OWASP Top 10 fuer Entwickler

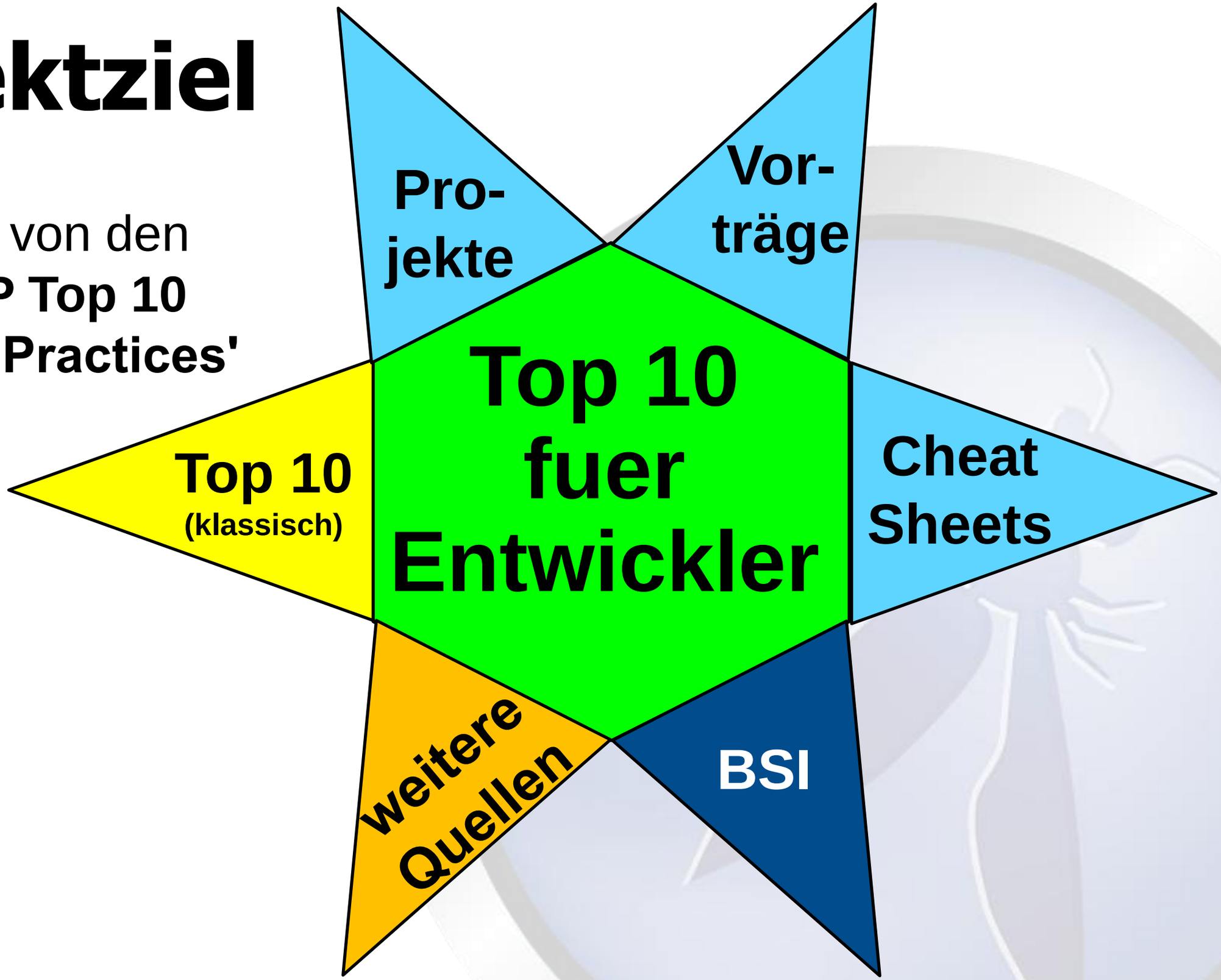
(Top 10 Developer Edition in German)

