OWASP
Open Web Application Security Project

Nachlese

Münchner OWASP-Stammtisch, 24.01.2017

Achim Hoffmann
Thomas Herzog
Torsten Gigler

Vorträge (1)

OWASP
Open Web Application Security Project

Name	Vortrag
Martin Knobloch	**OWASP 101** Kurzvorstellung von OWASP und einzelner Projekte ⇨ ***nicht enthalten***
Tobias Millauer (Daimler)	**Keynote:** **CarIT Security: Facing Information Security Threats**
Christian Schneider	**Java Deserialization Attacks -** Angriff & Verteidigung
Daniel Kefer and René Reuter	**Security Requirements im Software Development Lifecycle**
Thomas Patzke	**Durchsuchen und analysieren von HTTP-Daten mit dem WASE-Framework**

Vorträge (2)

OWASP
Open Web Application Security Project

Name	Vortrag
Andreas Falk	**Sicher in die Cloud mit Angular 2 und Spring Boot**
Siegfried Rasthofer	**Reverse Engineering Android Apps With CodeInspect** ⇨ ***nicht enthalten***
Patrick Spiegel	**NoSQL Injection revisited**
Lukas Weichselbaum, Michele Spanguolo, Artur Janc und Sebastian Lekies	**CSP Is Dead, Long Live CSP!** On the Insecurity of Whitelists and the Future of Content Security Policy
Matthias Rohr	**Sicherheit agil Testen**
Sebastian Schinzel	**DROWN** - oder warum TLS-Konfiguration schwer ist

Lightning Talks

OWASP
Open Web Application Security Project

Name	Vortrag
Bastian Braun	**Der Secure Development Lifecycle in der agilen Praxis** ⇨ ***nicht enthalten***
Björn Kimminich	**What's new in OWASP Juice Shop?**
Juraj Somorovsky	**TLS-Attacker** Systematic Fuzzing and Testing of TLS Libraries

https://www.owasp.org/index.php/German_OWASP_Day_2016#Programm

Keynote: CarIT Security: Facing Information Security Threats (1) [Tobias Millauer (Daimler)]

OWASP
Open Web Application Security Project

Herausforderungen

- **Safety ist eine der wichtigsten Eigenschaften eines Autos**
- **The connected car ⇒immer komplexere Vernetzung von Fahrzeugen mit externen Netzen**
- **Das Problem: Das Auto ist auf so etwas nicht vorbereitet**
- **Annahme: völlig isoliertes Netzwerk**
- **Die Protokollstandards im CAN-Bus enthalten bisher nur wenige Sicherheitseigenschaften**
- **⇒ Jeder mit Zugriff stellt ein erhebliches Risiko dar.**
- **⇒ Bedrohung für Safety**

Zusatzinformationen: Developments in Car Hacking (www.sans.org)
https://www.sans.org/reading-room/whitepapers/ICS/developments-car-hacking-36607

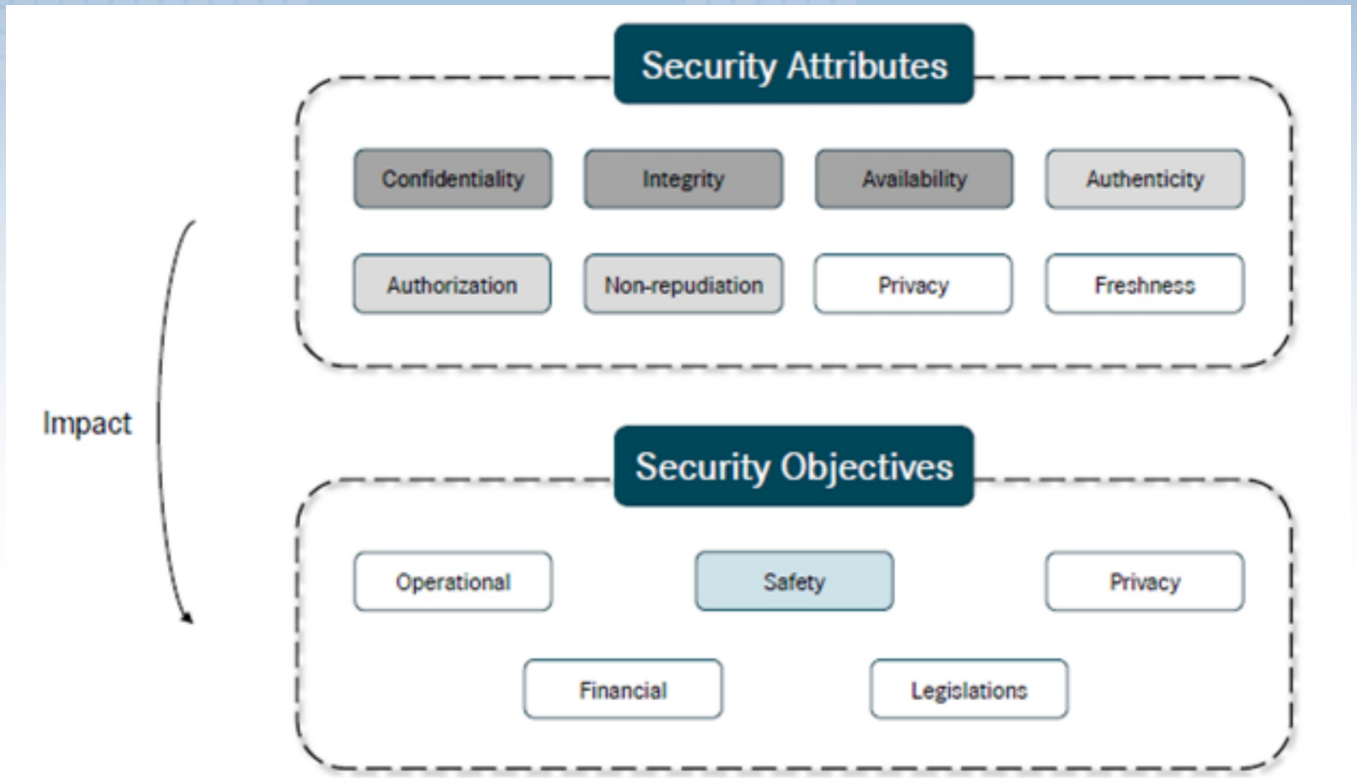
Keynote: CarIT Security: Facing Information Security Threats (3)
[Tobias Millauer (Daimler)]

