

Compass Security

Bedrohungen Web-Applikationen

17. März 2006

Dokumentname: compass_security_bedrohungen_web_applikationen_v2.5.doc
Version: V 2.5
Autor(en): Walter Sprenger, Compass Security AG
Lieferungsdatum: 17. März 2006
Klassifikation: ÖFFENTLICH

Inhaltsverzeichnis

| | |
|---|----------|
| 1 ANGRIFFE AUF WEB APPLIKATIONEN | 3 |
| 1.1 Einleitung | 3 |
| 1.2 Compass Security | 3 |
| 1.3 Zielpublikum dieses Dokuments | 3 |
| 1.4 Bedeutung von Fachbegriffen | 4 |
| 1.4.1 Entry Server | 4 |
| 1.4.2 Presentation Server | 4 |
| 1.4.3 Business Server | 4 |
| 1.4.4 Phishing | 4 |
| 1.5 Web Application Security Lab | 5 |
| | |
| 2 CHECKLISTE..... | 6 |
| 2.1 Authentisierung | 6 |
| 2.2 Session Handling | 7 |
| 2.3 Autorisierung | 8 |
| 2.4 Input Validation | 8 |
| 2.5 Konfiguration | 10 |
| 2.6 Logging | 11 |
| 2.7 Error Handling | 12 |
| 2.8 Verschlüsselung | 12 |
| 2.9 Sonstiges | 13 |

1 Angriffe auf Web Applikationen

1.1 Einleitung

Das OWASP¹ (Open Web Application Security Project) hat gute Richtlinien erarbeitet für die sichere Entwicklung von Web Applikationen. Dieses Dokument baut auf diesen Empfehlungen auf und ist durch die Erfahrungswerte von Compass Security ergänzt worden. In Kapitel 2 werden Security Bedrohungen und dazugehörige Gegenmassnahmen aufgezeigt. Dieses Kapitel ist das Herz dieses Dokumentes und kann als Raster für ein Security Konzept verwendet werden.

1.2 Compass Security

Compass Security Network Computing AG (CSNC) ist eine Schweizer Firma, die im Februar 1999 durch Walter Sprenger und Ivan Buetler gegründet wurde. Compass Security hat sich von seit Beginn an mit "Security Assessments" beschäftigt und bietet dieses Know-How in Form von Penetration Tests und Security Reviews an. Seit 2004 werden auch zunehmend forensische Untersuchungen durchgeführt. Das Know-How wird in der Zusammenarbeit mit der FH Rapperswil und der HSW (Hochschule für Wirtschaft) im Fach „Informatik Sicherheit“ laufend erweitert.

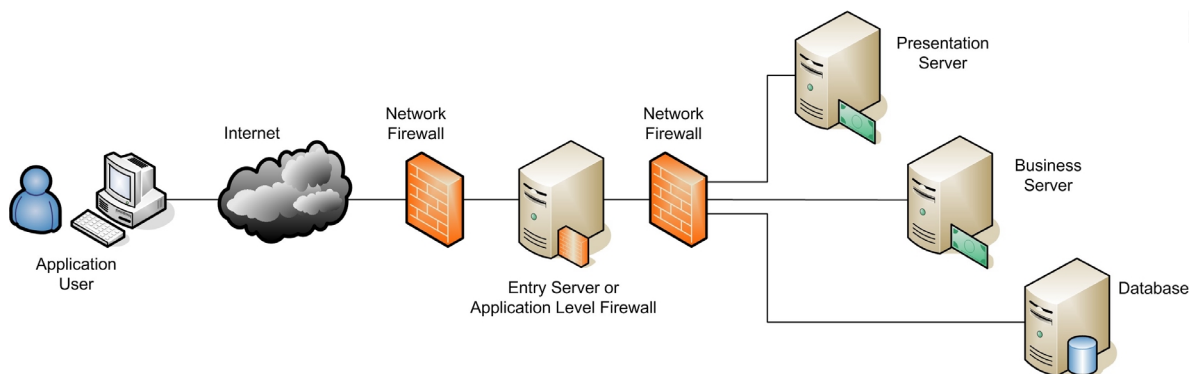
1.3 Zielpublikum dieses Dokuments

Dieses Dokument ist in einer technisch gehaltenen Sprache verfasst und setzt Kenntnisse über die Protokolle wie HTTP, HTTPS unter Berücksichtigung von Sessions, Cookies, Cookie Attribute etc. voraus.

