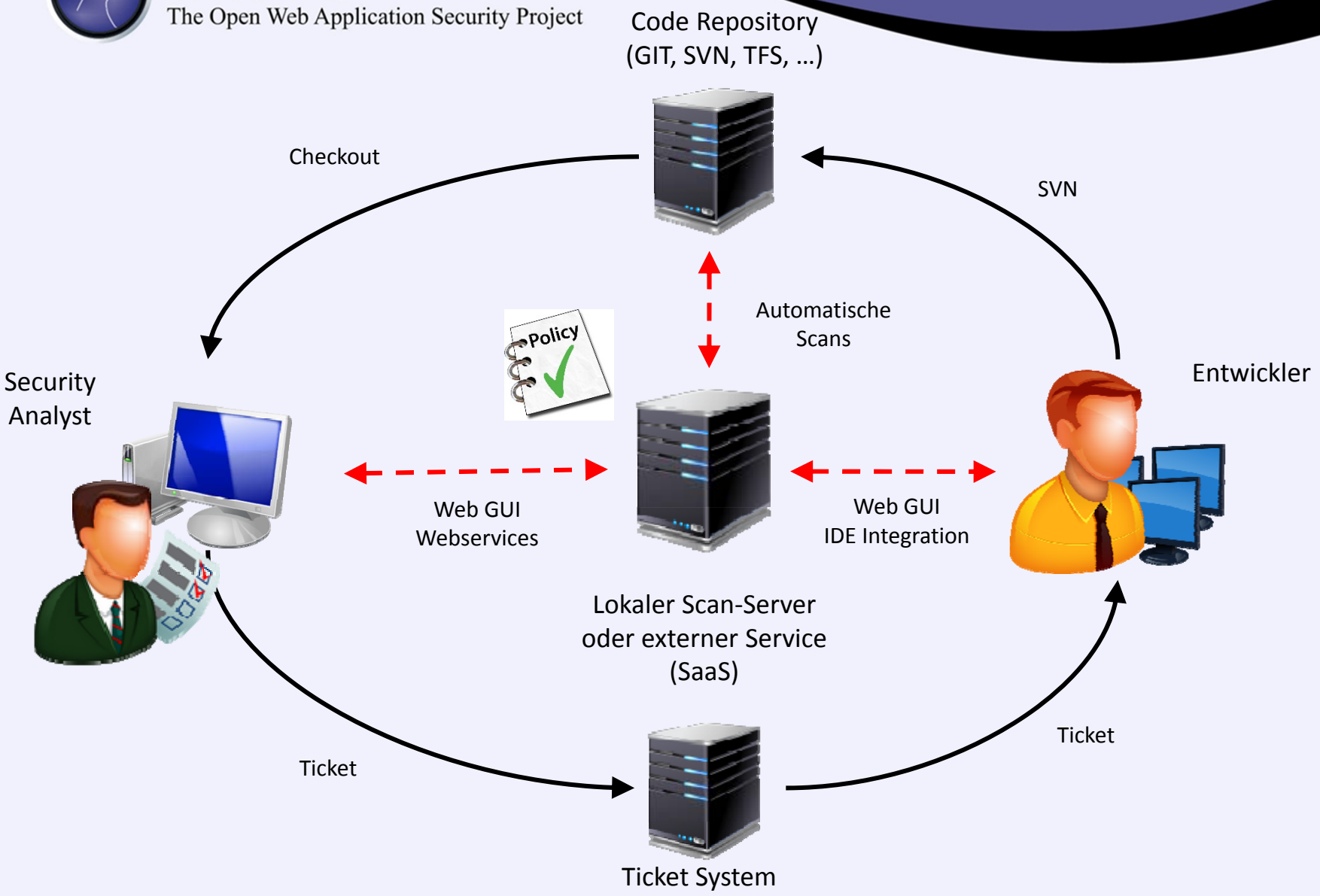
SAST-Deployment: Tools der 1. Generation

