Kickstart für sichere Webanwendungen

Thomas Schreiber
SecureNet GmbH, München
OWASP Germany Co-Leader

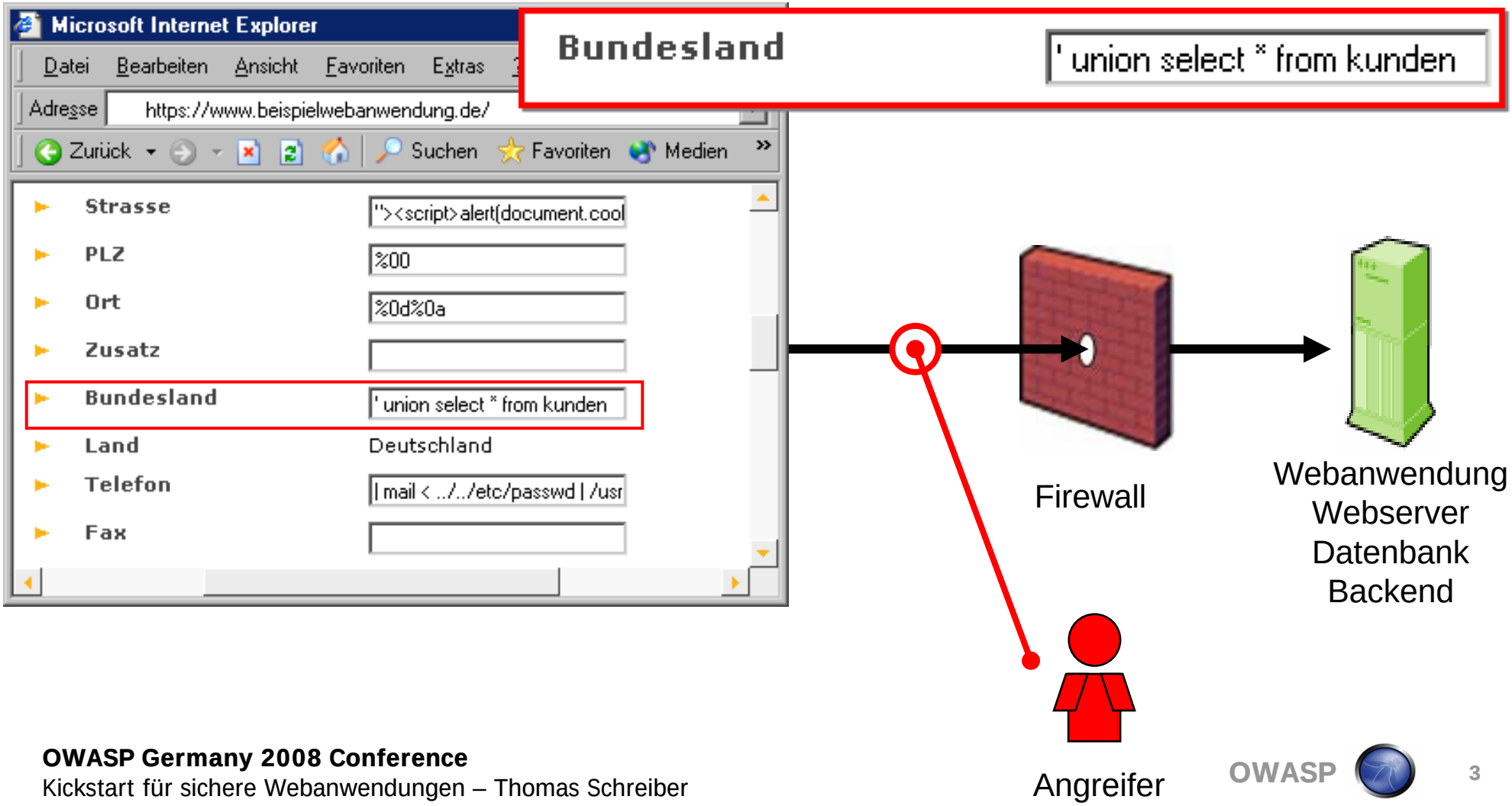
OWASP

Frankfurt, 25.11.08


Ihr Netz ist sicher.
Und Ihre Webanwendung?
Web Application Security by SecureNet
TOR 25
KEIN ZUTRITT!
BEKANNTMACHUNG:
DIESE STADT IST VON EINER STADTMAUER NEUESTER TECHNOLOGIE UMGEBEN!
EROBERUNGSVERSUCHE DAHER ZWECKLOS!!
TOR 80/443
NUR FÜR MARKTBESUCHER
(c) SecureNet GmbH, München

Was ist Web Application Security?

