

Christopher Kunz • Peter Prochaska

# PHP-Sicherheit

PHP/MySQL-Webanwendungen  
sicher programmieren

dpunkt.verlag

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Über dieses Buch	1
1.2	Was ist Sicherheit?	3
1.3	Wichtige Begriffe	4
1.4	Sicherheitskonzepte	7
1.5	ISO 17799	8
1.6	Wie verkaufe ich Sicherheit?	10
1.7	Wichtige Informationsquellen	11
1.7.1	Mailinglisten	12
1.7.2	Füll Disclosure	12
1.7.3	BugTraq	13
1.7.4	Webappsec	14
1.7.5	phpsec	14
1.8	OWASP	15
1.9	PHP Security Consortium	16
1.10	PHP-Sicherheit.de	16
<b>2</b>	<b>Informationsgewinnung</b>	<b>17</b>
2.1	Grundlagen	17
2.2	Webserver erkennen	18
2.2.1	Server-Banner erfragen	19
2.2.2	Webserver-Verhalten interpretieren	21
2.2.3	Tools für Webserver-Fingerprinting	22
2.3	Betriebssystem erkennen	22
2.4	PHP-Installation erkennen	23
2.5	Datenbanksystem erkennen	24

2.6	Datei-Altlasten . . . . .	26
2.6.1	Temporäre Dateien . . . . .	26
2.6.2	Include- und Backup-Dateien . . . . .	27
2.6.3	Dateien von Entwicklungswerkzeugen . . . . .	28
2.6.4	Vergessene oder »versteckte« PHP-Dateien . . . . .	29
2.7	Pfade . . . . .	29
2.7.1	mod_speling . . . . .	29
2.7.2	robots.txt . . . . .	31
2.7.3	Standardpfade . . . . .	31
2.7.4	Pfade verkürzen . . . . .	33
2.8	Kommentare aus HTML-Dateien . . . . .	34
2.9	Applikationen erkennen . . . . .	34
2.9.1	Das Aussehen/Layout . . . . .	35
2.9.2	Das Vorhandensein bestimmter Dateien . . . . .	35
2.9.3	Header-Felder . . . . .	35
2.9.4	Bestimmte Pfade . . . . .	36
2.9.5	Kommentare im Quellcode . . . . .	36
2.10	Default-User . . . . .	37
2.11	Google Hacking . . . . .	37
2.12	Fazit . . . . .	38
<b>3</b>	<b>Parametermanipulation</b>	<b>39</b>
3.1	Grundlagen . . . . .	39
3.2	Werkzeuge zur Parametermanipulation . . . . .	42
3.2.1	Parametermanipulation mit dem Browser . . . . .	43
3.2.2	Einen Proxy benutzen . . . . .	44
3.3	Angriffsszenarien und Lösungen . . . . .	46
3.3.1	Fehlererzeugung . . . . .	46
3.3.2	HTTP Response Splitting . . . . .	48
3.3.3	Remote Command Execution . . . . .	52
3.3.4	Angriffe auf Dateisystemfunktionen . . . . .	55
3.3.5	Angriffe auf Shell-Ebene . . . . .	55
3.3.6	Cookie Poisoning . . . . .	56
3.3.7	Manipulation von Formulardaten . . . . .	57
3.3.8	Vordefinierte PHP-Variablen manipulieren . . . . .	58
3.4	Variablen richtig prüfen . . . . .	58
3.4.1	Auf Datentyp prüfen . . . . .	59
3.4.2	Datenlänge prüfen . . . . .	60
3.4.3	Inhalte prüfen . . . . .	61

3.4.4	Whitelist-Prüfungen	63
3.4.5	Blacklist-Prüfung	64
3.4.6	Clientseitige Validierung	66
3.5	register_globals	66
3.6	Fazit	69
<b>4</b>	<b>Cross-Site Scripting</b>	<b>71</b>
4.1	Grenzenlose Angriffe	71
4.2	Was ist Cross-Site Scripting?	72
4.3	Warum XSS gefährlich ist	73
4.4	Erhöhte Gefahr dank Browserkomfort	74
4.5	Formularvervollständigung verhindern	75
4.6	XSS in LANs und WANs	76
4.7	XSS - einige Beispiele	77
4.8	Ein klassisches XSS	78
4.9	Angriffspunkte für XSS	79
4.10	Angriffe verschleiern - XSS Cheat Sheet	80
4.11	Einfache Gegenmaßnahmen	84
4.12	XSS verbieten, HTML erlauben - wie?	86
4.12.1	BBCode	86
4.12.2	HTML-Filter mit XSS-Entfernung	88
4.13	Die Zwischenablage per XSS auslesen	90
4.14	XSS-Angriffe über DOM	91
4.15	XSS in HTTP-Headern	94
4.15.1	Angriffe der ersten Ordnung mit Headern	94
4.15.2	Second Order XSS per Header	95
4.16	Second Order XSS per RSS	97
4.17	Cross-Site Request Forgery (CSRF)	98
4.18	CSRF als Firewall-Brecher	101
4.19	CSRF in BBCode	102
4.20	Schutz gegen CSRF	103
4.21	Unheilise Allianz - CSRF und XSS	104