OWASP
Open Web Application Security Project

HEAVENS Security Model

(HEAling Vulnerabilities to ENhance Software Security and Safety)

⇒**Security aspects in safety engineering**



Keynote: CarIT Security: Facing Information Security Threats (4) [Tobias Millauer (Daimler)]

OWASP
Open Web Application Security Project

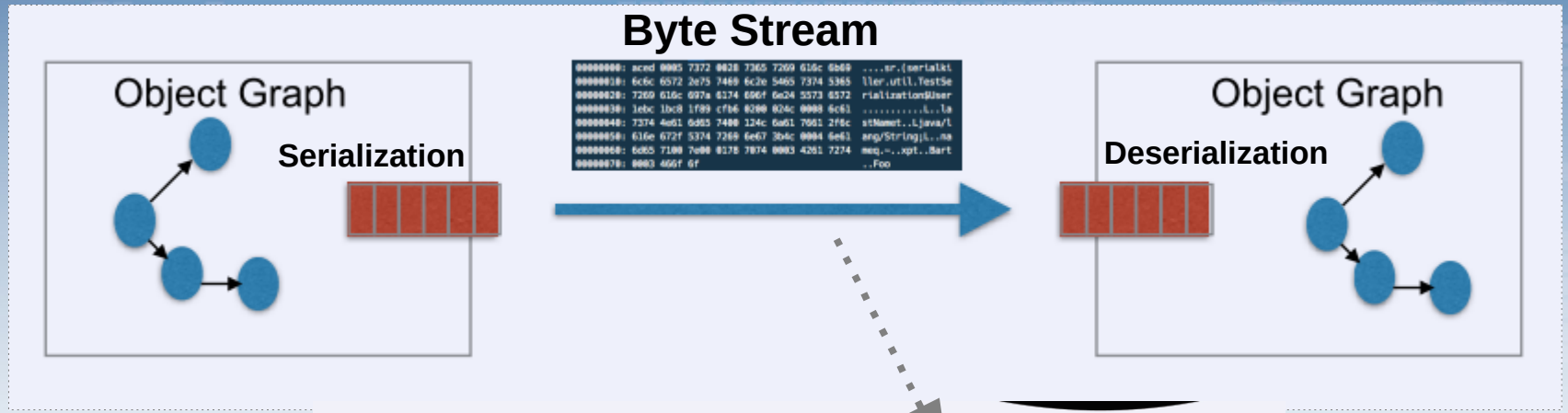
Vehicle Cybersecurity

- **Protective/preventive measures and techniques**
  These measures, such as isolation of safety-critical control systems networks or encryption, implement hardware and software solutions that lower the likelihood of a successful hack and diminish the potential impact of a successful hack.
- **Real-time intrusion (hacking) detection measures**
  These measures continually monitor signatures of potential intrusions in the electronic system architecture.
- **Real-time response methods**
  These measures mitigate the potential adverse effects of a successful hack, preserving the driver's ability to control the vehicle.
- **Assessment of solutions**
  This involves methods such as information sharing and analysis of a hack by affected parties, development of a fix, and dissemination of the fix to all relevant stakeholders.

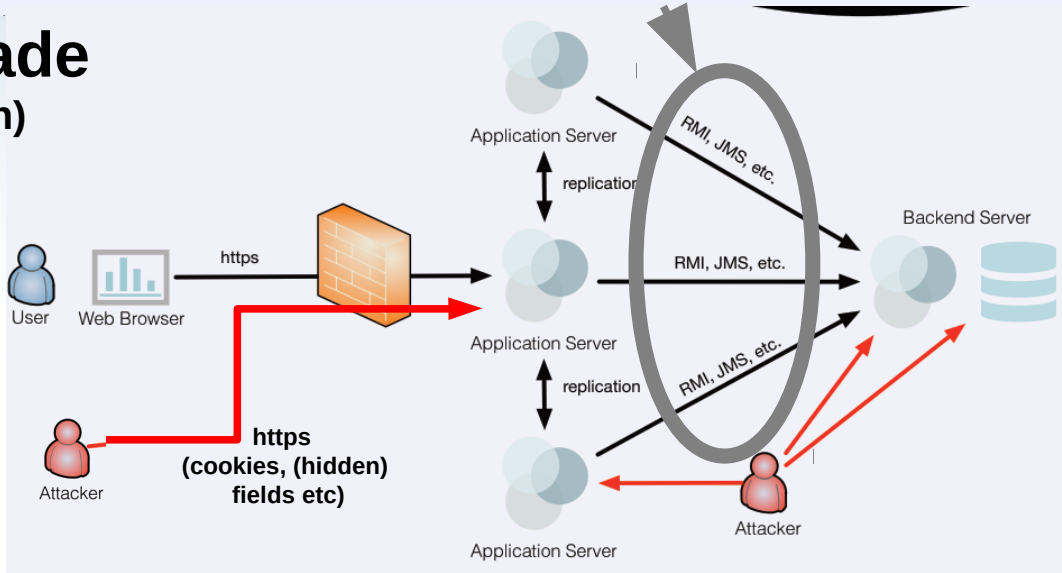
Java Deserialization Attacks (1)
(Angriff & Verteidigung)
[Christian Schneider]

OWASP
Open Web Application Security Project

Java Serialization ➔ Deserialization



Angriffspfade
(intern und extern)



Java Deserialization Attacks (2)
(Angriff & Verteidigung)
[Christian Schneider]