Angriffspunkte

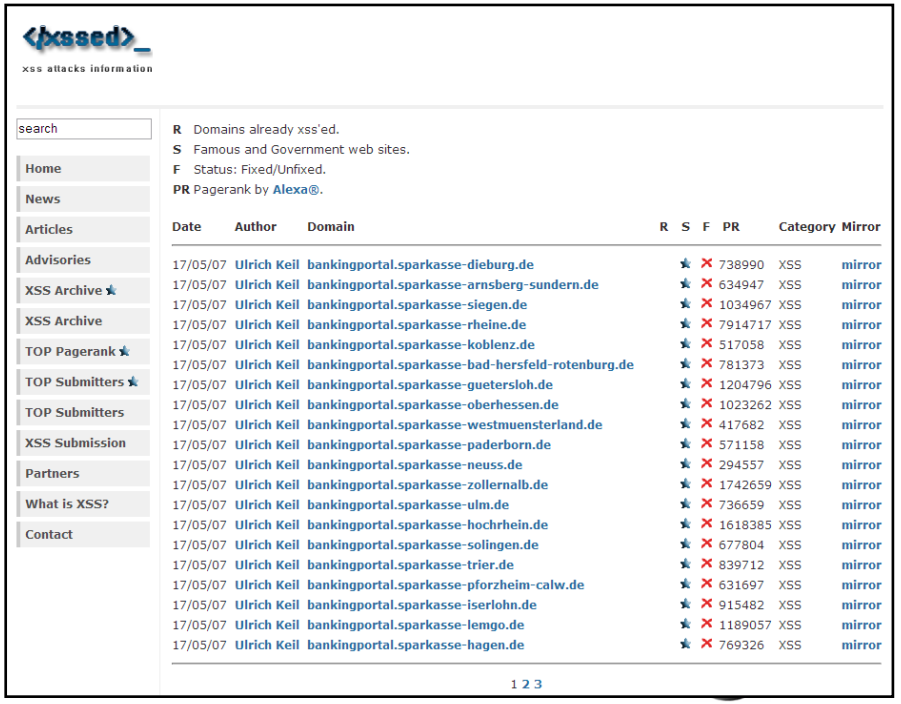


Web Application Security

Aktuelle Situation

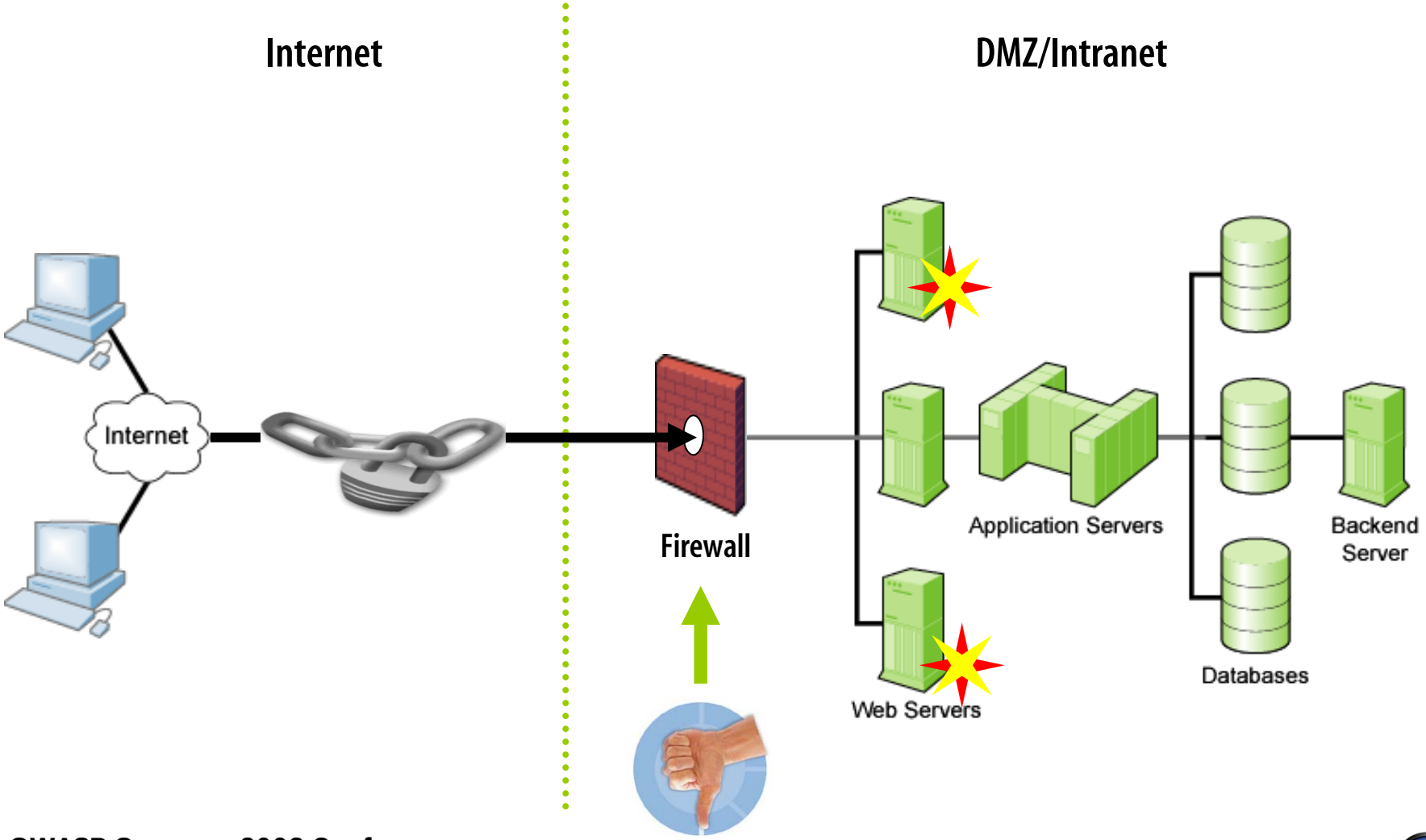
- Eigene Erfahrungen: In 6 Jahren WAS-Pentests und Audits haben wir so gut wie keine Webanwendung angetroffen, die unbeanstandet ist. 8 von 10 Webanwendungen haben ernsthafte Sicherheitsprobleme!
- OWASP: 80% aller Webanwendungen sind unsicher
- Gartner: 75% aller Angriffe erfolgen mittlerweile auf der Anwendungsebene
- PCI Compliance ab 2008 verbindlich! ➔ Codereview oder Pentest erforderlich.

www.xssed.com



Die heutige e-Business-Infrastruktur

... ist ungeschützt



5-Ebenen-Modell

	Ebene	Inhalt
5	Semantik	**Schutz vor Täuschung und Betrug**
4	Logik	**Absicherung von Prozessen und Workflows als Ganzes**
3	Implementierung	**Vermeiden von Programmierfehlern, die zu Schwachstellen führen**
2	Technologie	**Richtige Wahl und sicherer Einsatz von Technologie**
1	System	**Absicherung der auf der Systemplattform eingesetzten Software**
0	Netzwerk & Host	**Absicherung von Host und Netzwerk**

Beispiele:

Phishing-Schutz Informationspreisgabe
"Passwort vergessen"-Fkt. Benutzer Lock-Out
Cross-Site Scripting SQL-Injection
Verschlüsselung Authentisierung
Known Vulnerabilities Konfigurationsfehler

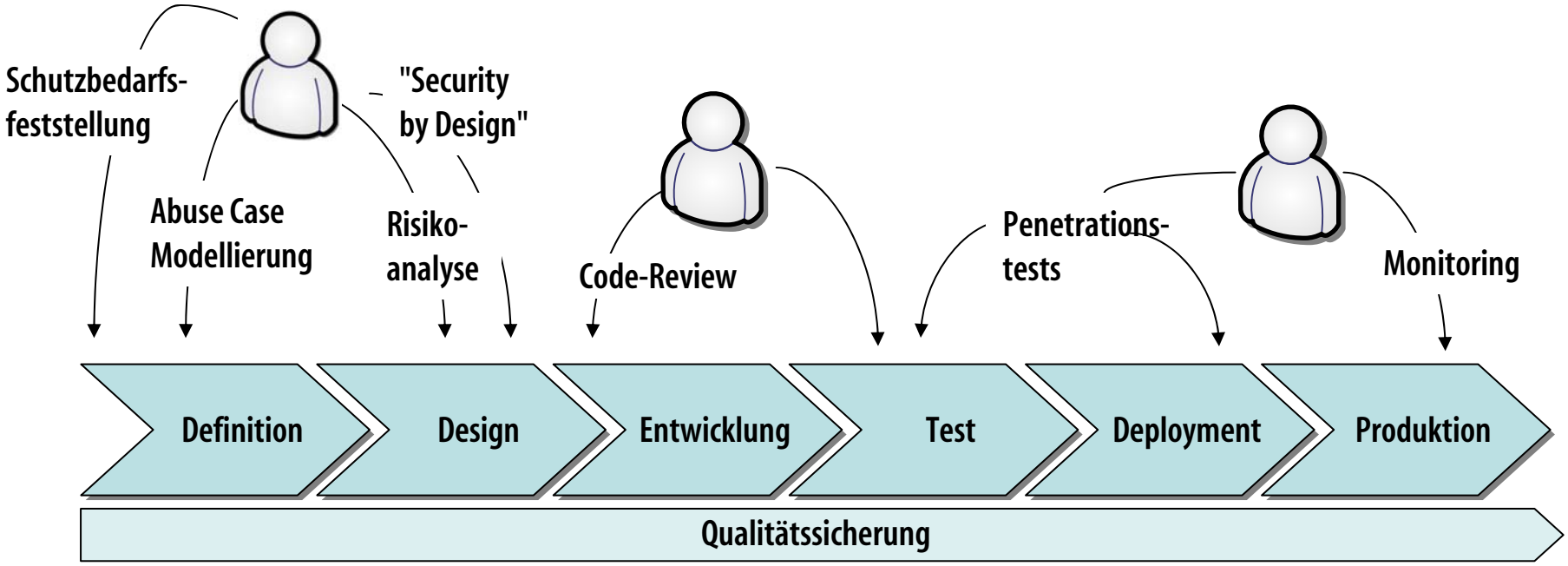
5-Ebenen-Modell

Anwendung

	Ebene	Skills	Tool-unterstützung
5	Semantik	**Corporate Identity und Unternehmens-kommunikation**	. . . . .
4	Logik	**Kenntnisse der Geschäftsprozesse**	■. . . . .
3	Implementierung	**Softwareentwicklungsskills**	■■■■. . . . . (PT) ■■■■■■■. (SCA)
2	Technologie	**Allg. IT-Security**	■■. . . . .
1	System	**Netzwerk- und Systemadministration**	■■■■■■■. .
0	Netzwerk & Host		