Das Zielpublikum dieser Checkliste sind Security Verantwortliche und Web Entwickler, damit diese eine Hilfestellung bei der Implementation von sicheren Webanwendungen haben.

¹ OWASP: <http://www.owasp.org/>

1.4 Bedeutung von Fachbegriffen



1.4.1 Entry Server

Ein Entry Server (oder ein Application Level Firewall) ist eine der Applikation vorgeschaltete Sicherheits-Komponente, welche eine Prüfung der Requests erlaubt. Beispiele für Entry Server sind: Reverse Proxy mit Authentisierungsmodul wie mod_but², Visonys Airlock, IBM WebSeal, Tetrade SES, keyon TESS, AdNovum nevisProxy.

1.4.2 Presentation Server

Gemäss J2EE Standard ist der „Presentation Server“ eine Komponente, welche lediglich die Präsentation der Web Applikation durchführt, so wie dies in einer 3-Tier Architektur vorgesehen ist.

1.4.3 Business Server

Gemäss J2EE Standard ist der „Business Server“ eine Komponente, welche die Business Logik implementiert.

1.4.4 Phishing

Compass Security ist der Meinung, dass technisch gesehen lediglich ein Client Zertifikat als Schutz vor Phishing mit „Man in the Middle“ implementiert werden muss.

Challenge/Response Verfahren bei der Authentisierung erschweren Phishing Attacken, bei denen der Angreifer Teile der originalen Webseite durch gefälschte Seiten auf einem Fremdsystem anbietet.

² http://www.but.ch/mod_but/

1.5 Web Application Security Lab

Die Ausbildung der Entwickler und Security Verantwortlichen von Webapplikationen ist sehr wichtig bei der Erstellung von sicheren Anwendungen. Der Compass „Application Security Lab“ Kurs geht in praktischer Form auf die Sicherheit von Webapplikationen ein. Während drei Tagen tauchen Sie in die Welt von Browser, Cookie, Sessions, Verschlüsselung und Entry Server ein. Ist es zum Beispiel möglich, dass ein Kunde des OnlineBanking auf Daten eines anderen Kunden zugreifen kann? Ist die Autorisierung, Datenisolierung und Session Handling sauber implementiert? Compass Security hat die grössten Fehler von Webapplikationen in einer selbst entwickelten Webapplikation eingebaut - die es während der Dauer des Kurses zu analysieren gilt. Viele praktische Übungen! Informationen dazu finden Sie unter folgendem Link:

<http://www.csnc.ch/static/services/training/index.html>

Die Kurse sind über ISACA Schweiz³ erhältlich, oder direkt als Firmenkurs vor Ort.