OWASP
Open Web Application Security Project

Handlungsempfehlungen

- **Remove Gadget** → **zu viele 'Gadgets' in versch. Klassen**
- **AdHoc Security Manager** → **Angriffspfad über 'Garbage Collector' bleibt (via finalize)**
- **Defensive Deserialization** → **Blacklists können umgangen werden**

⇨ **KEINE nicht-vertrauenswürdigen Daten deserialisieren**
(andere Verfahren zur Datenübertragung nutzen; z.B. JSON, XML)

⇨ **Zweit-beste Lösung: defensive Deserialisierung mit voraus-schauender ObjectInputStream-Prüfung, z.B. SWAT,**
Risiken: Prüfung kann selbst zum Einfallstor werden, DOS-Gefahr

⇨ **Nach Deserialisierung suchen**
- Codeanalyse
- Pentest: Tools, oder Netzwerktrace mit den Magic Bytes
`0xAC 0xED|Base64: rO0AB|Base64+Kompr.:0x1F8B 0x0800 bzw. H4sIA …`

Durchsuchen und analysieren von HTTP-Daten mit dem WASE-Framework (1) [Thomas Patzke]

OWASP
Open Web Application Security Project

WASE – Web Audit Search Engine

- HTTP-Requests und -Responses nach bestimmten Eigenschaften durchsuchen/filtern
- WASE-Framework: Elasticsearch, Kibana WASE
  - WASE: ElasticBurp, WASEProxy, WASEQuery

- https://github.com/thomaspatzke/WASE
- Live Demo: http://wase-demo.patzke.org

Durchsuchen und analysieren von HTTP-Daten mit dem WASE-Framework (2) [Thomas Patzke]

OWASP
Open Web Application Security Project

WASE Quotes

Try to do one of the following with your tool of choice:

- Search all POST requests that don‘t contain a CSRF token
- List all values of a parameter or cookie that encountered while a web application security test
- List all values of a security header with its corresponding URL
- List all URLs where inferred content type is HTML while the server tells something different about its content type
- Show all HTML responses without a doctype definition
- Find all external script references
- Discover unsafe or nonse HTTP security header values

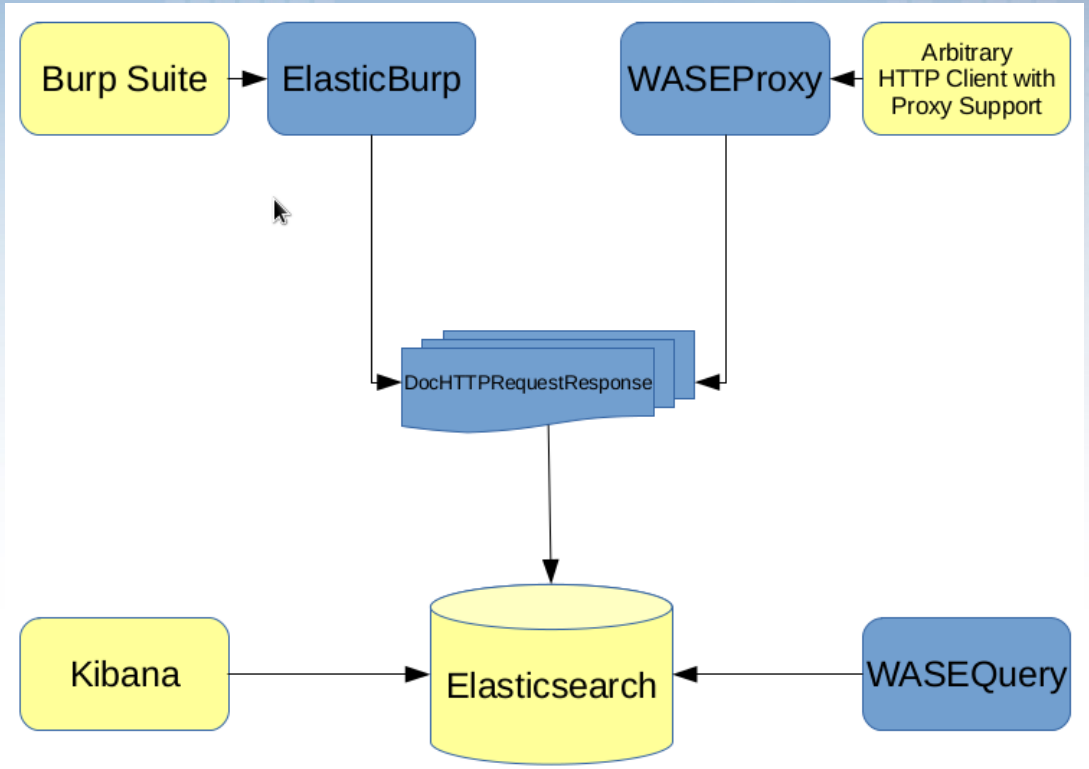
  ⟶ Complex searches and analytics in web

- application security tests
- mass scans of web sites
- malware analysis

Durchsuchen und analysieren von HTTP-Daten mit dem WASE-Framework (3) [Thomas Patzke]

OWASP
Open Web Application Security Project

- Elasticsearch: search and analytics engine for textual data
- Kibana: web frontend for Elasticsearch
- WASE:
  - Definition of a data structure for HTTP
  - requests/responses for Elasticsearch
  - ElasticBurp, WASEProxy, WASEQuery



Security Requirements im Software Development Lifecycle (1)