OWASP
The Open Web Application Security Project



SAST Deployment: Heutige Integration

OWASP
The Open Web Application Security Project

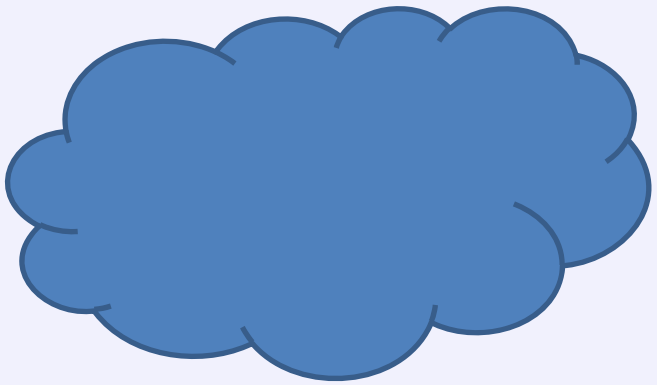


On-Premise vs. Off-Premise

OWASP
The Open Web Application Security Project

On-Premise
(Lokal intalliert)

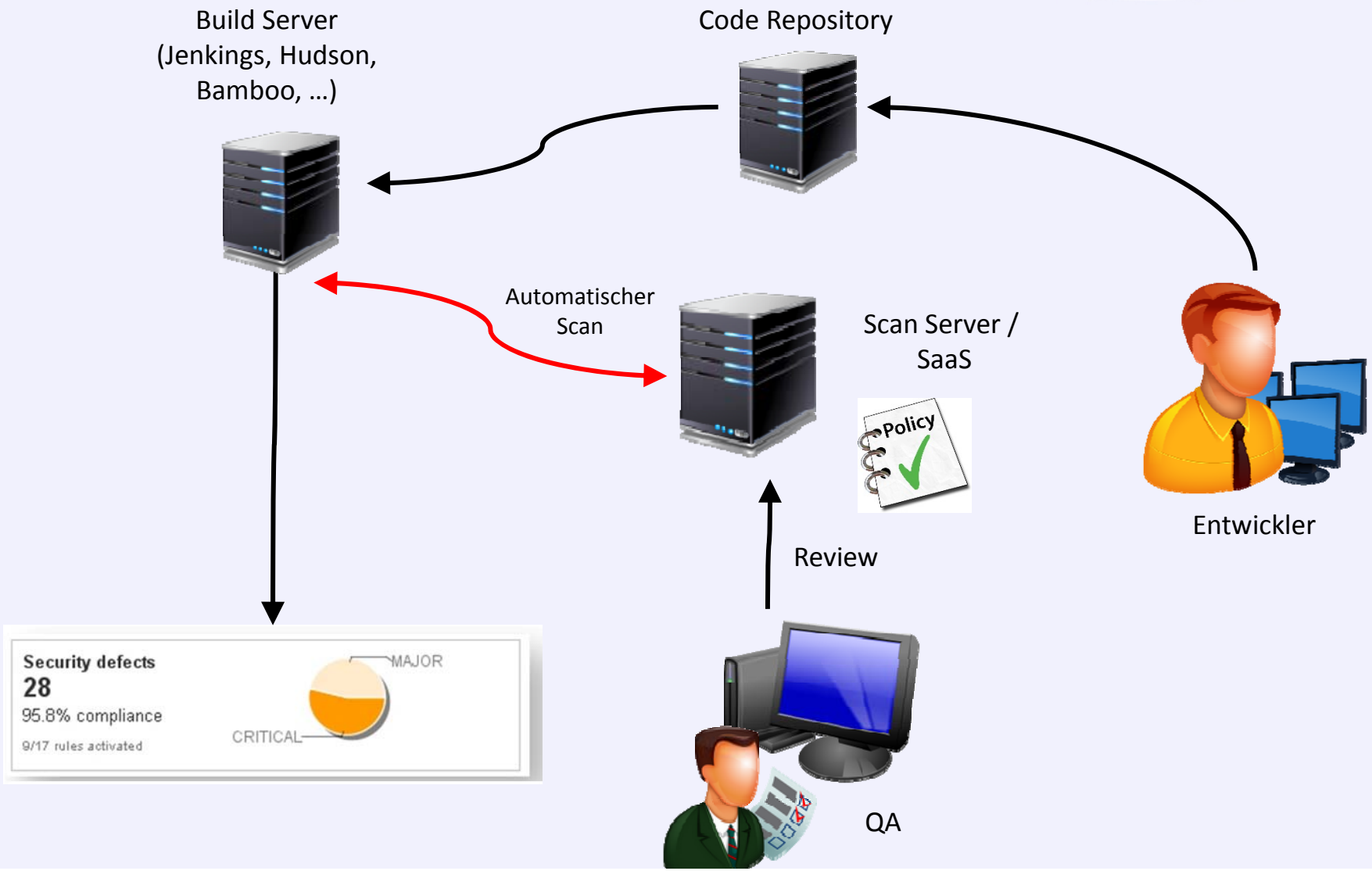
Vs.