<b>5</b>	<b>SQL-Injection</b>	<b>105</b>
5.1	Grundlagen	105
5.2	Auffinden von SQL-Injection-Möglichkeiten	107
5.2.1	GET-Parameter	108
5.2.2	POST-Parameter	109
5.2.3	Cookie-Parameter	110
5.2.4	Server-Variablen	111
5.3	Syntax einer SQL-Injection	113
5.3.1	Sonderzeichen in SQL	114
5.3.2	Schlüsselwörter in SQL	115
5.3.3	Einfache SQL-Injection	115
5.3.4	UNION-Injections	117
5.4	Advanced SQL-Injection	120
5.4.1	LOAD_FILE	120
5.4.2	Denial-of-Service mit SQL-Injection	121
5.4.3	ORDER BY Injection	121
5.5	Wie kann man sich vor SQL-Injection schützen?	123
5.5.1	Sonderzeichen maskieren	123
5.5.2	Ist Schlüsselwort-Filterung ein wirksamer Schutz?	123
5.5.3	Parameter Binding/Prepared Statements	124
5.5.4	Stored Procedures	125
5.6	Fazit	126
<b>6</b>	<b>Autorisierung und Authentisierung</b>	<b>127</b>
6.1	Beliebte Fehler in Login-Formularen	127
6.1.1	Falsche Request-Methode	127
6.1.2	Falsche SQL-Abfrage	128
6.1.3	SQL-Injection	129
6.1.4	XSS	129
6.2	Authentisierungssicherheit	130
6.2.1	SSL	130
6.2.2	Behandlung von Passwörtern	132
6.2.3	Benutzernamen und Kennungen	133
6.2.4	Sichere Passwörter	134
6.2.5	Passwort-Sicherheit bestimmen	137
6.2.6	Vergessene Passwörter	140
6.3	Spam-Vermeidung mit CAPTCHAs	145

<b>7</b>	<b>Sessions</b>	<b>149</b>
7.1	Grundlagen . . . . .	149
7.2	Permissive oder strikte Session-Systeme. . . . .	151
7.3	Session-Speicherung. . . . .	152
7.4	Schwache Session-ID-Generierungsalgorithmen. . . . .	154
7.5	Session-Timeout. . . . .	155
7.6	Bruteforcing von Sessions. . . . .	156
7.7	Session Hijacking. . . . .	157
7.8	Session Fixation. . . . .	159
7.9	Zusätzliche Abwehrmethoden. . . . .	160
	7.9.1 Page Ticket System. . . . .	160
	7.9.2 Session-Dateien mittels Cronjob löschen. . . . .	161
	7.9.3 Session-ID aus dem Referrer löschen. . . . .	161
7.10	Fazit . . . . .	162
<b>8</b>	<b>Upload-Formulare</b>	<b>163</b>
8.1	Grundlagen. . . . .	163
8.2	Aufbau eines Upload-Formulars. . . . .	163
8.3	PHP-interne Verarbeitung. . . . .	164
8.4	Speicherung der hochgeladenen Dateien. . . . .	165
8.5	Bildüberprüfung. . . . .	166
8.6	PHP-Code in ein Bild einfügen. . . . .	167
8.7	Andere Dateitypen überprüfen. . . . .	168
8.8	Gefährliche Zip-Archive. . . . .	169
8.9	Fazit . . . . .	169
<b>9</b>	<b>PHP intern</b>	<b>171</b>
9.1	Fehler in PHP. . . . .	171
	9.1.1 File-Upload-Bug. . . . .	171
	9.1.2 CGI-Lücke in PHP 4.3.0. . . . .	172
	9.1.3 Unsichere (De-)Serialisierung. . . . .	172
	9.1.4 Gefährliches Speicherlimit. . . . .	172
	9.1.5 Bewertung. . . . .	172
9.2	Bestandteile eines sicheren Servers. . . . .	173
9.3	Unix oder Windows?. . . . .	174
9.4	Bleiben Sie aktuell!. . . . .	175