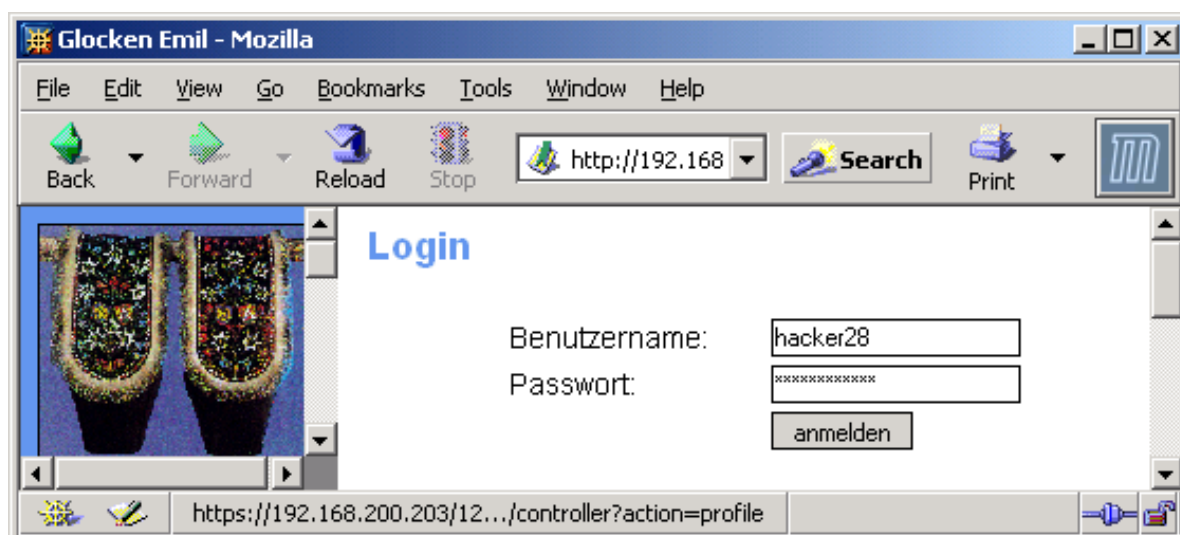


Abbildung: Lern Anwendung von Compass Security: SQL Injection

³ <http://www.isaca.ch/>

2 Checkliste

Die folgende Checkliste zeigt Bedrohungen und Gegenmassnahmen auf, welche bei HTML basierten Anwendungen zu berücksichtigen sind.

2.1 Authentisierung

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|-----------------|---|--|
| 1 | Authentisierung | Phishing mit kopierter Login Seite | <ul style="list-style-type: none"> • Challenge-Response Verfahren oder SSL Client Zertifikat |
| 2 | Authentisierung | Phishing mit Man in the Middle (Umleitung des Verkehrs über den Angreifer) | <ul style="list-style-type: none"> • SSL Client Zertifikat • Benutzer informieren |
| 3 | Authentisierung | Phishing durch Ausnutzen von Redirection Schwächen (URL Redirection Attack) | <ul style="list-style-type: none"> • Redirect URL prüfen • nur auf die eigene Domain weiterleiten |
| 4 | Authentisierung | Benutzernamen Enumeration | <ul style="list-style-type: none"> • Fehlermeldung „Benutzername oder Passwort falsch“ anstatt „Benutzer ungültig“ oder „Passwort falsch“ |
| 5 | Authentisierung | Benutzer aussperren | <ul style="list-style-type: none"> • Zeitverzögerung einbauen |
| 6 | Authentisierung | Schwache Passworte (Default Passworte, Policy beim Ändern des Passwortes) | <ul style="list-style-type: none"> • Starkes Passwort beim Passwort Wechsel erzwingen |
| 7 | Authentisierung | Schwächen in der Selbstregistrierung von Benutzern ausnutzen | <ul style="list-style-type: none"> • Nur Gastrechte zuteilen • Autorisierung prüfen |
| 8 | Authentisierung | Umgehung der Authentisierung (z.B. Login mit Default oder Test Benutzername/ Passwort) | <ul style="list-style-type: none"> • Default und Testbenutzer löschen. • Entry Server muss prüfen, ob Benutzer authentisiert ist. |
| 9 | Authentisierung | Nach dem Logout den Back-Button des Browsers drücken bis die Meldung erscheint „Resend POSTDATA“. Durch klicken von OK ist der Angreifer wieder eingeloggt. | <ul style="list-style-type: none"> • Das Servlet, welches die Authentisierung durchführt darf die Antwortseite nicht selber ausliefern, sondern muss mittels eines Redirects auf die Antwortseite verweisen. Dadurch „vergisst“ der Browser die POST Daten. |

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|-----------------|--|--|
| 10 | Authentisierung | Dem Schutzbedarf der Daten nicht angepasste Authentisierungsvariante. Z.B. nur Benutzername und Passwort für sehr schützenswerte Daten. | <ul style="list-style-type: none"> Ermitteln des Schutzbedarfs und rechtlicher Rahmenbedingungen Einsatz von adäquaten Authentisierungsvarianten wie z.B. Einmal-Passworte, Challenge-Response oder SSL Client Zertifikate |
| 11 | Authentisierung | Passwort kann mittels „Passwort vergessen“ Funktionalität zugeschickt werden. Angreifer kann sich ein fremdes Passwort schicken lassen oder Passwort eines anderen Benutzers erlauschen. | <ul style="list-style-type: none"> Sicherstellen, dass nur der Eigentümer das Passwort abrufen kann. Evtl. mit Secure Answer/Question schützen. Auf anderem Weg zukommen lassen (Post/SMS) |
| 12 | Authentisierung | Der Browser merkt sich Benutzername oder Passwort. | <p>Auf dem Formular das Feature "autocomplete" deaktivieren:</p> <ul style="list-style-type: none"> "autocomplete=off" |

