


Diese Leseprobe haben Sie beim
 edv-buchversand.de heruntergeladen.
Das Buch können Sie online in unserem
Shop bestellen.

[Hier zum Shop](#)

Auf einen Blick

1	Über dieses Buch	17
2	Das Ziel erkunden	59
3	Zustandsbasierte Angriffe	105
4	Angriffe auf die Authentifizierung	145
5	Cross-Site-Scripting	227
6	SQL-Injection	345
7	Injection jenseits von SQL	407
8	Dateioperationen: von Directory-Traversal bis Datei-Uploads	453
9	Ein kleiner Exkurs: Pufferüberläufe, Integer-Schwachstellen und Format-Strings	509
10	Angriffe auf die Architektur	527
11	Angriffe auf den Webserver	539

Inhalt

Materialien zum Buch	16
1 Über dieses Buch	17
1.1 Was dieses Buch ist	17
1.1.1 Was dieses Buch nicht ist	17
1.1.2 Wie ist das Buch aufgebaut?	18
1.1.3 Danksagungen	18
1.2 Welche Schwachstellen gibt es?	18
1.2.1 Die OWASP Top 10	20
1.2.2 Weitere Schwachstellen	21
1.2.3 Gliederung des Buchs	21
1.3 Wie findet man Schwachstellen?	23
1.3.1 Manuelle oder automatische Suche?	24
1.4 Einige Tools im Überblick	24
1.4.1 Der OWASP Zed Attack Proxy (ZAP)	24
1.4.2 Browser Exploitation Framework BeEF	29
1.4.3 Mini MySquatOr	30
1.4.4 THC-Hydra	30
1.4.5 Kali Linux	30
1.5 Die Demo-Anwendung	32
1.5.1 Systemvoraussetzungen	33
1.5.2 Die Installation der Demo-Anwendung allgemein	33
1.5.3 Die Installation in einer Linux-VM	35
1.6 OWASP Top 10 Platz 10: Insufficient Logging & Monitoring	53
1.6.1 Auch Nichtstun kann gefährlich sein	54
1.6.2 Wann ist eine Anwendung betroffen?	55
1.6.3 Einige Angriffsszenarien	55
1.6.4 Unzureichendes Logging und Monitoring verhindern	56
1.7 Links	57

2	Das Ziel erkunden	59
2.1	Offensichtliche Informationen sammeln	59
2.1.1	Der Aufbau der Webanwendung	59
2.1.2	Die Webseiten	60
2.1.3	Die Parameter	61
2.2	Nicht offensichtliche Informationen sammeln	62
2.2.1	Nicht verlinkte Verzeichnisse, Dateien	62
2.2.2	Was läuft außer der Anwendung auf dem Server?	62
2.2.3	Die eigene Anwendung aufräumen	63
2.3	Die Untersuchung des Clients	63
2.3.1	Untersuchung statischer Seiten	63
2.3.2	Untersuchung von JavaScript-Clients	65
2.4	Die Demo-Anwendung	69
2.4.1	Die Startseite »/app/index.php«	69
2.4.2	Das Stylesheet »style.css«	78
2.4.3	Wir bleiben auf der Startseite »/app/index.php«	79
2.4.4	Das Backend »/app/backend/index.php«	88
2.4.5	Der Administrationsbereich »admin/index.php«	99
2.5	Links	104
3	Zustandsbasierte Angriffe	105
3.1	Angriffe auf Zustandsinformationen	105
3.1.1	Versteckte Felder	106
3.1.2	URL-Parameter	107
3.1.3	Cookie-Parameter	108
3.2	URL-Jumping	111
3.2.1	Schwachstellen finden	111
3.2.2	Angriffe über URL-Jumping abwehren	112
3.3	Session-Hijacking	114
3.3.1	Schwachstellen finden	115
3.3.2	Session-Hijacking abwehren	116
3.4	Session-Fixation	117
3.4.1	Schwachstellen finden	118
3.4.2	Session-Fixation verhindern	119