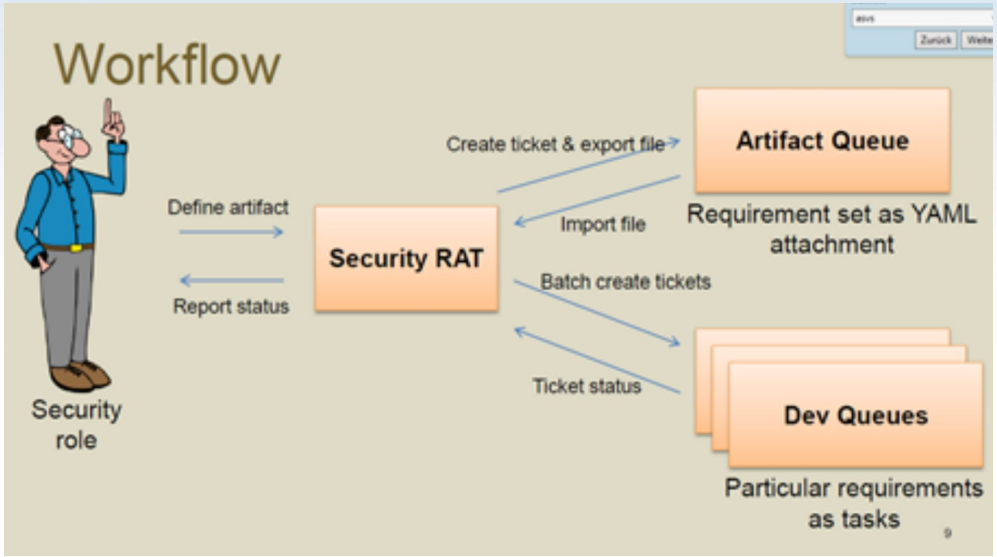
[Daniel Kefer and René Reuter]

German OWASP Day 2016 - Darmstadt - 2016

OWASP Open Web Application Security Project

Management von Security Requirements

- Security RAT (Requirements Automation Tool)
- Systemeigenschaften (z.B. Security Header) & Lifecycle-Aktivitäten (z.B.Pentests)
- leicht anpassbar an konkrete Software-Architektur
- Integration in Ticketsystem (Jira)
- weitgehende Automatisierung



Security Requirements im Software Development Lifecycle (2)

[Daniel Kefer and René Reuter]

German OWASP Day 2016 - Darmstadt - 2016

OWASP Open Web Application Security Project

Technologie:

https://github.com/SecurityRAT

bisher drei Projekte:

- **SecurityRAT (das eigentliche Tool)**
- **Security-Requirements (das voreingestellte Requirementset)**
- **securityrat.github.io (Dokumentation)**

Sicher in die Cloud mit Angular 2 und Spring Boot (1) [Andreas Falk]

OWASP
Open Web Application Security Project

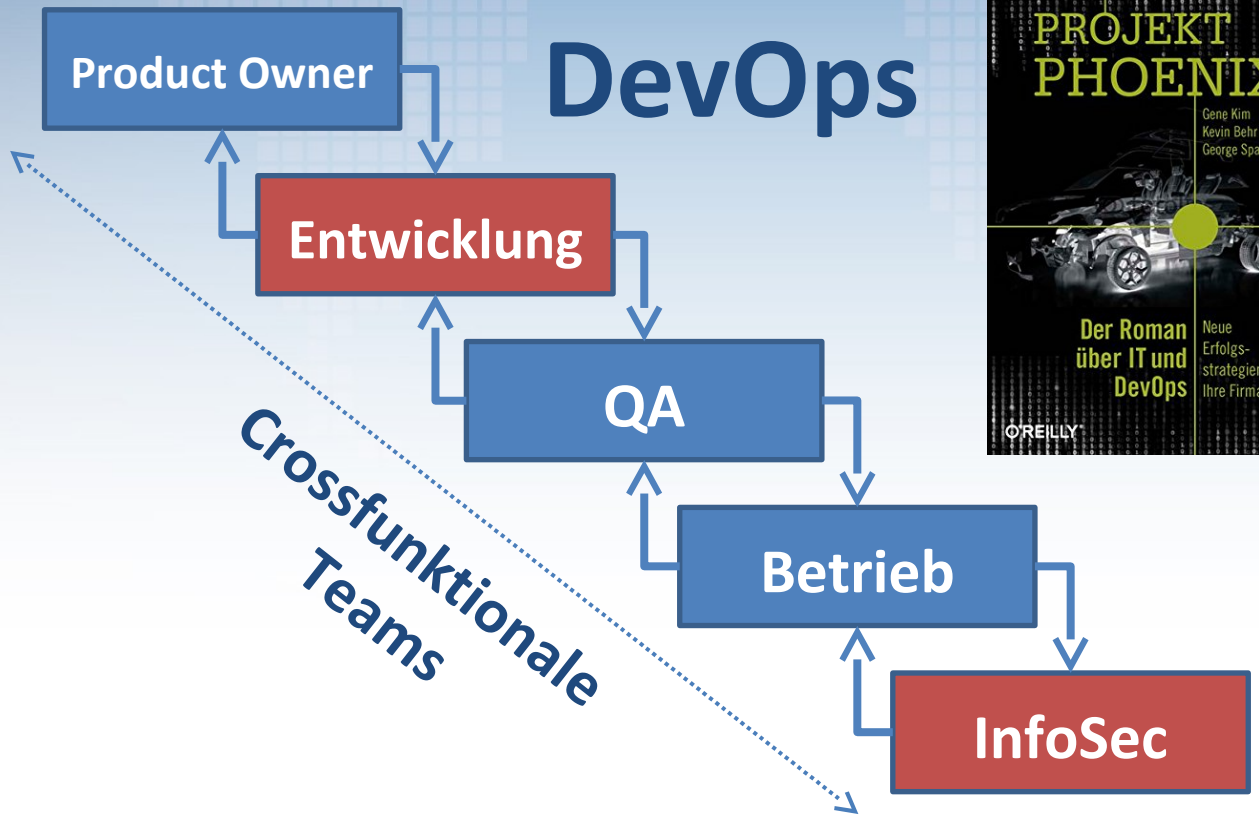
- Unterschied Angular 1 vs. 2 (in Bezug auf Sicherheit)
- Sicherheits-“Funktionen“ von Angular 2
- Sicherheits-“Funktionen“ von Spring Security
- Sicherheits-“Funktionen“ von Spring Data JPA
- Beispiele mit OAuth2, OpenID

