



mitp

Marc
Schönefeld

Java Security

Sicherheitslücken identifizieren und vermeiden

Inhaltsverzeichnis

	Einführung	11
I	Grundlagen	15
I.1	Grundlagen der Java-Sicherheitsarchitektur	15
I.1.1	Grundpfeiler der IT-Sicherheit	15
I.1.2	Vertraulichkeit	15
I.1.3	Integrität	15
I.1.4	Verfügbarkeit	16
I.1.5	Nicht-Abstreitbarkeit	16
I.2	Sichere Produktentwicklung	16
I.3	Verteilte Ebenen der Sicherheit	19
I.3.1	Sicherheitsaspekte verteilter Systeme	19
I.3.2	Sicherheitsbedrohungen im Kontext von Java-Unternehmensanwendungen	21
I.4	Java-Sprachdesign und die JVM	24
I.4.1	Highlights der Java-Programmiersprache	24
I.4.2	Das Java Runtime Environment	26
I.5	Das Java-Sicherheitsmodell	28
I.5.1	Das erste Sandbox-Modell	28
I.5.2	Das erweiterte Sandbox-Modell	29
I.5.3	Feingranulare Vergabe von Rechten	29
I.5.4	Implementierung des Sandbox-Modells	30
I.5.5	Der Access Controller	38
I.6	Praktischer Exkurs: Ermittlung von Least Privilege Policies	40
I.6.1	Manuelle Ermittlung einer Least Privilege Policy	40
I.6.2	Automatische Ermittlung einer Least Privilege Policy	43
I.6.3	Zusammenfassung	49
2	Sichere Java-Programmierung	51
2.1	Ziele des sicheren Programmierens	51
2.2	Die einzelnen Grundschutzmaßnahmen	51
2.3	Privilegierter Code	52
2.4	Nicht-finale statische Felder	56

2.5	Anpassung von Sichtbarkeitsbereichen	57
2.6	Abgeschlossene Paketdefinitionen	59
2.7	Unveränderliche Objekte	61
2.8	Defensive Serialisierung	62
2.9	Native Methoden	64
2.I0	Sicherer Umgang mit Heap-Speicher	67
2.II	Zusammenfassung	69
3	Werkzeuge	71
3.1	JDK-eigene Werkzeuge	71
3.1.1	JDB	71
3.1.2	DTrace	77
3.1.3	Javap	79
3.2	Zusätzliche Werkzeuge	80
3.2.1	Findbugs	80
3.2.2	Byteman	84
3.2.3	Decompiler	91
3.2.4	JAD	92
3.2.5	JD-GUI	93
3.3	Zusammenfassung	94
4	Verwundbarkeiten	95
4.1	Definitionen	95
4.1.1	Schwachstelle	95
4.1.2	CWE	96
4.1.3	CVE	97
4.1.4	CVSS2	98
4.1.5	Berechnung von CVSS2-Werten	99
4.1.6	oDay, Exploit und Embargo	108
4.2	Beispiele für Verwundbarkeiten	108
4.2.1	CWE-119: Unzureichende Überprüfung der Bereichsgrenzen bei Zugriffen auf Speicherblöcke	108
4.2.2	CWE-805: Zugriff auf Speicherblock mit falschen Längenwert	113
4.2.3	CWE-835: Schleife mit unerreichbarer Abbruchbedingung	121
4.2.4	CWE-267: Erweiterte Privilegien erlauben unsicheren Aktionen	124

4.2.5	CWE-668: Freigabe einer Ressource in nicht vorgesehener Umgebung.	137
4.2.6	CWE-209: Informations-Preisgabe durch Fehlermeldungen	141
4.2.7	CWE-271: Fehlender Verzicht auf Privilegien	146
4.2.8	CWE-213: Vorsätzliche Preisgabe vertraulicher Daten	149
4.2.9	CWE-77: Befehls-Injektion durch fehlende Neutralisierung von Spezialelementen zur Kommandozeilen-Ausführung	153
4.3	Fazit	156
5	Malware und Java	157
5.1	Vorgefertigte Malware	158
5.2	Malware-Angriffsschema	158
5.2.1	Ablauf eines Angriffs	159
5.2.2	Verschleierung von Java-Malware	159
5.2.3	Analyse einer Java-Malware	160
5.3	CWE-502, die Schwachstelle hinter der Malware	173
5.4	Zusammenfassung	176
6	Aktive Fehlersuche	177
6.1	Fuzzing	177
6.1.1	Methoden	177
6.1.2	Einsatzbereiche	178
6.2	Fuzzer im praktischen Einsatz	178
6.2.1	Fuzzing mit zzuf	180
6.2.2	Fuzzing mit Honggfuzz	181
6.2.3	Untersuchung mit Crash-Analyse-Tools	183
6.2.4	Untersuchung mit dem Debugger	186
6.2.5	Ein Java-Fuzzer	188
6.3	Zusammenfassung	194
7	Schwächen in Enterprise Java	195
7.1	Enterprise Java	195
7.2	Sicherheitsmängel in Enterprise Java	195
7.3	CWE-227: Unzureichende Erfüllung von API-Vereinbarungen	196
7.3.1	Vorbereitung des Angriffs	197
7.3.2	Ausführen des Angriffs	199

7.3.3	Abhärtungsansätze	199
7.3.4	Zusätzliche Verwundbarkeiten	200
7.3.5	Behebung der Schwachstelle	201
7.3.6	Zusammenfassung	201
7.4	CWE-264: Fehler in der Verarbeitung von Rechten, Privilegien und der Zugriffskontrolle.	202
7.4.1	Die JMX-Console	203
7.4.2	Verb tampering.	204
7.4.3	Angriffsvorbereitung	204
7.4.4	Ausnutzung der Schwachstelle	205
7.4.5	Vermeidung	205
7.5	CWE-Design-Error: Fehlerursache liegt im Programmwurf, nicht in der Implementierung	206
7.5.1	Die Verwundbarkeit	207
7.5.2	Undokumentiertes Verhalten	207
7.5.3	Mitigation	208
7.6	CWE-20: Unzureichende Überprüfung von Eingabedaten führt zur Beeinflussung des Kontrollflusses	209
7.6.1	Motivation	209
7.6.2	Hintergrund	209
7.6.3	Demonstration des Angriffs	215
7.6.4	Fehlerbehebung	216
7.6.5	Zusammenfassung.	219
7.7	Fazit	219
8	Bytecode-basierte Audits	221
8.1	Bytecode-Grundlagen	221
8.1.1	javap.	221
8.1.2	jasmin	225
8.1.3	BCEL	227
8.1.4	Bytecode-Feinheiten	229
8.1.5	Weitere Informationen	230
8.2	Bytecode-orientierte Security-Audits	231
8.3	Das jDetect-Plugin	232
8.4	Erstellung eines Detektors	232
8.5	Paketierung	238
8.6	Zusammenfassung	244

9	Tipps und Tricks	245
9.1	Der Java-Compiler	245
9.1.1	Beispiele verbotener Klassen	246
9.1.2	Das Symbol-File enträtselt	247
9.1.3	Wirksamkeit außerhalb von javac	250
9.2	Methoden visualisieren	250
9.2.1	Implementierung	251
9.2.2	Grafische Darstellung	260
9.3	Fazit	260
	Schlusswort	261
	Stichwortverzeichnis	263