Web Application Security

Aktionspunkte im Überblick



Awareness schaffen
Entwickler und Verantwortliche schulen

Verträge mit externen Dienstleistern anpassen
Ausschreibungsbedingungen anpassen

Secure Coding Guidelines
Secure Coding Checkliste

Anpassung Prozesse
und Organisation

Infrastruktur auf WAS-Belange abstimmen
Aufbau einer zweiten Sicherungslinie (WAF...)

WAS != "Layer 8"-Security

- WAS wird nach wie vor als Fortschreibung dessen, was sich im Bereich der klassischen Security bewährt hat, auf die Anwendungsebene verstanden.
- Das ist "suboptimal":
  - Beispiel WAF
  - Beispiel Penetrationstest
  - Beispiel Blackboxansatz
- WAS ist Software Security → ist sichere Softwareentwicklung → ist ein Software Engineering Thema → ist QA (Qualitätssicherung)
  - Sicherheit in Webanwendungen ist ein Qualitätskriterium und sollte auch qualitätssichernden Verfahren folgen

Der Schlüssel für Qualität ist die Umsetzung eines Qualitätssicherungsprozesses – nicht die Suche nach besseren Wegen um Fehler in den fertiggestellten SW-Produkten zu finden und zu beheben.
→ Dieses Prinzip muss auch für die Sicherheit von Software gelten!
Beispiel SQL-Injection

WAS != "Layer 8"-Security

Beispiel: Toolunterstützung / Automatisierung

- Auf Netzwerk- und Hostebene sind automatisierte Verfahren sehr leistungsfähig
  - Nessus & Co.
- Auf der Anwendungebene ist dies weit weniger der Fall (Applicationsecuritscanner)
  - Ganz gut: XSS-Erkennung / Serverfehler und Systemebene / Regressionstests

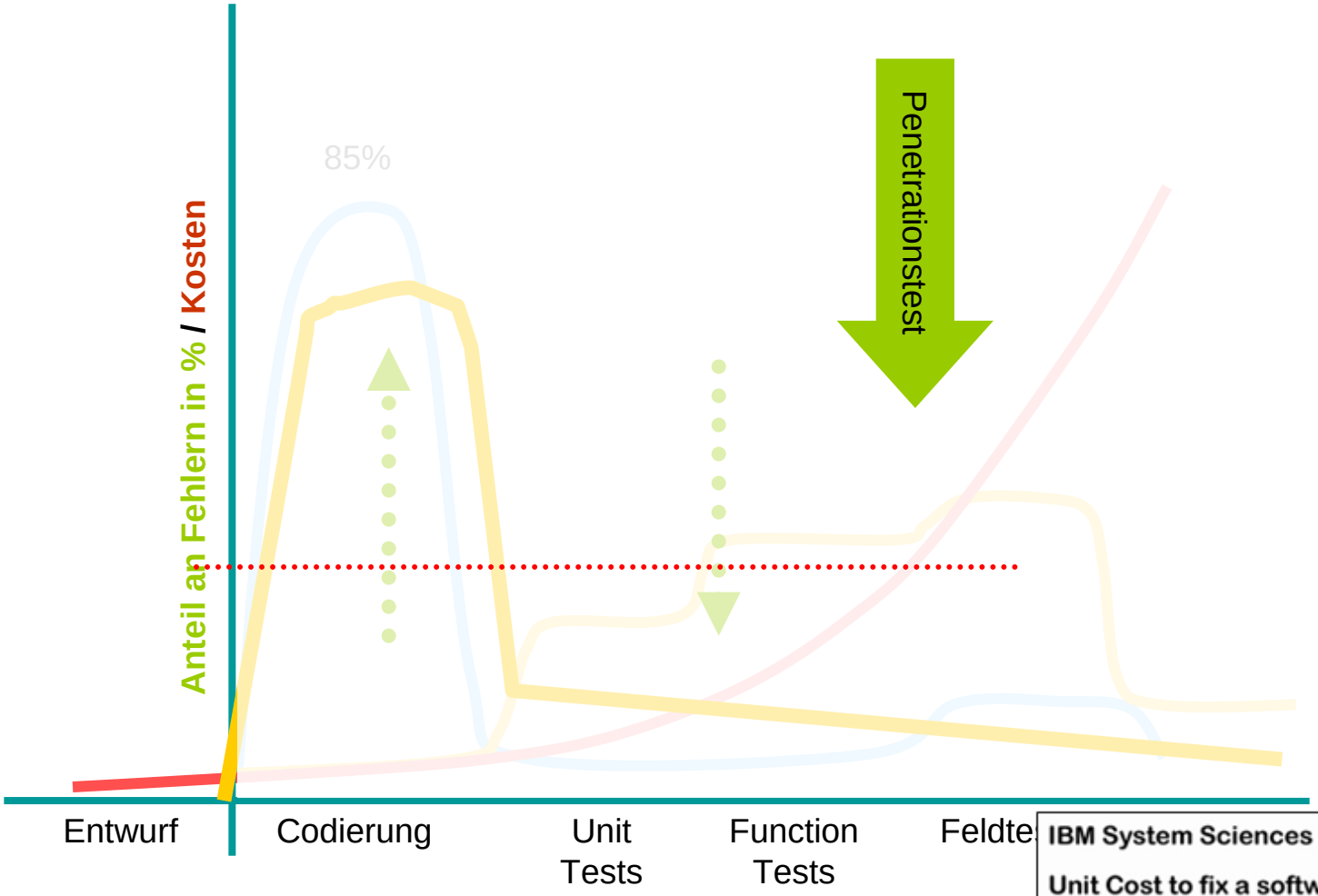
aber:

  - Was wurde getestet (Abdeckungsgrad)
  - Verlust der Session
  - Verlust des Anwendungskontexts / Blockierung
  - Einschränkung auf Anwendung
  - Hoher Konfigurationsaufwand
  - Verhinderung schädlicher Testfälle

BTW: Situation kann deutlich verbessert werden dadurch, dass die Anwendungsarchitektur für automatisches Testen optimiert wird.

WAS != "Layer 8"-Security

Beispiel: Der richtige Zeitpunkt



Quelle: Applied Software Measurement, Capers Jones, 1996

- % an Defects, die in dieser Phase **eingebracht** werden
- % an Defects, die in dieser Phase **gefunden** werden
- Kosten für die Behebung eines Defects bei Entdeckung in dieser Phase

IBM System Sciences Institute Survey

Unit Cost to fix a software security hole in the deployment phase of an SDLC is 100 x that of fixing the same hole at the design stage

Exponential cost the later you catch (leave) it!