Off-Premise
(Cloud-Based)

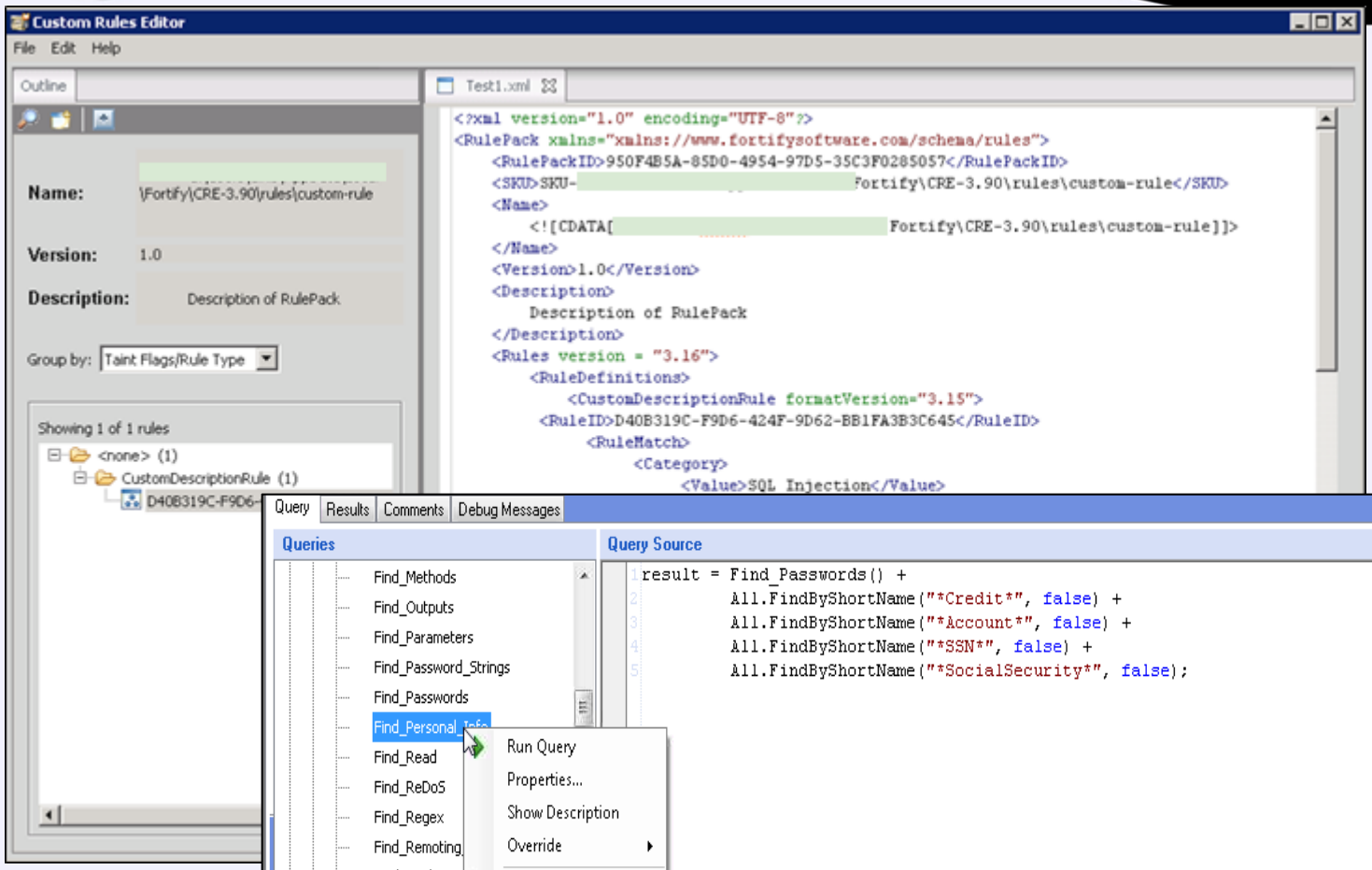
SAST Deployment: Heutige Integration (2)