3.5	Clickjacking	120
3.5.1	Schwachstellen finden	124
3.5.2	Clickjacking verhindern	124
3.6	Cross-Site-Request-Forgery	126
3.6.1	Schwachstellen suchen	128
3.6.2	CSRF verhindern	128
3.7	Die Demo-Anwendung	129
3.7.1	Angriffe auf Zustandsinformationen	129
3.7.2	URL-Jumping	136
3.7.3	Session-Hijacking	138
3.7.4	Session-Fixation	139
3.7.5	Clickjacking	140
3.7.6	CSRF	140
3.8	Links	143
4	Angriffe auf die Authentifizierung	145
4.1	Einführung: Authentifizierung – einfach, und doch so schwer ...	145
4.1.1	Ein wichtiger Angriff fehlt hier: das Umgehen der Authentifizierung durch SQL-Injection	145
4.1.2	Sehr beliebt: Defaultwerte	146
4.1.3	Authentifizierungsmechanismen	153
4.2	Designfehler	154
4.2.1	Schlechte Passwörter	155
4.2.2	Schwache Passwortregeln erkennen	155
4.2.3	Angriff mit dem digitalen Vorschlaghammer: Brute Force	156
4.2.4	Design-Schwachstellen finden	157
4.2.5	Brute-Force-Angriffe verhindern	160
4.3	Ungeschützt übertragene Zugangsdaten	163
4.3.1	HTTPS unsicher verwendet	164
4.3.2	Schwachstellen suchen	165
4.3.3	Verschlüsseln, aber richtig!	169
4.4	Funktion zum Ändern des Passworts	173
4.4.1	Funktion zur Passwortänderung überflüssig?	174
4.4.2	Unsichere Passwortänderung	174
4.4.3	Schwachstellen in der Passwortänderung finden	174
4.4.4	Angriffe auf die Passwortänderung verhindern	175

4.5	Passwort-Reset	178
4.5.1	Vergessenes Passwort	179
4.5.2	»Vergessenes Passwort«-Funktion testen	181
4.5.3	Eine sichere Lösung für den Passwort-Reset	182
4.6	»Remember me«	182
4.6.1	Persistenter Cookie – persistente Schwachstelle	183
4.6.2	Sichere Variante ist nicht zwingend sicher	184
4.6.3	Schwachstellen in »Remember me« suchen	184
4.7	User Impersonation	184
4.7.1	Designfehler und -schwachstellen	185
4.7.2	Schwachstellen suchen	186
4.8	»Halbe Passwörter«	186
4.8.1	Gute Passwörter rein	187
4.8.2	Schlechte Passwörter raus	187
4.8.3	Passwortprüfung testen	188
4.9	Nicht eindeutige Benutzernamen und doppelte Benutzer	188
4.9.1	NULL oder nichts?	189
4.9.2	Mögliche Angriffe	190
4.9.3	Schwachstellen suchen	191
4.10	Vorhersagbare Benutzernamen	191
4.10.1	Schwachstellen suchen	192
4.11	Vorhersagbare Passwörter	193
4.11.1	Schwachstelle suchen	193
4.11.2	Sichere Startpasswörter	193
4.12	Übertragung der Zugangsdaten »out of band«	194
4.12.1	Schwachstellen suchen	195
4.12.2	Schwachstellen verhindern	195
4.13	Fehler in mehrstufigen Systemen	196
4.13.1	Schwachstellen suchen	197
4.13.2	Falsche Frage	197
4.14	Unsichere Speicherung von Zugangsdaten	198
4.14.1	Exkurs: Hashfunktionen allgemein	199
4.14.2	Hashfunktionen und Passwörter	200
4.14.3	Allgemeine Angriffe auf Hashfunktionen	201
4.14.4	Angriffe auf Passwort-Hashwerte	202
4.14.5	Sichere Passwort-Hashes: Gemütlichkeit schafft Sicherheit	205
4.15	Fail-Open-Mechanismen	206

4.16	Die Demo-Anwendung	206
4.17	Sichere Authentifizierung	212
4.17.1	Verwendung sicherer Zugangsdaten	213
4.17.2	Sichere Verwendung der Zugangsdaten	215
4.17.3	Korrekte Prüfung der Zugangsdaten	216
4.17.4	Preisgabe von Informationen verhindern	218
4.17.5	Überwachung tut not	219
4.17.6	Sorgfalt, Sorgfalt, Sorgfalt!	220
4.17.7	Zwei-Faktor-Authentifizierung im Überblick	220
4.18	Links	221
5	Cross-Site-Scripting	227
5.1	XSS-Schwachstellen im Überblick	227
5.1.1	Reflektiertes XSS	229
5.1.2	Persistentes XSS	229
5.1.3	DOM-basiertes XSS	230
5.2	Schutzmaßnahmen im Browser	232
5.2.1	Die Same-Origin Policy	232
5.2.2	Die Content Security Policy (CSP)	233
5.2.3	XSS-Filter im Browser	237
5.3	Angriffe über XSS	238
5.3.1	Ein paar einfache Beispiele	238
5.3.2	Jetzt wollen wir mal eskalieren	239
5.3.3	Angriffe im lokalen Netz: Portscan über JavaScript	245
5.3.4	Angriffe auf den Benutzerrechner	255
5.3.5	Fazit	263
5.4	Webwürmer	263
5.4.1	Der erste seiner Art: Samy	264
5.4.2	Nach Samy kam Yamanner	270
5.4.3	Ein letztes Beispiel: der Orkut-XSS-Wurm	276
5.4.4	Eine Einordnung	281
5.5	Reflektiertes XSS finden	282
5.5.1	Gefährliche Eingaben	283
5.5.2	Mögliche Schwachstellen finden	283
5.5.3	Mögliche Schwachstellen prüfen	285
5.5.4	Zusammenfassung	288