SQL-Injection

Die bunte Welt der SQL-Syntax

```
1 OR 1=1
1 OR (1)=(1)
1 OR (DROP TABLE users)=(1)
CONCAT(CHAR(39),CHAR(07),CHAR(39))
1 OR ASCII(2) = ASCII(2)
   ( MD5(), BIN(), HEX(), VERSION(),
   USER(), bit_length(), SPACE() ...)
1 OR 1 IS NOT NULL
1 OR NULL IS NULL


1' HAVING 1 #1 !
1' OR id=1 HAVING 1 #1 !
a'or-1='-1
a'or!1='!1
a'or!(1)='1
a'or@1='@1
a'or-1 XOR'0
1'OR!(false) #1 !
1'OR-(true) #a !
a' OR if(-1=-1,true,false)#!
```

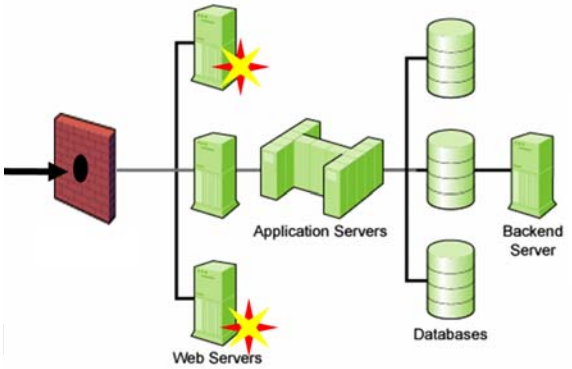
```
1' OR 1&'1
1' OR 1|'1
1' OR 1^'1
1' OR 1%'1
1' OR '1' & 1
1' OR '1' && '1
1' OR '1' XOR '0
1' OR "1" ^ '0
1' OR '1' ^ '0
1' OR '1'|'2
1' OR '1' XOR '0


1 OR+1=1
1 OR+(1)=(1)
1 OR+'1'=(1)
1 OR+'1'=1
1 OR '1'!=0
```

WAS != "Layer 8"-Security

Beispiel: Web Application Firewall

WAFs sind eine zusätzliche Sicherungslinie!



- URL-Filterung (Data-Validation)
- URL-Encryption
- Site Usage Enforcement
  - **Blocken von Zugriffen** abhängig von der Wiederholungshäufig dem Zeitverhalten und Merkmalen wie der IP-Adresse, dem User-Agent- und Referer-Header.
  - **Referer-Checking**: Durch intelligente Prüfung des Referer-Headers, den der Browser bei jedem Zugriff mitschickt, lassen sich (bis zu einem gewissen Grad) Phishing-Attacken und missbräuchliche Verlinkung der eigenen Seiten erkennen und verhindern.
- Behebung von Schwachstellen
  - Aber nur in begründeten Ausnahmefällen
- Lösung des "Patchdilemmas"
- Beispiele: Hyperguard (dt. Hersteller), Airlock (schweizer Hersteller), viele andere…

Vorgehen bei bereits bestehenden Webanwendungen

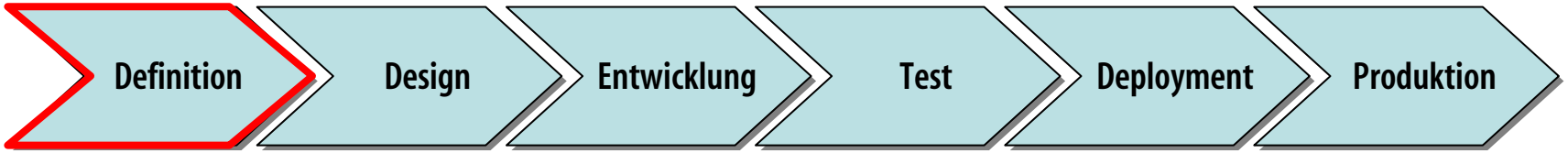
Alte Webanwendungen sind gesondert zu behandeln!

- Charakteristika bestehender Webanwendungen
  - Keine klaren Sicherheitsanforderungen
  - Geringes Verständnis der WAS
  - Frameworks und Architekturansätze ohne 'eingebaute' Sicherheit
  - Nachträglicher Einbau von Sicherheit ist i.d.R. teuer
- Ist-Situation feststellen
  - Sicherheitsanalyse der Website als Ganzes
  - Sicherheitsanalyse jeder einzelnen Webanwendung ("Pentest")
  - Risikoanalyse: Behebung notwendig? / Risiko tragbar?
- Festlegung von Schutzmaßnahmen / Behebung der Schwachstellen
  - Var 1: Schwachstellen in der Anwendung fixen
  - **Var 2: Schwachstellen nicht fixen ➔ Sicherheit am Perimeter herstellen (Application Firewall / WebShield)**
  - Var 3: Neuimplementierung oder Abschalten

➔ Je älter desto unsicherer!

Sicherer SDLC

Definitionsphase



Definitionsphase

- Projektplanung erfolgt
- Anforderungen werden analysiert und definiert
- Pflichtenheft wird erstellt

- **Schutzbedarfsfeststellung**
  - **siehe Schutzbedarfsfeststellung gemäß Kapitel 4.3 der IT-Grundschutz-Vorgehensweise**
- **Abuse Case Modellierung**
- **Risikoanalyse**
- **STRIDE**
- **DREAD**
- **Common Vulnerability Scoring System (CVSS).**

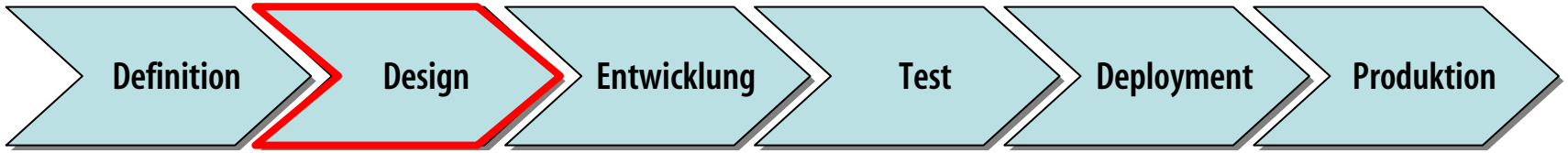
STRIDE / DREAD

- **STRIDE**: Klassifizierungsschema für Art der Bedrohungen/Angriffe
  - **S**poofing
  - **T**ampering
  - **R**epudiation
  - **I**nformation Disclosure
  - **D**enial-of-Service
  - **E**levation of Privilege

- **DREAD**: Klassifizierungs- und Bewertungsschema für das Risiko, das von der betrachteten Bedrohung ausgeht.
  - **D**amage Potential
  - **R**eproducibility
  - **E**xploitability
  - **A**ffected Users
  - **D**iscoverability

