Inhalt

Einleitung **1**
E.1 Überblick 1
E.2 Testsystem – Plattform und Architektur 2
E.3 Darstellungskonventionen 4
E.4 Voraussetzungen des Lesers 5
E.5 Anregungen, Fragen und weitere Themenbereiche 5

1 Buffer Overflows **7**
1.1 Einteilung und Klassifizierung 8
1.2 Prozess- und Speicherorganisation 9
1.3 Buffer Overflows der 1. Generation: Klassische Stack-basierte Buffer Overflows 18
1.3.1 Der Stack – Funktionsweise 19
1.3.2 Register 20
1.3.3 Der Stack – Steuerung 23
1.3.4 Funktionen 25
1.3.5 Schwachstelle 40
1.3.6 Angriffsmöglichkeit – Denial of Service 49
1.3.7 Angriffsmöglichkeit – Gezielte Modifikation des Programmflusses 52
1.3.8 Angriffsmöglichkeit – Ausführung eingeschleusten Programmcodes 61
1.4 Buffer Overflows der 2. Generation: Off-by-Ones und Frame Pointer Overwrites 94
1.4.1 Frame Pointer Overwrite 96
1.4.2 Manipulation von Zeigern 126
1.5 Buffer Overflows der 3. Generation: BSS Overflows 129
1.5.1 Manipulation von Zeigern 132
1.5.2 Off-by-One 148
1.6 Buffer Overflows der 4. Generation: Heap Overflows 151

2 Buffer Overflows – Gegenmaßnahmen **187**
2.1 Gegenmaßnahmen – Verschiedene Ansätze 187
2.2 Ursachenbekämpfung 191
2.3 Gegenmaßnahme – Sichere Programmierung 191
2.3.1 Unsichere Bibliotheksfunktionen 192
2.4 Gegenmaßnahme – Source Code Audit 239
2.5 Gegenmaßnahme – Automatisierte Software-Tests 241
2.5.1 Statische Analysen – Source Code Analyzer 242
2.5.2 Dynamische Analysen – Tracer 271

2.6 Gegenmaßnahme – Binary Audit 286
2.6.1 Fault Injection 287
2.6.2 Reverse Engineering 295
2.7 Bekämpfung der Auswirkungen 303
2.8 Gegenmaßnahme – Compiler-Erweiterungen 304
2.8.1 Bounds Checking 304
2.8.2 StackGuard 305
2.8.3 /GS-Option von Microsoft 347
2.8.4 StackShield 350
2.8.5 Zusammenfassung: Möglichkeiten und Grenzen von StackGuard, /GS-Option und StackShield 385
2.8.6 Weitere Compiler-Erweiterungen 386
2.9 Gegenmaßnahme – Wrapper für »unsichere« Bibliotheksfunktionen 387
2.9.1 Libsafe 387
2.10 Gegenmaßnahme – Modifikation der Prozessumgebung . . 396
2.10.1 Non-executable Stack 396
2.10.2 PaX 431
2.10.3 Weitere Implementationen 461

3 Format-String-Schwachstellen 463
3.1 Funktionen zur formatierten Ausgabe 463
3.2 Der Format-String 465
3.3 Schwachstelle 467
3.4 Stack 467
3.5 Angriffsmöglichkeit – Denial of Service 488
3.6 Angriffsmöglichkeit – Gezieltes Überschreiben von Speicherstellen 491
3.6.1 One-Shot-Methode 492
3.6.2 Short-Write-Methode 506
3.6.3 Per-Byte-Write-Methode 511
3.6.4 Ausführung eingeschleusten Programmcodes 512
3.7 Fazit 523

4 Format-String-Schwachstellen – Gegenmaßnahmen 525
4.1 Gegenmaßnahmen – Verschiedene Ansätze 525
4.2 Ursachenbekämpfung 526
4.3 Gegenmaßnahme – Sichere Programmierung und Source Code Audit 526
4.4 Statische Analysen – Source Code Analyzer 531
4.4.1 Lexikalische Source Code Analyzer 531
4.4.2 Semantische Source Code Analyzer 546
4.4.3 Source Code Analyzer – Type Qualifiers 553
4.5 Gegenmaßnahme – Binary Audit 560
4.6 Bekämpfung der Auswirkungen 561

4.7 Gegenmaßnahme – Wrapper für fehleranfällige Bibliotheksfunktionen 561
4.7.1 Libformat 561
4.7.2 FormatGuard 565
4.8 Modifikationen der Prozessumgebung 571
4.8.1 Openwall: Non-executable Stack 571
4.8.2 PaX 592

5 Weitere Möglichkeiten 613
5.1 Alternative Programmiersprachen – Vorzüge, Risiken und Nebenwirkungen 613
5.2 Principle of Least Privilege – Einschränkung von Rechten . . 624

Anhang A – Assembler-Codes 627

Anhang B – Shellcodes 628

Anhang C – fosbi 630

Anhang D – No-NOP-Technik 638

Anhang E – flawfinder 644

Literaturverzeichnis 649

Index 658