Torsten Gigler

torsten *punkt* gigler *bei* owasp *punkt* org

Projektziel

Brücke von den
OWASP Top 10
zu **‚Good Practices'**



Projektsteckbrief

- **Speziell auf Entwickler zugeschnittene Version der OWASP Top 10**
- **Brücke zwischen Theorie und Praxis**
- **Code-Beispiele in verschiedenen Programmiersprachen (Start mit Java)**
- **Weitere Informationen für Entwickler**
- **Wiki (Internationalisierung des englischen Top-10-Wikis)**
- **Projektstart: 25.3.2013**
- **Kern-Team: Achim, Ralf, Thomas, Torsten**
- **Aktive Projektunterstützer: bisher 7**
- **Mailing Liste: 14**

Beispielseite: Injection (1)

https://www.owasp.org/index.php/Germany/Projekte/Top_10_fuer_Entwickler/A1_Injection

← Top_10_fuer_Entwickler/Risiken

Einleitung
Die Top-10-Risiken

Top_10_fuer_Entwickler/A2_Cross-Site Scripting (XSS) →

Seite in Bearbeitung (BAUSTELLE!!) + Test für Top 10-Style 2013

A1 Injection (Anfragen an den Interpreter manipulieren)

Bedrohungs-quelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
?	Ausnutzbarkeit EINFACH	Verbreitung HÄUFIG	Auffindbarkeit DURCHSCHNITTLICH	Auswirkung SCHWERWIEGEND	?
Jeder, der Daten, die nicht ausreichend geprüft werden, an das System übermitteln kann: externe und interne Nutzer sowie Administratoren.	Der Angreifer sendet einfache text-basierte Angriffe, die die Syntax des Zielinterpreters missbrauchen. Fast jede Datenquelle kann einen Injection-Vektor darstellen, einschließlich interner Quellen.	Injection-Schwachstellen tauchen auf, wenn eine Anwendung nicht vertrauens-würdige Daten an einen Interpreter weiterleitet. Injection Schwachstellen sind weit verbreitet, besonders in altem Code; sie finden sich in SQL-, LDAP- und XPath-Anfragen, Systembefehlen, Programm-parametern usw. Injection-Schwachstellen lassen sich durch Code-Prüfungen einfach entdecken, schwieriger durch externe Tests. Scanner und Fuzzer können hier unterstützen.		Injection kann zu Datenverlust oder -verfälschung, Fehlen von Zurechenbarkeit oder Zugangssperre führen. Unter Umständen kann es zu einer vollständigen Systemübernahme kommen.	Der Wert betroffener Daten für das Unternehmen sowie die Laufzeitumgebung des Interpreters sind zu berücksichtigen. Daten können entwendet, verändert, gelöscht werden. Kann Image-Schaden enstehen?

Mögliche Angriffsszenarien

Die Anwendung nutzt ungeprüfte Eingabedaten bei der Konstruktion der **verwundbaren** SQL-Abfrage:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") +"'";
```

Durch das alleinige Vertrauen auf Frameworks kann die Abfrage **verwundbar** bleiben (z.B. Hibernate Query Language (HQL)):

```
Query unsafeHQLQuery = session.createQuery("from accounts where
custID='"+request.getParameter("id")+"'");
```

Der Angreifer verändert den 'id'-Parameter im Browser und übermittelt: `' or '1'='1`. Hierdurch wird die Logik der Anfrage so verändert, dass alle Datensätze der Tabelle accounts ohne Einschränkung auf den Kunden zurückgegeben werden.