Sicherer SDLC

Designphase

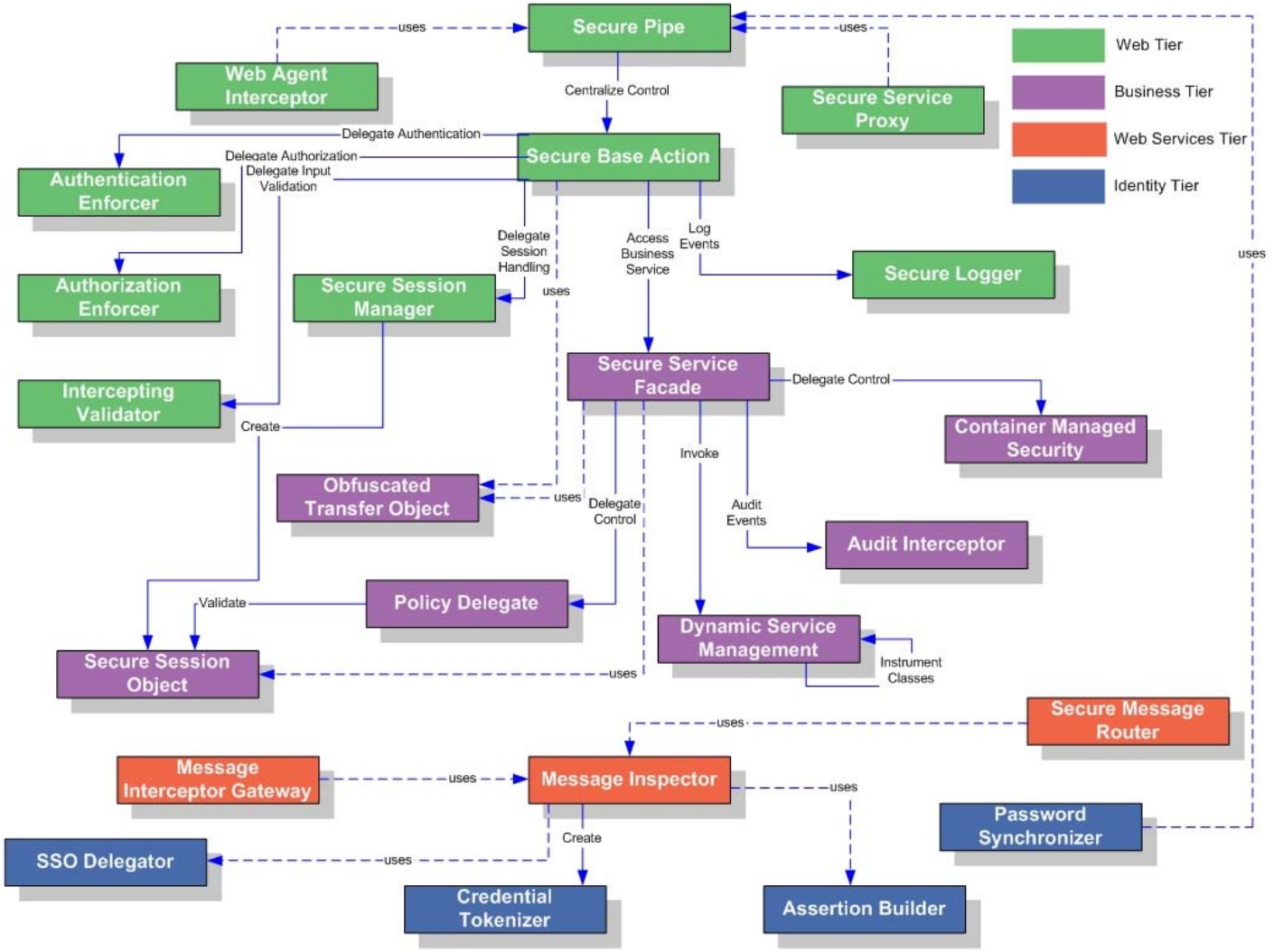


Designphase

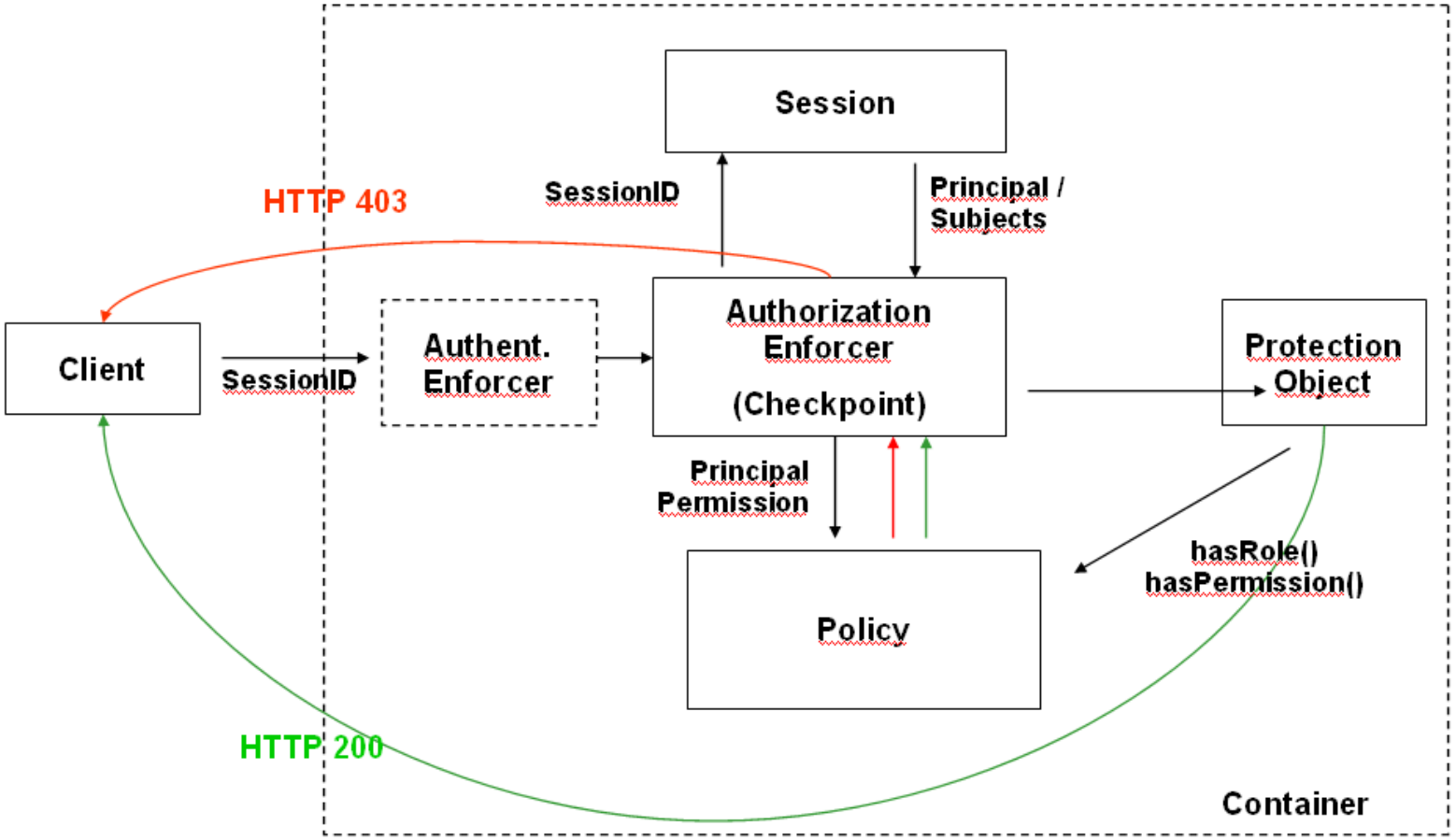
- Fachkonzept wird erstellt
- IT-Konzept wird erstellt
- Softwarearchitektur wird entworfen
- Frameworks, Libraries werden ausgewählt