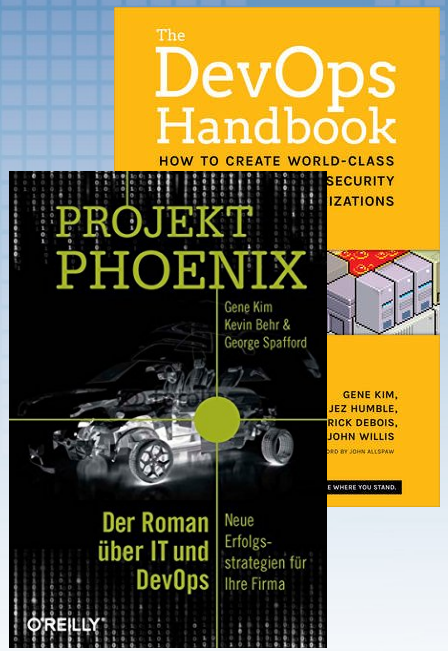
Sicher in die Cloud mit Angular 2 und Spring Boot (2) [Andreas Falk]

German OWASP Day 2016 - Darmstadt - 2016

OWASP Open Web Application Security Project

DevOps





Sicher in die Cloud mit Angular 2 und Spring Boot (3) [Andreas Falk]

OWASP
Open Web Application Security Project

- **CloudFoundry: Rotate – Repair – Repave**
  - **Repave: „*What if every server inside my data center had a maximum lifetime of two hours?***
  - ***This approach would frustrate malware writers, because it limits the amount of time to exploit known vulnerabilities before they are patched.*“**

NoSQL Injection Revisited (1)
[Patrick Spiegel]

OWASP
Open Web Application Security Project

Im Fokus (DB, Typ)

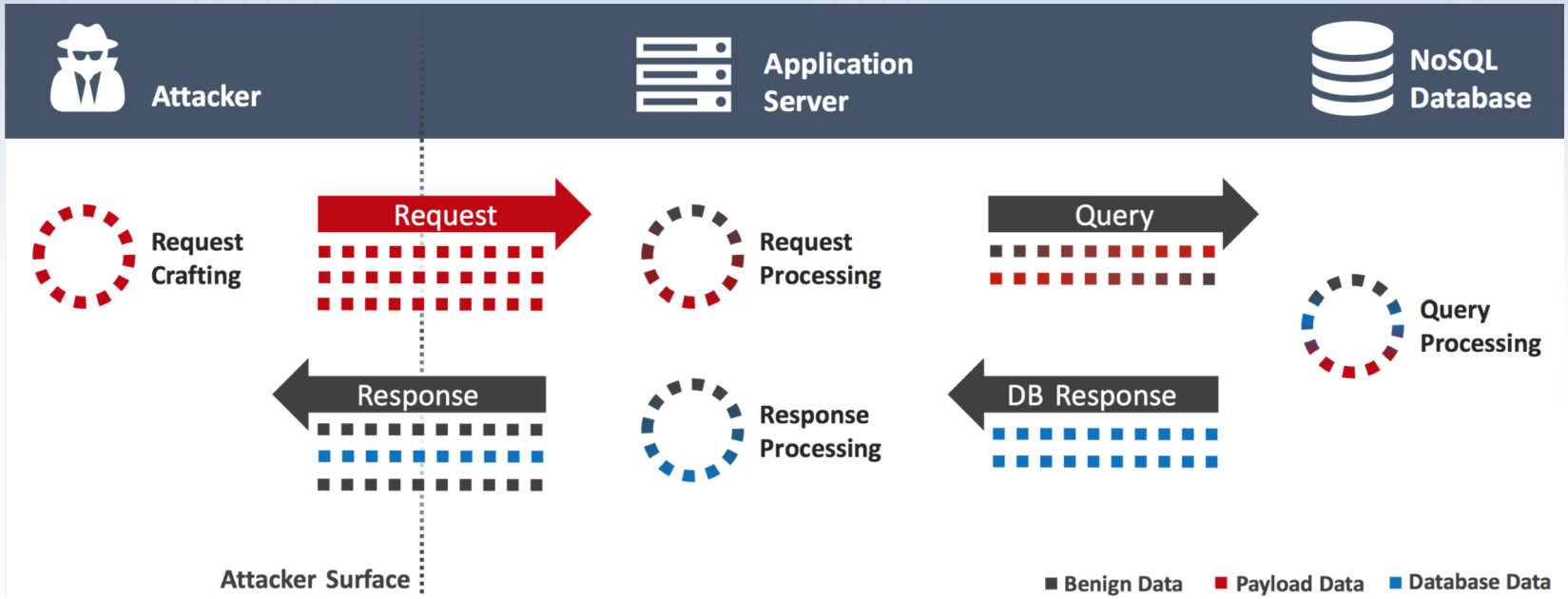
mongoDB Document store

MEMCACHED Key-value cache

redis Key-value store

CouchDB relax Document store

Angriffspfad



NoSQL Injection Revisited (2)
[Patrick Spiegel]

OWASP
Open Web Application Security Project

Beispiele für Angriffe

Login bypass (z.B. MongoDB)

```
// NodeJS with Express.js
db.collection('users').find({
  "user": req.query.user,
  "password": req.query.password
});
```

Beispiel-Angriffe

https://example.org/login?user=patrick&password[%26ne]=

"password": {"&ne": ""}

Check bypass (z.B. CouchDB)

```
// NodeJS with Express.js
function getDocument(key, callback) {
  if (key === "secretDoc" || key[0] === "_") {
    callback("Not authorized!");
  } else {
    couch.use('documents').get(key, callback);
  }
}
getDocument(req.query.key);
```

https://example.org/get?user[]=secretDoc

Array-Injection

https://example.org/get?user[]=_all_docs

NoSQL Injection Revisited (3)
[Patrick Spiegel]

OWASP
Open Web Application Security Project

Beispiele für Verteidigungsmaßnahmen

Is type casting a solution

```
// NodeJS with Express.js
db.collection('users').find({
  "user": req.query.user.toString(),
  "password": req.query.password.toString()
});
```

Beispiel-Angriffe

https://example.org/login?user=patrick&password[%26ne]=

- ☑ Secure against type manipulation
- ☒ Not flexible enough for unstructured data
- ☒ Easy to forget in practice ...