2.2 Session Handling

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|------------------|---|--|
| 13 | Session Handling | Logout beendet nicht alle Sessions (Mit Back Button gelangt nachfolgender Benutzer wieder in authentifizierte Bereiche) | <ul style="list-style-type: none"> Logout Request mit POST absetzen. Server-seitig alle Sessions beenden und Cookies beim Benutzer löschen. |
| 14 | Session Handling | Session Prediction | <ul style="list-style-type: none"> Generieren von eindeutigen und zufälligen, langen Session Identifikatoren |
| 15 | Session Handling | Session Fixation | <ul style="list-style-type: none"> Nach einem erfolgreichen Login muss eine neue Session generiert werden falls zuvor schon eine erzeugt wurde Session Identifikator darf nicht vom Client übernommen werden |
| 16 | Session Handling | Session Hijacking | <ul style="list-style-type: none"> Verschlüsseln des Verkehrs Zufällige Session Identifikatoren Cookie Settings (secure Flag, HTTPOnly Flag) Verhindern von CSS (Cross Site Scripting) |

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|------------------|---|---|
| 17 | Session Handling | Session des Entry Servers, des Presentation Servers und des Business Servers müssen eindeutig sein. Es darf keine Verwechslung geben da sonst ein Benutzer die Daten eines anderen Benutzers sieht. | <ul style="list-style-type: none"> • Sauberes Session Handling • Benutzerkontext muss auf jedem System eindeutig identifizierbar sein |

2.3 Autorisierung

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|---------------|--|---|
| 18 | Autorisierung | Umgehung Autorisationprüfung auf Funktionen (mehr Rechte erlangen) | <ul style="list-style-type: none"> • Berechtigungen für Funktionen muss bei jedem Request server-seitig geprüft werden |
| 19 | Autorisierung | Umgehung Autorisationsprüfung auf Daten | <ul style="list-style-type: none"> • Berechtigungen für Daten muss bei jedem Request server-seitig geprüft werden |

2.4 Input Validation

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|--------------------|----------------------------------|--|
| 20 | Eingabevalidierung | SQL Injection | <ul style="list-style-type: none"> • Technischer Datenbank User sollte minimale Rechte haben • Prepared Statements verwenden • Eingabeprüfung |
| 21 | Eingabevalidierung | OS Command Injection | <ul style="list-style-type: none"> • Validieren der Eingaben • Keine direkten Betriebssystemaufrufe zulassen |
| 22 | Eingabevalidierung | LDAP Injection | <ul style="list-style-type: none"> • Validieren der Eingaben • Gefährliche Zeichen entfernen (z.B. Strichpunkt ;) |
| 23 | Eingabevalidierung | Cross Site Scripting (CSS / XSS) | <ul style="list-style-type: none"> • Validieren der Eingaben • Bei der Anzeige der Daten müssen diese HTML encodiert werden |

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|---|--|---|
| 24 | Eingabevalidierung | Missbrauch von Mail-Funktionalität | <ul style="list-style-type: none"> • Empfänger müssen in der Applikation konfiguriert werden und dürfen nicht in den HTML Seiten definiert sein • SMTP Headers entfernen falls Mail an Kunde verschickt wird |
| 25 | Eingabevalidierung | Missbrauch von File Uploads (Upload in Applikationsverzeichnisse oder Modifikation von Webseiten, Upload von Viren, Upload falsch formatierter Dateien) | <ul style="list-style-type: none"> • Grösse beschränken • Dateitypen einschränken • Datei nicht auf Filesystem schreiben sondern direkt im Speicher verarbeiten • Wenn Datei auf Filesystem gespeichert wird: Einmalige Namen verwenden, Pfadinformationen entfernen und in Verzeichnis schreiben, das vom Internet her nicht erreichbar ist • Syntax der erwarteten Datei prüfen • Dateien auf Viren prüfen falls diese auf anderen Systemen verarbeitet werden (z.B. Excel, Word Dateien) |
| 26 | Eingabevalidierung | Codierungsschwächen ausnützen (z.B. doppelt codierte URLs werden vom Entry Server durchgelassen aber vom Backend Server interpretiert -> Directory Traversal) | <ul style="list-style-type: none"> • Request mit URL Encoding auf dem Entry Server einmal decodieren und dann Sicherheitsfunktionen anwenden. Falls doppeltes Encoding auftritt URL verwerfen |
| 27 | Eingabevalidierung / Konfiguration | Modifikation von Webseiten (Web-Authoring Funktionalität vorhanden oder Freitext Felder in z.B. Mitteilungsbrett) | <ul style="list-style-type: none"> • Validieren der Eingaben • Bei der Anzeige der Daten müssen diese HTML encodiert werden |
| 28 | Eingabevalidierung / Protokollvalidierung | HTTP Request Smuggling: Mittels modifizierter HTTP Requests wird ein Request in einen anderen verpackt (smuggling). Dadurch kann Zugriff auf andere Ressourcen erlangt werden oder der Cache modifiziert werden, so dass nachfolgende Benutzer falsche Seiten sehen. | <ul style="list-style-type: none"> • Strikte Validierung des HTTP Protokolls. Muss vom HTTP Server Hersteller implementiert/gefixt werden. • Entry Server einsetzen welche diese Art von Requests filtert. |