- **Designpatterns**
- **Security-Erweiterungen**
  - **Java**
    - **OWASP ESAPI**
    - **OWASP Stinger**
    - **OWASP AntiSamy**
    - **HDIV**
    - **CSRFGuard**
  - **.NET**
    - **OWASP ESAPI (Beta)**
    - **OWASP AntiSammy**
  - **PHP**
    - **Suhosin**
    - **OWASP ESAPI (Beta)**

Core Security Patterns Catalog



Pattern: Authorisation Enforcer



Sicherer SDLC

Entwicklungsphase

Entwicklungsphase

- Codierung
- Unit-Tests (Test driven Development)

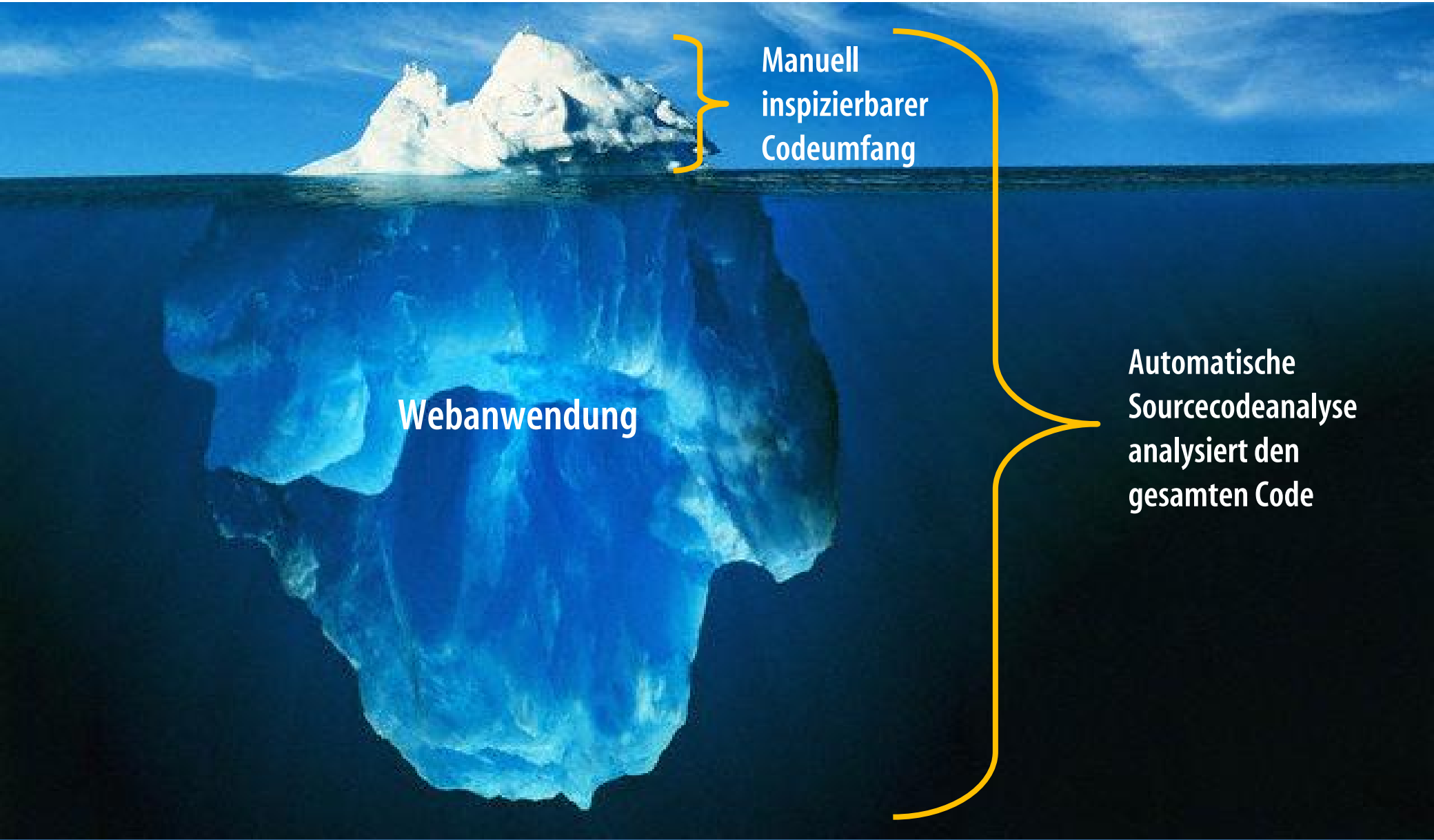
- **Statische Sourcecodeanalyse**
  - **OWASP LAPSE (Java)**
  - **FxCop (.NET)**
  - **Fortify, Ounce**
  - **Manuelle SCA**

Sourcecodeanalyse

Manuelle Sourcecodeanalyse

- Umfassende manuelle Sourcecodeanalyse ist i.d.R. nicht möglich
- "Quick-Wins": Selektive Sourcecodeanaylse von verbreiteten Schwachstellen und leicht auffindbaren Schwachstellen:
  - Inspektion sämtlicher Datenbank-Zugriffe
  - Inspektion sämtlicher Shell-Aufrufe und Systemcalls
  - Inspektion der Verwendung von Benutzereingaben und Input, der von aussen manipulierbar ist
  - Betrachtung des Session Managements
  - Rollen- und Rechtevergabe und Behandlung der Rechte im Kontrollfluß der Anwendung
  - Kapselung des Benutzer- und Sessionkontextes
  - Multithreading, Raceconditions
  - Check auf Buffer-Overflows (C, C++)
  - Exception Handling (Java), Fehlerbehandlung (C, C++)

Sourcecodeanalyse



Sourcecodeanalyse

Automatische Sourcecodeanalyse

- Seit Kurzem existieren leistungsfähige Analysetools
  - Umfassende Analyse des Codes
  - Zu einem Bruchteil der Kosten
  - Kann während der Entwicklung geschehen
  - Hoher Lerneffekt bei Entwicklern
  - Delta-Analysen
  - "Das Pferd am vorderen Ende aufgezäumt"
- Kommerzielle Produkte
  - Fortify
  - Ounce Labs
  - Coverity
- Open Source Tools
  - LAPSE (Java)
  - FxCop (.Net)
  - RATS (C, C++, Perl, PHP, Python)
  - SWAAT (Java, .Net, PHP)

Was Automatische Sourcecodeanalyse leistet

- Dramatische Verbesserung der Sicherheit
  - gezielte Analyse statt stochastische Suche
  - setzt dort an, wo es am effektivsten ist
- Generelle Erhöhung der Qualität
  - Vollabdeckung / hohe Geschwindigkeit
  - Führt auf die Quelle des Problems
- Beeinflussung der Prozesse auf "natürliche" Weise
  - das Projekt bekommt automatisch einen "Drive" in Richtung Sicherheit
- Kontrolle externer Zulieferungen (insbes. Off-Shore)
  - z.B. "Stealth-Backdoors"
- Höchst effektive Schulung der Entwickler
- Nachweis der Compliance
  - PCI und sonstige Vorschriften

➔ SCA ist ein ganz wesentliches Element bei der Herstellung nachhaltiger Sicherheit auf Anwendungsebene.