CSP Is Dead, Long Live CSP!

On the Insecurity of Whitelists and the Future of Content Security Policy (1)

[Sebastian Lekies (Vortragender) u.A.]

OWASP
Open Web Application Security Project

Do CSP-Policies work in practice?

Google-Index mit 100 Mrd. Seiten

- ⇒ **1.6 Mio Hosts mit CSP**
- ⇒ **26011 verschiedene CSP-Policies**
- ⇒ **94,7% leicht umgehbar**

CSP Is Dead, Long Live CSP!

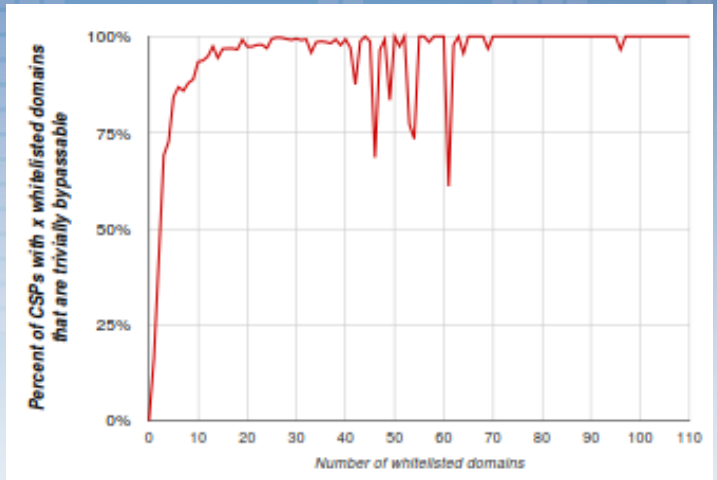
On the Insecurity of Whitelists and the Future of Content Security Policy (2)

[Sebastian Lekies (Vortragender) u.A.]

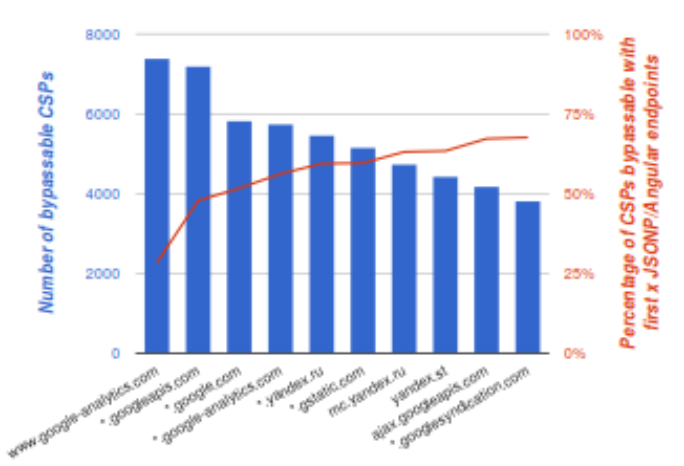
German OWASP Day 2016 - Darmstadt - 2016

OWASP Open Web Application Security Project

- **Übliche Fehler:**
  - **Unsafe-inline in script-src**
  - **Wildcards in script-src**
  - **Fehlende object-src- oder default-src-Direktive**



- **CSP umgehen:**
  - **Unsichere Endpunkte in Whitelists (z.B. JASONP oder AngularJS)**



Quelle: https://research.google.com/pubs/archive/45542.pdf

CSP Is Dead, Long Live CSP!

On the Insecurity of Whitelists and the Future of Content Security Policy (3)

[Sebastian Lekies (Vortragender) u.A.]

OWASP
Open Web Application Security Project

Wie geht das besser?

Erweiterung in CSP3 ⇒ statt Domain-Whitelists “nonce” und “strict-dynamic” verwenden (s.a. https://csp.withgoogle.com)
Nonce: kryptographische Einmal-Token zur Absicherung der Skripte

Strict-dynamic: nonce wird an dynamisch generierte Skripte vererbt (aber nur document.createElement)

```
<script nonce="r4nd0m">
  var s = document.createElement("script");
  s.src = "//example.com/bar.js";
  document.body.appendChild(s);
</script>
```

```
<script nonce="r4nd0m">
  var s = "<script ";
  s += "src=//example.com/bar.js></script>";
  document.write(s);
</script>
```

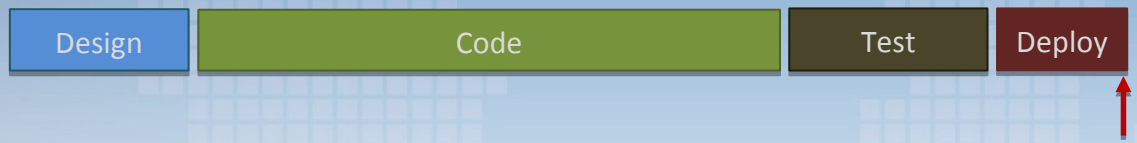
Sicherheit agil Testen (1)
[Matthias Rohr]