9.5	Installation	175
9.5.1	Installation als Apache-Modul	176
9.5.2	CGI	177
9.6	suExec	179
9.7	Safe Mode	181
9.7.1	Einrichtung des Safe Mode	182
9.7.2	safe_mode_exec_dir	182
9.7.3	safe_mode_include_dir	183
9.7.4	Umgebungsvariablen im Safe Mode	183
9.7.5	Safe Mode considered harmful?	184
9.8	Weitere PHP-Einstellungen	186
9.8.1	open_basedir	186
9.8.2	disablejunctions	187
9.8.3	disable_classes	187
9.8.4	max_execution_time	188
9.8.5	max_input_time	188
9.8.6	memory_limit	188
9.8.7	Upload-Einstellungen	189
9.8.8	allow_url_fopen	190
9.8.9	register_globals	190
9.9	Code-Sandboxing mit runkit	191
9.10	Externe Ansätze	193
9.10.1	suPHP	193
9.10.2	FastCGI	197
9.10.3	Das Apache-Modul mod_suid	198
9.11	Rootjail-Lösungen	202
9.11.1	BSD-Rootjails	203
9.11.2	User Mode Linux	203
9.11.3	mod_security	203
9.11.4	mod_chroot	204
9.12	Fazit	204
<b>10</b>	<b>PHP-Hardening</b>	<b>207</b>
10.1	Warum PHP härten?	207
10.1.1	Buffer Overflows	208
10.1.2	Schutz vor Pufferüberläufen im Hardening-Patch	209
10.1.3	Schutz vor Format-String-Schwachstellen	209
10.1.4	Include-Schutz gegen Remote-Includes und Nullbytes	210

10.1.5	Funktions- und Evaluations-Beschränkungen	212
10.1.6	Schutz gegen Response Splitting	212
10.1.7	Variablenschutz	212
10.1.8	SQL Intrusion Detection	213
10.1.9	Logging	213
10.1.10	Sichere XMLRPC-Bibliothek	213
10.1.11	Schreibgeschütztes Speicherlimit	213
10.1.12	Kryptographische Funktionen	214
10.2	Prinzipien hinter dem Hardening-Patch	214
10.3	Installation	215
10.4	Zusammenarbeit mit Caching-Lösungen	219
10.5	Konfiguration	219
10.5.1	Generelle Optionen	220
10.5.2	Log-Dateien	223
10.5.3	Alarm-Skript	225
10.5.4	Variablenfilter	225
10.5.5	Upload-Konfiguration	227
10.6	Beispielkonfiguration	229
10.7	Fazit und Ausblick	230
<b>11</b>	<b>Apache-Hardening mit mod_security</b>	<b>231</b>
11.1	Einsatzgebiet von mod_security	231
11.2	Und so funktioniert's	232
11.3	Gefahren durch mod_security	233
11.4	Installation	234
11.5	Konfiguration	235
11.6	Regelwerk von mod_security	238
11.6.1	Starre Regeln und dynamische Klassifizierung	238
11.6.2	So funktionieren Regeln in mod_security	240
11.6.3	Wehrhaft - Filter-Aktionen	240
11.6.4	SecFilter und SecFilterSelective	243
11.6.5	Normalisierung von Anfragen	245
11.6.6	Anwendungsspezifische Regeln mit SecFilter	245
11.6.7	Verkettete Regeln mit selektiver Suche	246
11.7	Alarm-Skript für mod_security	248
11.8	Rootjail-Umgebungen mit mod_security	248
11.9	Fazit	250



<b>Anhang</b>	<b>251</b>	
<b>A</b>	<b>Checkliste für sichere Webapplikationen</b>	<b>253</b>
<b>B</b>	<b>Wichtige Optionen in php.ini</b>	<b>257</b>
B.1	variables_order . . . . .	257
B.2	register_globals . . . . .	258
B.3	register_long_arrays . . . . .	258
B.4	register_argc_argv . . . . .	258
B.5	post_max_size . . . . .	259
B.6	magic_quotes_gpc . . . . .	259
B.7	magic_quotes_runtime . . . . .	259
B.8	always_populate_raw_post_data . . . . .	259
B.9	allow_urljopen . . . . .	260
<b>C</b>	<b>Liste aller Schwachstellen mit Gefahrenpotenzial-Bewertung</b>	<b>261</b>
C.1	Cross-Site Scripting . . . . .	261
C.2	Information Disclosure . . . . .	261
C.3	Füll Path Disclosure . . . . .	262
C.4	SQL-Injection . . . . .	262
C.5	HTTP Response Splitting . . . . .	262
C.6	Cross-Site Request Forgery . . . . .	263
C.7	Remote Command Execution . . . . .	263
<b>D</b>	<b>Glossar</b>	<b>265</b>
	<b>Stichwortverzeichnis</b>	<b>273</b>