OWASP
The Open Web Application Security Project



Custom Rules

OWASP
The Open Web Application Security Project

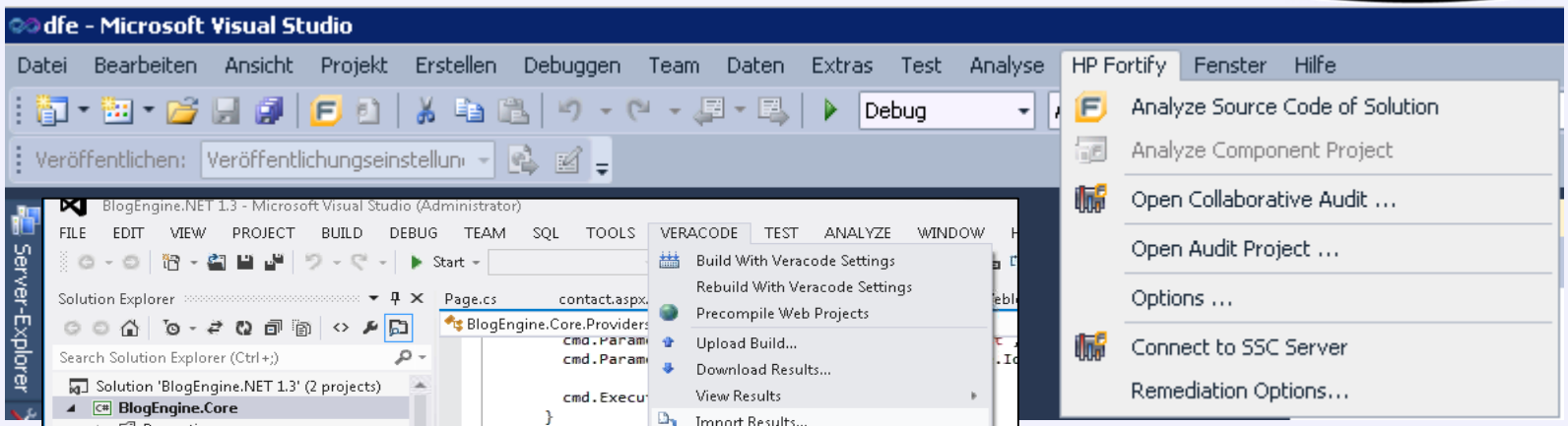


Fortify

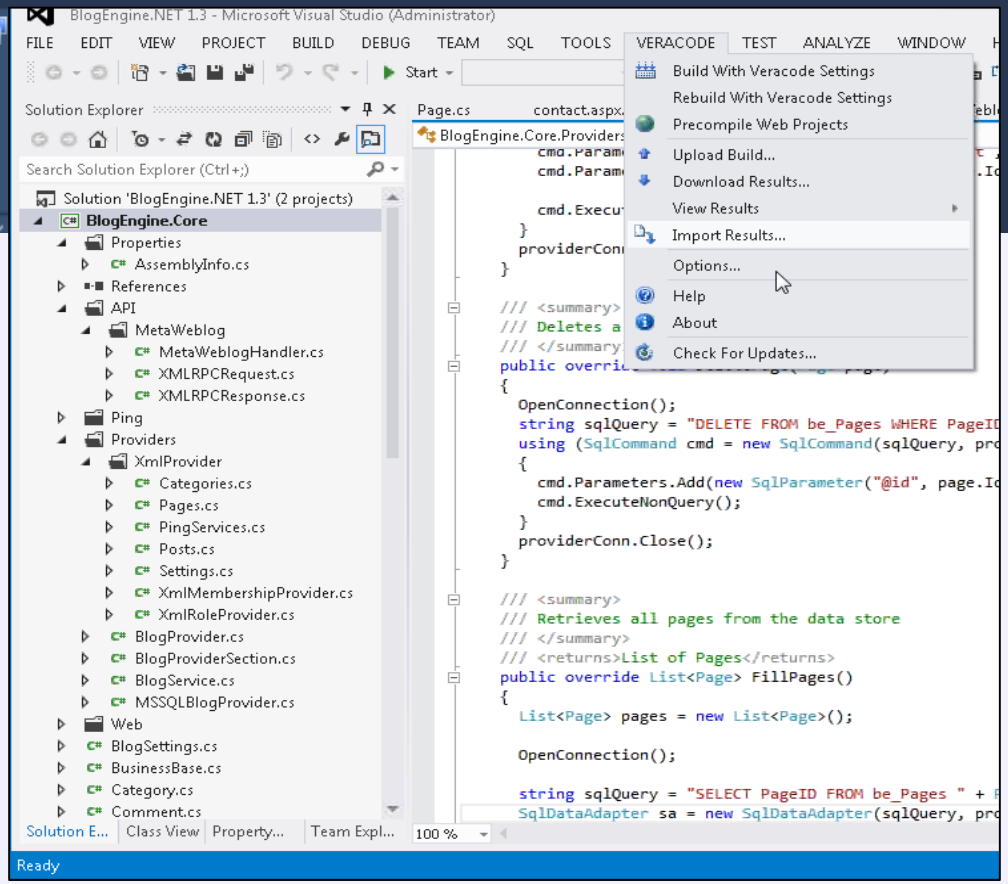
Checkmarx

Entwickler IDE Integration

OWASP
The Open Web Application Security Project



Fortify Visual Studio Plugin



Veracode Visual Studio Plugin

Wenige Hersteller, große Unterschiede