German OWASP Day 2016 - Darmstadt - 2016

OWASP Open Web Application Security Project

Vergleich von Vorgehensmodellen zur Software-Entwicklung

1. Waterfall

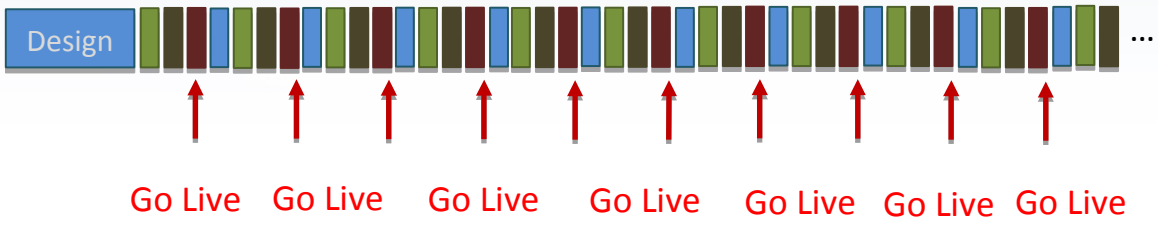


e.g. 1-2 releases a year

2. Agile (e.g. SCRUM) without Continuous Deployment

2-4 weeks per Sprint
e.g. 2-4 releases each YEAR

3. Agile with Continuous Deployment (DevOps)



2-4 weeks per Sprint
e.g. 1-2 releases each DAY

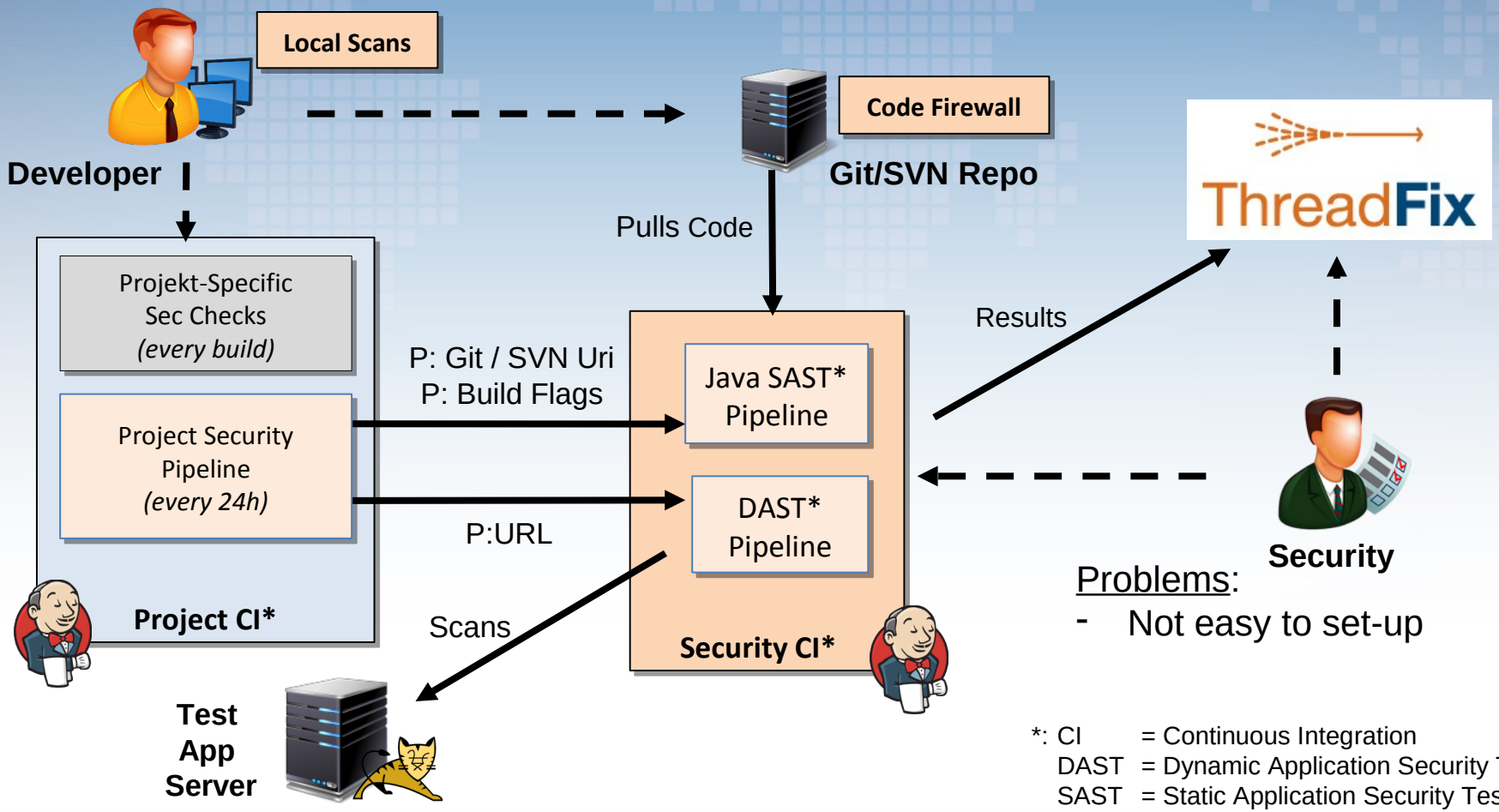
Based on a figure of Dephix

Sicherheit agil Testen (2)
[Matthias Rohr]

German OWASP Day 2016 - Darmstadt - 2016

OWASP Open Web Application Security Project

Integration in den Buid, z.B.: AppSec Scan Factory



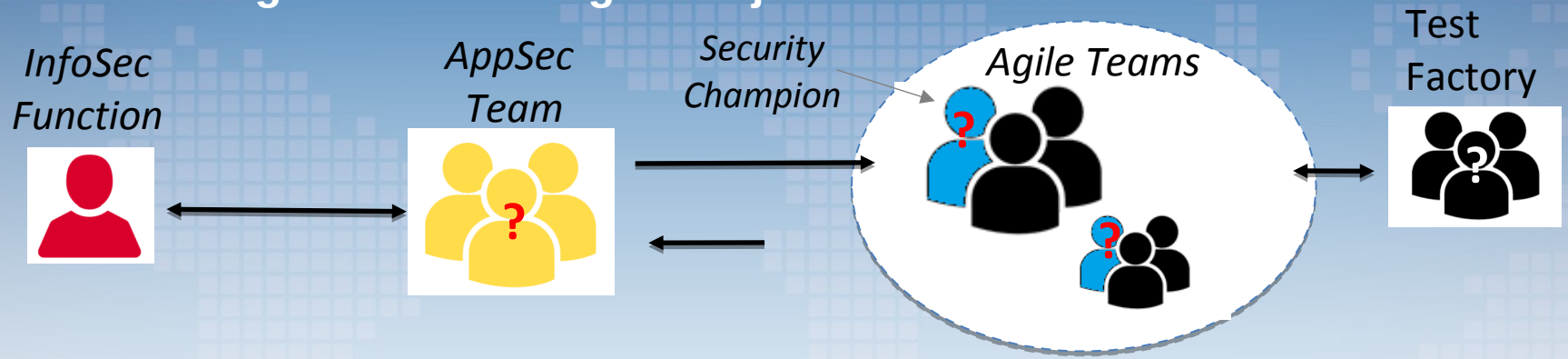
*: CI = Continuous Integration
DAST = Dynamic Application Security Testing
SAST = Static Application Security Testing