```
http://example.com/app/accountView?id=' or '1'='1
```

Im schlimmsten Fall nutzt der Angreifer die Schwachstelle, um spezielle Funktionalitäten der Datenbank zu nutzen, die ihm ermöglichen, die Datenbank und möglicherweise den Server der Datenbank zu übernehmen.

Wie kann ich 'Injection' verhindern?

Das Verhindern von Injection erfordert das konsequente Trennen von Eingabedaten und Befehlen.

1. Der bevorzugte Ansatz ist die Nutzung einer sicheren API, die den Aufruf von Interpretern vermeidet oder eine typ-gebundene Schnittstelle bereitstellt. Seien Sie vorsichtig bei APIs, z. B. Stored Procedures, die trotz Parametrisierung anfällig für Injection sein können.
2. Wenn eine typsichere API nicht verfügbar ist, sollten Sie Metazeichen unter Berücksichtigung der jeweiligen Syntax sorgfältig entschärfen. OWASP's ESAPI stellt hierfür spezielle Routinen bereit.
3. Auch die Eingabeprüfung gegen Positivlisten nach Kanonisierung wird empfohlen, ist aber kein vollständiger Schutz, da viele Anwendungen Metazeichen in den Eingaben erfordern. OWASP's ESAPI stellt Routinen zur Eingabeprüfung gegen Positivlisten bereit.

Beispielseite: Injection (2)

https://www.owasp.org/index.php/Germany/Projekte/Top_10_fuer_Entwickler/A1_Injection

JAVA | dotNET | PHP | Test

Verteidigungs-Option 1 gegen 'Injection':

Prepared Statements (Parameterized Queries) [nur für SQL]

Java - Standard

```
String custname = request.getParameter("customerName"); // This should REALLY be
validated too
// perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

Java - Hibernate

```
String userSuppliedParameter = request.getParameter("id"); // This should REALLY
be validated too
// perform input validation to detect attacks
Query safeHQLQuery = session.createQuery("from Inventory where
productID=:productid");
safeHQLQuery.setParameter("productid", userSuppliedParameter);
```

vgl OWASP Query_Parameterization Cheat Sheet

Verteidigungs-Option 2 gegen 'Injection':

Benutzereingaben sorgfältig prüfen und entschärfen (Escaping All User Supplied Input) [bisher nur für SQL]

Die Metazeichen sind je Backend-Typ und meist auch je Backend-Hersteller unterschiedlich (z.B. Datenbank-Hersteller), vgl OWASP's ESAPI.

```
Codec ORACLE_CODEC = new OracleCodec(); //added
String query = "SELECT user_id FROM user_data WHERE user_name = '" +
   ESAPI.encoder().encodeForSQL( ORACLE_CODEC, //added
   req.getParameter("userID") ) + "' and user_password = '" +
   ESAPI.encoder().encodeForSQL( ORACLE_CODEC, //added
   req.getParameter("pwd") ) +"'";
```

Verteidigungs-Option 3 gegen 'Injection':

Benutzereingaben sorgfältig mittels Positivlisten prüfen (White List Input Validation)

Diese Technik sollte bei SQL nur als Zusatzmaßnahme, oder als Notlösung für alte Software in Betracht gezogen werden, für die anderen Typen ist sie die einzige, bekannte Maßnahme. Die Benutzereingaben sind zunachst zu normalisieren (= kanonisieren). APIs, wie OWASP's ESAPI erledigen dies automatisch. Beschränken Sie bei den Regeln für die Postivlisten die erlaubten Zeichen, ggf. die erlaubte Zeichenfolge und den gültigen Wertebereich bzw. die Länge der erwarteten Eingabe. Seien Sie besonders vorsichtig, wenn Sie Zeichen erlauben möchten, die das Backend als Metazeichen benutzt, z.B. ' und " (vgl Potenziell gefährliche Zeichen für Interpreter). Wandeln Sie diese jeweils in ungefährliche Zeichen um, z.B. mittels Kodierung vor der weiteren Verarbeitung (Encoding, z.B. in ' und ").

```
//performing input validation
ESAPI.validator().getValidInput
...
String query = "SELECT user_id FROM user_data WHERE user_name = '" + ...
...
```

Referenzen

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Injection Flaws Article
- ESAPI Encoder API
- ESAPI Input Validation API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- OWASP Code Review Guide: Chapter on SQL Injection
- OWASP Code Review Guide: Command Injection
- OWASP Appsec: Episode 2 - Injection Attacks (Video)

Andere

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- BSI: IT-Grundschutz Baustein Webanwendungen: 'Hilfsmittel' (Potenziell gefährliche Zeichen für Interpreter)