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|--|--|---|
| 29 | Eingabevalidierung / Protokollvalidierung | HTTP Response Splitting: Mittels CR/LF Zeichen wird der Web Server dazu veranlasst, die Antwort aufzusplitten. Dadurch kann ein Angreifer z.B. selbst generierte Header in die Antwort einfügen, welche z.B. auf einem Entry Server eine freischaltung weiterer URL's bewirkt. | <ul style="list-style-type: none"> • Strikte Validierung des HTTP Protokolls. Muss vom HTTP Server Hersteller implementiert/gefixt werden. • Web Applikation darf CR/LF nicht in die Antworten einbetten. |

2.5 Konfiguration

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|---------------|---|---|
| 30 | Konfiguration | Cross Site Tracing (XST) | <ul style="list-style-type: none"> • HTTP Methode TRACE und OPTIONS auf dem Webserver deaktivieren |
| 31 | Konfiguration | System Compromise (Zugriff auf einen Server: Authentisierung/EntryServer/Pre sentation/Business/Database) | <ul style="list-style-type: none"> • Server Software laufend auf dem aktuellsten Stand halten • Berechtigungen so restriktiv wie möglich (Prozess Benutzer darf nur Leserechte auf die Prozess Binaries und Konfigurationsdateien haben) • System mit Firewall voneinander trennen • Evtl. Chroot verwenden |
| 32 | Konfiguration | Administrationsinterfaces vom Internet her erreichbar | <ul style="list-style-type: none"> • Administrations-Interfaces dürfen nicht vom Internet erreicht werden können. Separaten Webserver auf anderem Port oder mindestens Einschränkung basierend auf IP- Adresse und starke Authentisierung |
| 33 | Konfiguration | Fehler und Funktionalität in Beispiel Seiten ausnützen | <ul style="list-style-type: none"> • Test, Beispiel und Demo Seiten entfernen |
| 34 | Konfiguration | Verzeichnisinhalte werden angezeigt (Directory Browsing) | <ul style="list-style-type: none"> • Directory Browsing auf dem Web Server und Entry Server deaktivieren |
| 35 | Konfiguration | Zugriff auf Dateien ausserhalb Web-Applikation (z.B. Zugriff auf Konfiguration oder Schlüsselmaterial aufgrund zu wenig restriktiver Berechtigungen) | <ul style="list-style-type: none"> • Restriktive Konfiguration von Webserver und Servlet-Engines. Zugriff nur auf benötigte Dateien/ Verzeichnisse zulassen. |

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|---------------|--|---|
| 36 | Konfiguration | Zugriff auf Demo/Test-Umgebung und dadurch Zugriff auf produktive Daten | <ul style="list-style-type: none"> • Test-Umgebungen dürfen nur von bestimmten IP-Adressen erreicht werden können • In den Demo und Testumgebungen dürfen keine produktiven Daten vorkommen |
| 37 | Konfiguration | Zugriff auf Backup-Files auf dem produktiven Server | <ul style="list-style-type: none"> • Alle Dateien, welche für den Betrieb nicht direkt benötigt werden sollen entfernt werden |
| 38 | Konfiguration | Zugriff auf unverschlüsselte Passworte in Konfigurationsdateien | <ul style="list-style-type: none"> • Alle Passwort, welche auf den Systemen gespeichert werden, müssen verschlüsselt sein |
| 39 | Konfiguration | Zugriff auf Standard URL (z.B. /script /admin /backup /WEB-INF /snoop.jsp /test.html) | <ul style="list-style-type: none"> • Entfernen aller Standardverzeichnisse • Schützen der Verzeichnisse, damit nicht vom Internet zugegriffen werden kann • Sensitive Informationen aus den Verzeichnissen entfernen |
| 40 | Konfiguration | Daten respektive HTML Seiten werden im Cache des Browsers gespeichert und können durch einen nachfolgenden Benutzer eingesehen werden. | <ul style="list-style-type: none"> • Header setzen, welche den Browser anweisen die Seiten nicht zu speichern. „Pragma: no-cache“ „Cache-Control: no-cache, no-store“ |