5.6	Persistentes XSS finden	288
5.6.1	Drei Beispiele	289
5.6.2	Die Suche beginnt	290
5.6.3	Mögliche Schwachstellen prüfen	293
5.7	DOM-basiertes XSS finden	293
5.7.1	Ein reines Clientproblem	293
5.7.2	Möglichkeit 1: Testcode eingeben	293
5.7.3	Möglichkeit 2: Clientcode untersuchen	294
5.8	XSS verhindern	295
5.8.1	Reflektiertes und persistentes XSS verhindern	295
5.8.2	DOM-basiertes XSS verhindern	297
5.9	Die Demo-Anwendung	298
5.9.1	Es gibt viel zu tun	299
5.9.2	Die Fundstellen (und zukünftigen Untersuchungsobjekte) im Überblick	306
5.9.3	Ist persistentes XSS über Einträge möglich?	311
5.9.4	Manipulation im Proxy	314
5.9.5	Wie sicher ist die Beta-Version?	316
5.9.6	SVG – für die einen ein Bild, für die anderen ein XSS-Angriff	320
5.9.7	Ist DOM-basiertes XSS möglich?	325
5.9.8	Jetzt kommt BeEF!	326
5.10	Links	340
6	SQL-Injection	345
6.1	Einführung	345
6.2	SQL-Injection im Überblick	348
6.2.1	Ein erstes Beispiel	348
6.2.2	Sicher oder nicht?	349
6.2.3	SQL-Injection bei der Authentifizierung	350
6.3	SQL-Injection: Schwachstellen finden	351
6.3.1	Strings prüfen	352
6.3.2	SQL-Injection statt Zahlen	352
6.3.3	SQL-Injection in verschiedene Statements	354
6.3.4	SQL-Injection mit Filter	362
6.3.5	SQL-Injection trotz Escape	364
6.3.6	SQL Column Truncation	366
6.3.7	SQL-Injection 2. Ordnung	367

6.3.8	SQL-Injection für Fortgeschrittene	368
6.3.9	Blind SQL-Injection – SQL-Injection ohne direkte Ausgabe	368
6.4	Angriffe »in the wild«	371
6.4.1	Drive-by-Infektionen über SQL-Injection vorbereitet, zum Ersten	371
6.4.2	Drive-by-Infektionen über SQL-Injection vorbereitet, zum Zweiten	374
6.4.3	SEO-Spam statt Drive-by-Infektionen über SQL-Injection vorbereiten	376
6.5	SQL-Injection verhindern	378
6.5.1	Benutzereingaben filtern	379
6.5.2	Numerische Daten	379
6.5.3	SQL-Injection 2. Ordnung	379
6.5.4	Einfaches Filtern reicht nicht	379
6.5.5	Sicher: parametrisierter Aufruf von Prepared Statements oder Stored Procedures	380
6.5.6	Defense-in-Depth	381
6.6	Die Demo-Anwendung	382
6.6.1	Die ID für die Auswahl des Eintrags	382
6.6.2	Die Zugangsdaten für das Backend, Teil 1	386
6.6.3	Die Zugangsdaten für das Backend, Teil 2	389
6.6.4	Die Daten bei der Registrierung eines Benutzers	393
6.6.5	SQL-Injection ist schlecht, darüber Code einschleusen schlechter	398
6.6.6	Alles Handarbeit? Wozu gibt es Tools?	400
6.6.7	Kleiner Tipp zum Schluss	406
6.7	Links	406
7	Injection jenseits von SQL	407
7.1	OS-Command-Injection	407
7.1.1	Beispiele für Command-Injection	407
7.1.2	Command-Injection-Schwachstellen finden	409
7.1.3	Mögliche Angriffe	413
7.2	SMTP-Injection	413
7.2.1	SMTP-Header-Injection	414
7.2.2	SMTP-Command-Injection	417
7.2.3	SMTP-Injection-Schwachstellen finden	418
7.3	LDAP-Injection	419
7.3.1	LDAP – eine kurze Einführung	419
7.3.2	LDAP-Injection am Beispiel	420
7.3.3	LDAP-Injection-Schwachstellen finden	423