Sicherer SDLC

Testphase

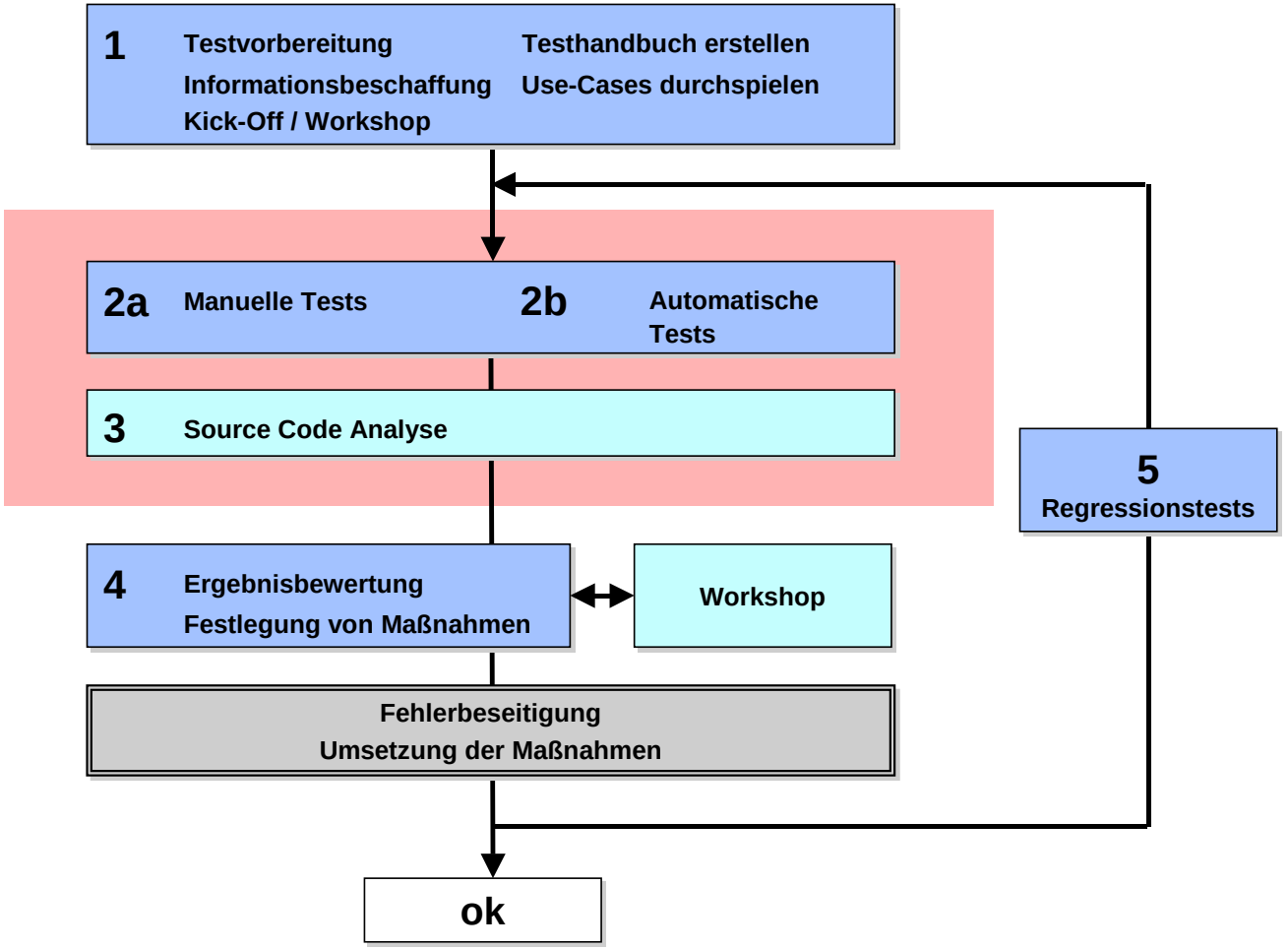
Testphase

- Funktionstest
- Lasttest
- Penetrationstest

- **Statische Sourcecodeanalyse**
  - **siehe vorher**
- **Dynamische Codeanalyse**
  - **auf Bytecodeebene**
  - **Fortify PTA**
- **Penetrationstests der Webanwendung**
  - **Manuelle Penetrationstests**
    - **eine Vielzahl von frei verfügbaren Tools stehen zur Verfügung:**
    - **WebScarab, burp, Paros, ...**
  - **Kommerzielle Application Security Scanner**
    - **WebInspect / AppScan / Hailstorm / Acunetix**
- **Metriken**
  - **Beispiel: Dynamisches SQL entdecken.**

Web Application Security Pentest

Idealisierter Ablauf



Web Application Security Pentest

- WAS Penetrationstests erfolgen heute häufig manuell
  - weil: effizienter
  - weil: größere Abdeckung
  - weil: besser steuerbar
- Zuhilfenahme einer Vielzahl von Tools
- Verwendung automatische Scanner wenn sinnvoll
  - AppScan
  - Webinspect
  - Hailstorm
  - Acunetix
- Betrachtung aller 5 Ebenen der WAS

WAS != "Layer 8"-Security

Beispiel: Blackboxansatz

- Blackbox ←→ Whitebox
  - Blackboxtest kommt manchmal der Suche nach der Stecknadel im Heuhaufen gleich
  - Der Pentester ist dem Angreifer in der Regel unterlegen
    - Angreifer: Aufwand und Intensität sind keine Grenzen gesetzt // proportinal zu Motivation und Interessenslage
    - Pentester testet alleine // Zahl der Angreifer kann beliebig hoch werden
  - Je umfassender die Informationen, desto höher die Qualität des Ergebnisses

  ➔ **Gib dem Tester ein Maximum an Informationen!**
    - Benutzerkennungen für sämtliche Rollen (auch speziellen Admin-Zugang testen)
    - Architekturbild/-beschreibung / Fach-/IV-Konzept/Pflichtenheft
    - Beschreibung Use-Cases
    - Schnittstellenbeschreibungen
    - Benutzerhandbuch / Bedienungsanleitung
    - QA Protokolle / Lasttest Ergebnisse
    - Ergenisse von Risiko Analyse / Threat Model

Web Application Security Pentest

Kriterien

- Mit oder ohne WAF testen?
  - Die Wurzel des Problems ist in der Anwendung zu suchen.
  - ➔ **Nur in begründeten Fällen ist Pentest mit eingeschalteter WAF sinnvoll.**
- Testumgebung ←→ Produktivumgebung
  - Testumgebung: Datenverlust ist nicht zu befürchten // Rücksichtnahme auf Produktivbetrieb nicht erforderlich
  - Produktivumgebung: Sämtliche realen Einflüsse gehen in den Test mit ein.
  - ➔ **In der Regel ist ein Test in einer Testumgebung (möglichst exaktes Abbild der Produktivumgebung!) vorzuziehen.**
- Internfunktionen mittesten?
  - Viele Webanwendungen haben Funktionsbereiche (Worksflows, Admin), die nicht im Internet zugänglich sind.
  - ➔ **Diese sollten in die Tests mit einbezogen werden, da vielfältige Angriffsmöglichkeiten bestehen.**