OWASP
The Open Web Application Security Project

- 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 eigner 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

Fazit

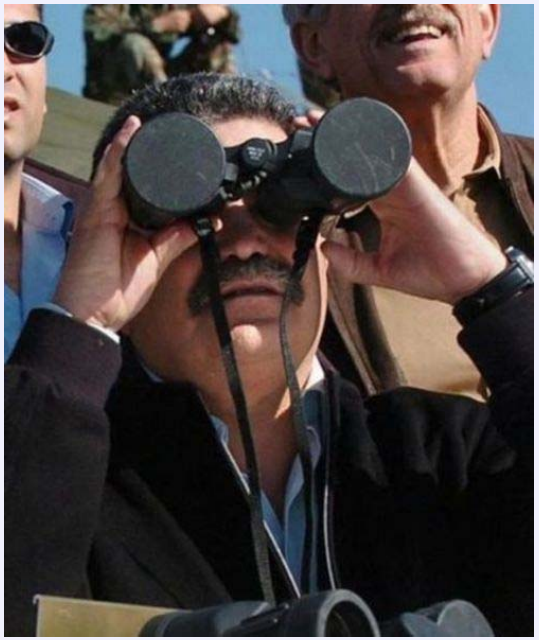
OWASP
The Open Web Application Security Project

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.

Fazit

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?