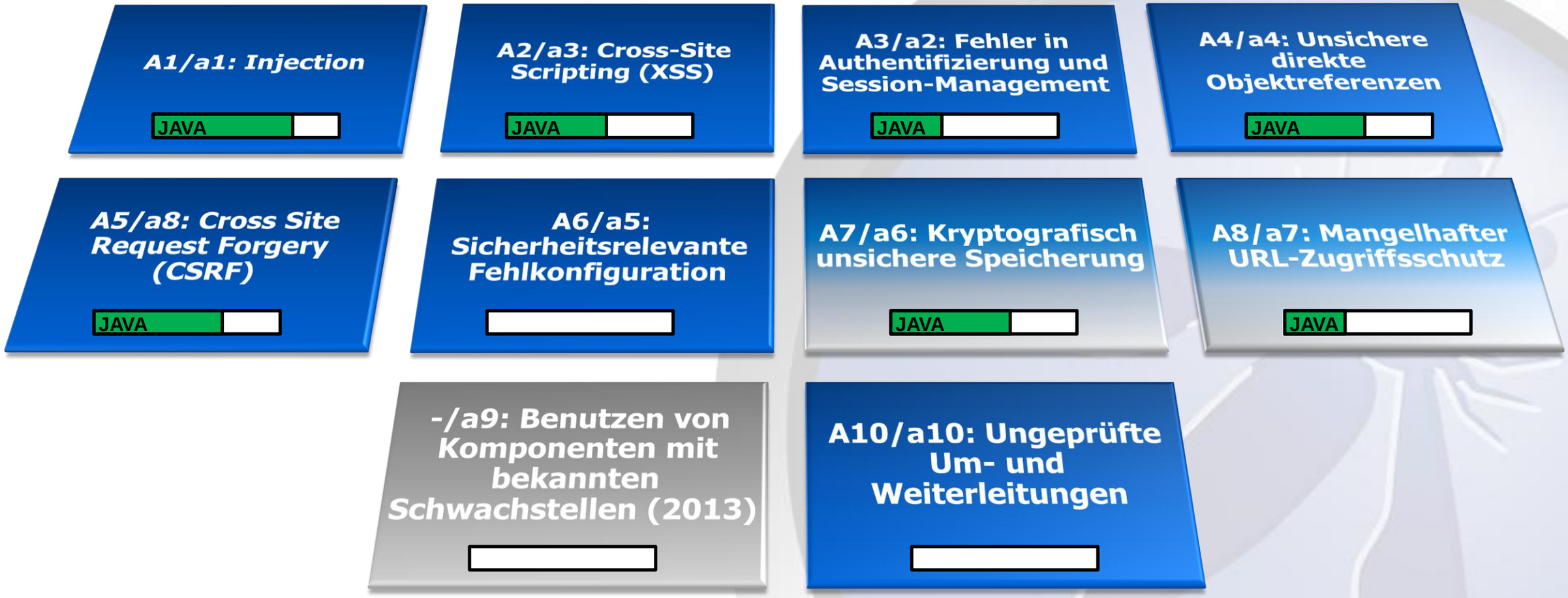
← Top_10_fuer_Entwickler/Risiken

Einleitung
Die Top-10-Risiken

Top_10_fuer_Entwickler/A2_Cross-Site Scripting (XSS) →

OWASP Top 10 fuer Entwickler

(Stand: 15. Mai 2013)



A2/a3 Top 10-2010/top 10-2013 rc1

JAVA Grober Fertigstellungsgrad (α-Release)

Anmerkung: Risiken, die sich in der im Review befindlichen Version der OWASP Top 10 - 2013 RC1 nicht ändern, sind blau und *Änderungen* sind *grau* unterlegt.

Ausblick

Open SAMM

Input vom BSI

alle Top 10

Top 10 fuer Entwickler

Top 10 2013 Deutsch (klassisch)

weitere Tipps

Top 10 Developer Edition (weitere Sprachen)

Review

weitere Programmier-sprachen

OWASP Top 10 fuer Entwickler

Wir freuen uns auf
Deine Ideen und
Deine Mitarbeit

☺

https://lists.owasp.org/mailman/listinfo/owasp_top_10_fuer_entwickler