7.4	NoSQL-Injection	424
7.4.1	NoSQL-Injection-Schwachstellen finden	425
7.5	Scriptcode-Injection	425
7.5.1	Dynamische Ausführung neu generierten Codes	426
7.5.2	Scriptcode-Injection-Schwachstellen finden	428
7.6	SOAP-Injection	429
7.6.1	Ein Beispiel	429
7.6.2	SOAP-Injection finden	433
7.6.3	SOAP-Injection ausnutzen	434
7.7	XPath-Injection	434
7.7.1	Ein einfaches Beispiel	434
7.7.2	Blind XPath-Injection	435
7.7.3	XPath-Injection-Schwachstellen finden	436
7.8	Injection-Angriffe abwehren	437
7.8.1	Command-Injection verhindern	437
7.8.2	SMTP-Injection verhindern	438
7.8.3	LDAP-Injection verhindern	438
7.8.4	NoSQL-Injection verhindern	438
7.8.5	Scriptcode-Injection verhindern	439
7.8.6	SOAP-Injection verhindern	439
7.8.7	XPath-Injection verhindern	439
7.9	Die Demo-Anwendung	440
7.9.1	OS-Command-Injection	440
7.9.2	SMTP-Injection	442
7.9.3	LDAP-, NoSQL-, Scriptcode-, SOAP- und XPath-Injection	451
7.10	Links	451

8 Dateioperationen: von Directory-Traversal bis Datei-Uploads 453

8.1	Directory-Traversal	453
8.1.1	Verzeichnis wechsele dich	455
8.1.2	Quelltext ausgeben lassen	458
8.1.3	Datei nicht gefunden	458
8.1.4	Ein fester Pfad als Präfix schreit nach Directory-Traversal	459
8.1.5	Eine feste Endung als Suffix ist ungünstiger	459
8.1.6	Ausführliche Fehlermeldungen erfreuen den Angreifer	460

8.1.7	Directory-Traversal ohne Kenntnis des Startverzeichnisses	460
8.1.8	Freie Auswahl	460
8.1.9	Schreiben ist besser als Lesen	460
8.1.10	To write or not to write?	461
8.2	Sonderfall PHP: Local File Inclusion	461
8.3	Der Schadcode: Webshells	463
8.3.1	Die c99-Shell von locus7s	463
8.3.2	Die r57shell v.1.42 – Edited By KingDefacer	471
8.4	Suche nach Directory-Traversal-Schwachstellen	475
8.4.1	Mögliche Angriffspunkte ermitteln	475
8.4.2	Verdächtige Parameter prüfen	476
8.4.3	Directory-Traversal-Sequenzen tarnen	477
8.4.4	Filterfunktionen austricksen	477
8.4.5	Suffix und/oder Präfix nach Wunsch	478
8.5	Abwehr von Directory-Traversal-Angriffen	479
8.5.1	Sicher: keine Dateisystemaufrufe mit Benutzerdaten	479
8.5.2	Risikant: Zugriff über benutzerkontrollierte Parameter	481
8.6	Remote File Inclusion	481
8.6.1	Einige Einschränkungen	482
8.6.2	Weiter mit PHP	483
8.6.3	Entfernte Dateien in PHP-Skripte einbinden	483
8.6.4	Der Schadcode: Webshells	484
8.6.5	RFI-Schwachstellen suchen	484
8.6.6	RFI-Schwachstellen verhindern	485
8.7	Datei-Uploads	485
8.7.1	Ein Beispiel für Datei-Uploads	485
8.7.2	Schwachstellen beheben, Angriffe abwehren	491
8.7.3	Datei-Upload absichern	495
8.7.4	Schwachstellen im Datei-Upload finden	496
8.8	Die Demo-Anwendung	496
8.8.1	Directory-Traversal und LFI über die Sprachauswahl	496
8.8.2	Directory-Traversal und LFI über das eingebundene Backend-Plugin	499
8.8.3	Directory-Traversal und LFI über das eingebundene Backend-Plugin der Beta-Version	500
8.8.4	Directory-Traversal über das weitergeleitete Backend-Plugin	502
8.8.5	Directory-Traversal mit Schreiben über das simulierte Kontaktformular	502
8.8.6	Angriff über die Upload-Funktion für Avatarbilder	504
8.9	Links	506