Web Application Security Pentest

Ergebnisbericht

- Benennung des Gefahrenpotentials mit

  OK niedrig mittel hoch

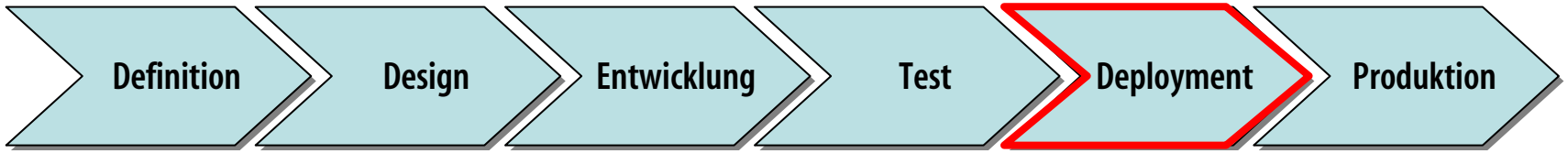
- Nicht nur Findings, auch OK-Fälle berichten
- Beschreibung von
  - der jeweiligen Schwachstelle, Bedrohung, Angriffstechnik
  - Mögliche Angriffsszenarien (damit der Verantwortliche in die Lage versetzt wird, das Schadenspotential zu beurteilen)
  - Exakte Darstellung des Testfalls mit Beispielen
  - Maßnahmenempfehlungen
  - Verweise auf weiterführende Darstellungen

SCA ←→ Pen Testing

Pen Testing	SCA
Suche nach der Stecknadel im Heuhaufen	Systematischer, umfassender Ansatz
Angreifer ist (sind!) dem Tester immer überlegen	Grundsätzliche Überlegenheit des Testers
Findings müssen vom Entwickler erst verstanden und lokalisiert werden	Findings sind bereits in der Sprache des Entwicklers
Tatsächlich erzielte Abdeckung schwer messbar	Vollabdeckung
Zum Zeitpunkt des Tests nicht 'sichtbare' Funktionen werden nicht getestet	Vollabdeckung
Kommt sehr spät im SDLC zum Einsatz (kurz vor dem Deployment)	Begleitet den SDLC vom Beginn der Entwicklungsphase
Ist nachträglich aufgesetzte Sicherheit	Ist inhärente Sicherheit
Anwendung muss fertig sein, bevor sie getestet werden kann	Komponententests sind möglich
Geringer Beitrag zur Schulung der Entwickler	Starke Rückwirkung auf die Verursacher und den Prozess
Schlecht geignet für Agile Programmierung	Sehr gut geeignet für Agile Programmierung
Liefert Aussage über die Sicherheit des Gesamtsystems (Webserver etc.)	Aussage nur bezogen auf die Anwendung
Leichte Durchführbarkeit	Sourcecode und Umgebung müssen bereitgestellt werden.
Einige Vulnerabilities sind sicher und leicht auffindbar	Bei einigen tut SCA sich schwer

Sicherer SDLC

Deploymentphase



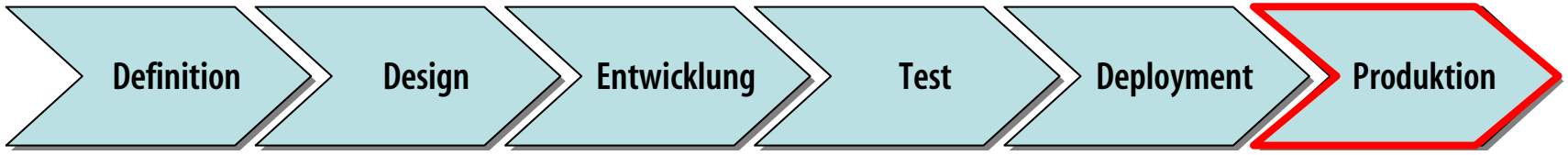
Deploymentphase

- Übernahme der Anwendung in die Produktion
- Integrationstests

- **Audit der Konfiguration**
  - **Webserver**
  - **Datenbanken**
  - **zB. PHPSecinfo**
  - **etc**
- **Penetrationstest des Systems**
  - **Penetrationstest des Webservers**
    - **liegengebliebene Files, ...**
  - **Penetrationstest der Anwendung**
    - **z.B. Session Management**

Sicherer SDLC

Produktivbetrieb



Produktivbetrieb

- Nutzung der Anwendung
- Wartung

- **Patchmanagement**
  - **Einspielen von Updates**
- **Monitoring**
  - **Angriffsversuche erkennen (=> Impulse für die Entwicklung)**
  - **Einbrüche erkennen**
  - **Bedrohungsszenarien verifizieren**

Web Application Firewall

siehe auch...

- OWASP Best Practices Guide zum Einsatz von Web Application Firewalls
  - http://www.owasp.org/images/1/1b/Best_Practices_Guide_WAF.pdf
  - Autor: Mitglieder des deutschen OWASP Chapters!
- Web Application Firewall Evaluation Criteria
  - http://www.webappsec.org/projects/wafec/

Web Application Security

Den SDLC mit Sicherheit ausstatten

Sicherer Softwareentwicklungs-prozess

- Methodisches Vorgehen über alle Phasen des SDLC
- Management von Risiken
- Risiken und Sicherheit meßbar machen

- **Verfahren**
  - OWASP **CLASP**
  - Cigital **Touchpoints**
  - Cigital **SSF (Software Security Framework)**
  - Fortify **Framework*BSA (Business Software Assurance)**
  - Microsoft **SDL** (eher nicht für Webanwendungen)

OWASP CLASP Projekt

Rollen im SDLC

CLASP: Comprehensive, Lightweight Application Security Process

- Methodischer Ansatz, um die Sicherheit in die frühen Phasen des Softwareerstellungsprozesses einzubringen
- Rollen
  - Architect
  - Designer
  - Implementer
  - Project Manager
  - Requirements Specifier
  - Security Auditor
  - Test Analyst

Eine Person kann mehrere Rollen übernehmen

http://www.owasp.org/index.php/Category:OWASP_CLASP_Project

OWASP CLASP Projekt

CLASP Best Practices

1. **Awareness herstellen**
2. **Sicherheitsanalysen / -audits / -pentests durchführen**
3. **Sicherheitsanforderungen ermitteln**
4. **Einführung von Sicherheit in die Entwicklung**
5. **Einführung von Prozessen zur Behebung von Sicherheitsmängeln**
6. **Definition und Überwachung von Sicherheitsmetriken**
7. **Veröffentlichung von Guidelines für den Betrieb**

Anwendungssicherheit herstellen

OWASP Materialien

Release Quality Projects

Release quality projects are generally the level of quality of professional tools or documents.

We have started the process of defining detailed guidelines which indicate what will be required from an OWASP Project in order for it to be classified an OWASP Release quality project (see Project Assessment Criteria). Please note that the projects below have **NOT** been evaluated under this criteria and might be re-classified once that process is completed.

Tools

OWASP WebGoat Project
an online training environment for hands-on learning about application security

OWASP WebScarab Project
a tool for performing all types of security testing on web applications and web services

Documentation

OWASP AppSec FAQ Project
FAQ covering many application security topics

OWASP Development Guide Project
a massive document covering all aspects of web application and web service security

OWASP Legal Project
a project focused on contracting for secure software

OWASP Testing Guide
a project focused on application security testing procedures and checklists

OWASP Top Ten Project
an awareness document that describes the top ten web application security vulnerabilities

Anwendungssicherheit herstellen

siehe auch...

Maßnahmenkatalog und Best Practices zur Sicherheit von Webanwendungen

http://www.bsi.de/literat/studien/websec/WebSec.pdf

iX-Artikel 03/2007:
"Best Practices für sichere Webanwendungen"

http://www.securenet.de/download/Best_Practices_fuer_sichere_Webanwendungen_iX0703.pdf

Vielen Dank!

Thomas Schreiber
SecureNet GmbH, München
OWASP Germany Co-Leader

OWASP
Frankfurt, 25.11.08