Sicherheit agil Testen (3)
[Matthias Rohr]

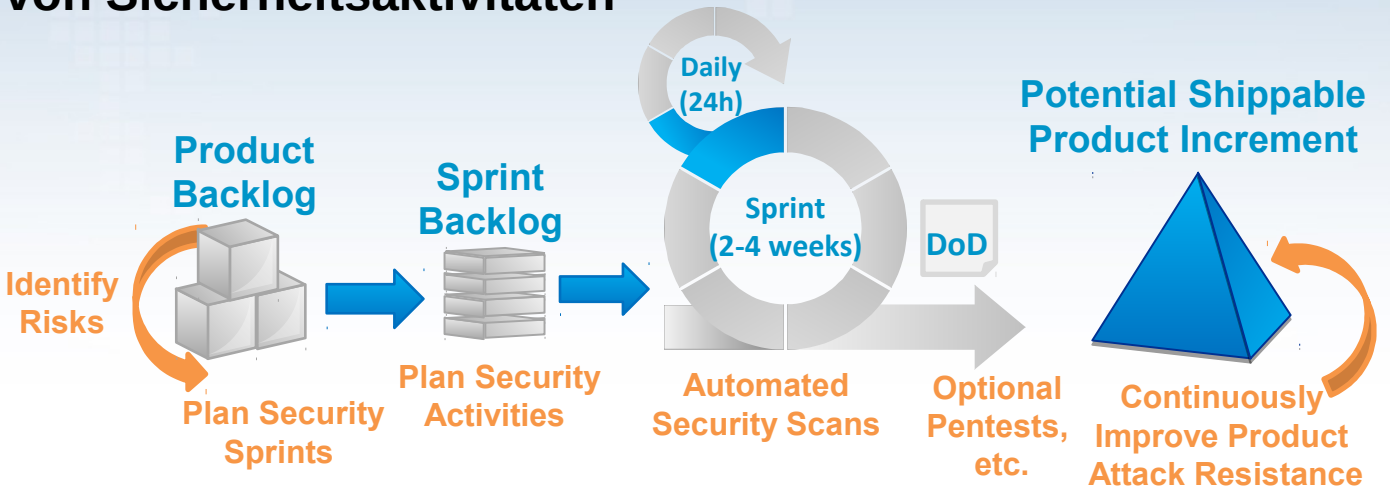
German OWASP Day 2016 - Darmstadt - 2016

OWASP Open Web Application Security Project

Sicherheitsorganisation für agile Projekte



Planung von Sicherheitsaktivitäten



Weitere Informationen: [Sicherheit im Software-Entwicklungsprozess (Jax 2015) [externer Link]]

DROWN (oder warum TLS-Konfiguration schwer ist)
[Sebastian Schinzel (und weitere „Bekannte“)]

OWASP
Open Web Application Security Project

- **beschreibt „Entdeckung“ von DROWN**
- **Ursachen:**
  - Bleichebacher Angriff (1998)
  - Server nutzen den selben Key
  - Server unterstützen Export Cipher
  - Implemetierungsfehler in openssl
  - neue Protokoll-Schwachstelle
- **Test: 15 Verbindungen mit 1920 Verschlüsselungen**

Lightning Talks (1)

OWASP
Open Web Application Security Project

- **What's new in OWASP Juice Shop?**
  https://www.owasp.org/images/5/5d/GOD16-Juice.pdf
  [Björn Kimminich]

  Webanwendung mit absichtlich eingebauten Schwachstellen (über 30), für Schulungszwecke

  Neu:
  - **OWASP-Projekt:** https://www.owasp.org/index.php/OWASP_Juice_Shop_Project
  - **mehrsprachig (u.a. D, E, F)**
  - **Cloud-Unterstützung (Heroku), VirtualBox VM**
  - **SSO-Login via Google-Account (OAUTH 2.0)**
  - **Meldungen, wenn ein 'Challenge' gelöst wurde**
  - **Sichern der erreichten 'Challenges' (Continue-Codes)**
  - **Jetzt 30+7 'Challenges'**
  - **Dokumentation:** https://www.gitbook.com/book/bkimminich/pwning-owasp-juice-shop/details

Lightning Talks (2)

OWASP
Open Web Application Security Project

- **TLS-Attacker** (Systematic Fuzzing and Testing of TLS Libraries)
  https://www.owasp.org/images/b/b4/GOD16-Jurai.pdf
  [Juraj Somorovsky]

  Flexibles Framework-Tool zum Testen von Schwachstellen in SSL/TLS-Bibliotheken

  - **Enthält eine XML-Beschreibungssprache mit der tief im TLS-Protokoll gezielt Änderungen vorgenommen werden können (z.B. Werte, Parameter, Protokollabfolge), Fuzzing**
  - **Damit wurden mehrere Schwachstellen in TLS-Libraries gefunden:**
    Padding oracle attack (CVE-2016-2107, CVE-2015-7824), Bleichenbacher attack, Missing length checks, Out-of-bound reads / writes
  - **Link:** https://github.com/RUB-NDS/TLS-Attacker