2.6 Logging

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|-----------|--|---|
| 41 | Logging | Sensitive Informationen in Logfiles (z.B. Sessions oder Passwörter in Logfiles) | <ul style="list-style-type: none"> • Keine sensitiven Informationen in Logdateien schreiben • Logdateien schützen, dürfen nie vom Internet erreichbar sein |
| 42 | Logging | Angriffe werden nicht erkannt | <ul style="list-style-type: none"> • Applikationen sollen alle Anfragen und Tätigkeiten in Logdateien protokollieren • Die Logdateien müssen automatisch verarbeitet werden können (Parsen nach kritischen Einträgen) |
| 43 | Logging | Fehlende Logeinträge für forensische Untersuchungen (kein Nachweis was ein Benutzer gemacht hat) | <ul style="list-style-type: none"> • Alle Logdateien müssen miteinander korreliert werden können (vom Entry Server bis zur Datenbank) |

2.7 Error Handling

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|------------------|--|--|
| 44 | Fehlerbehandlung | Fehlerbehandlung zu sprechend (Fehlermeldungen verraten interne Informationen oder geben dem Angreifer Hilfestellungen was auf dem Backend falsch läuft) | <ul style="list-style-type: none"> Eigene Fehlerseiten erstellen und alle Standard-Fehlerseiten ersetzen |
| 45 | Fehlerbehandlung | Manipulationen am URL, an Form Feldern oder Cookie Werten | <ul style="list-style-type: none"> Zugriffskontrolle bei allen Parametern durchführen Wertebereiche prüfen |

2.8 Verschlüsselung

| No | Category | Threat | Countermeasures |
|----|-----------------|---|--|
| 46 | Verschlüsselung | Verkehr belauschbar weil nicht verschlüsselt | <ul style="list-style-type: none"> Jeglichen Verkehr mit SSL verschlüsseln Secure Flag beim Cookie setzen Klare Trennung zwischen Verschlüsselten und unverschlüsselten Bereichen |
| 47 | Verschlüsselung | Ungültiges, abgelaufenes SSL Zertifikat im Einsatz. Der Benutzer kann nicht prüfen, ob er sich auf der richtigen Web Seite befindet. | <ul style="list-style-type: none"> Offiziell beglaubigte Zertifikate einsetzen |
| 48 | Verschlüsselung | Schwache SSL Ciphers werden vom Web Server oder vom Entry Server unterstützt. Angreifer kann schwach verschlüsselte SSL Verbindungen entschlüsseln. | <ul style="list-style-type: none"> Starke SSL Ciphers mit einer Schlüssellänge von mindestens 128 bit einsetzen |

2.9 Sonstiges

| Nr. | Kategorie | Bedrohung | Gegenmassnahme |
|-----|------------------|--|--|
| 49 | Schutz der Daten | Information Disclosure (Versionen, Produkte, Entwicklernamen, Source Code, Konfigurationen sichtbar) | <ul style="list-style-type: none"> • Keine versteckten Informationen in HTML/CSS Seiten • Nur benötigte Seiten/Servlets auf dem Server • Kein Zugriff auf Source Code / Class Files und Konfigurationsverzeichnisse • In allen Produkten Default Fehlermeldungen durch eigene Seiten ersetzen • Deaktivieren von Stack Traces |
| 50 | Funktionalität | Belauschen von sensibler Information welche per Email verschickt wird | <ul style="list-style-type: none"> • Keine sensiblen Informationen in Emails verschicken. (Text z.B. Neue Informationen in ihrem Online Banking Bereich) |