

Contents

	Foreword	xvii
	Preface	xxi
	Acknowledgments	xxv
	About the Author	xxvii
Chapter 1	Running with Scissors	1
	1.1 Gauging the Threat	5
	What Is the Cost?	6
	Who Is the Threat?	8
	Software Security	11
	1.2 Security Concepts	12
	Security Policy	14
	Security Flaws	14
	Vulnerabilities	15
	Exploits	16
	Mitigations	17
	1.3 C and C++	17
	A Brief History	19
	What Is the Problem with C?	21
	Legacy Code	24
	Other Languages	25
	1.4 Development Platforms	25
	Operating Systems	26
	Compilers	26

1.5	Summary	27
1.6	Further Reading	28
Chapter 2	Strings	29
2.1	Character Strings	29
	String Data Type	30
	UTF-8	32
	Wide Strings	33
	String Literals	34
	Strings in C++	36
	Character Types	37
	Sizing Strings	39
2.2	Common String Manipulation Errors	42
	Improperly Bounded String Copies	42
	Off-by-One Errors	47
	Null-Termination Errors	48
	String Truncation	49
	String Errors without Functions	49
2.3	String Vulnerabilities and Exploits	50
	Tainted Data	51
	Security Flaw: IsPasswordOK	52
	Buffer Overflows	53
	Process Memory Organization	54
	Stack Management	55
	Stack Smashing	59
	Code Injection	64
	Arc Injection	69
	Return-Oriented Programming	71
2.4	Mitigation Strategies for Strings	72
	String Handling	73
	C11 Annex K Bounds-Checking Interfaces	73
	Dynamic Allocation Functions	76
	C++ <code>std::basic_string</code>	80
	Invalidating String Object References	81
	Other Common Mistakes in <code>basic_string</code> Usage	83
2.5	String-Handling Functions	84
	<code>gets()</code>	84
	C99	84
	C11 Annex K Bounds-Checking Interfaces: <code>gets_s()</code>	86
	Dynamic Allocation Functions	87
	<code>strcpy()</code> and <code>strcat()</code>	89
	C99	89
	<code>strncpy()</code> and <code>strncat()</code>	93
	<code>memcpy()</code> and <code>memmove()</code>	100
	<code>strlen()</code>	100

2.6	Runtime Protection Strategies	101
	Detection and Recovery	101
	Input Validation	102
	Object Size Checking	102
	Visual Studio Compiler-Generated Runtime Checks	106
	Stack Canaries	108
	Stack-Smashing Protector (ProPolice)	110
	Operating System Strategies	111
	Detection and Recovery	111
	Nonexecutable Stacks	113
	W ^X	113
	PaX	115
	Future Directions	116
2.7	Notable Vulnerabilities	117
	Remote Login	117
	Kerberos	118
2.8	Summary	118
2.9	Further Reading	120
Chapter 3	Pointer Subterfuge	121
3.1	Data Locations	122
3.2	Function Pointers	123
3.3	Object Pointers	124
3.4	Modifying the Instruction Pointer	125
3.5	Global Offset Table	127
3.6	The <code>.dtors</code> Section	129
3.7	Virtual Pointers	131
3.8	The <code>atexit()</code> and <code>on_exit()</code> Functions	133
3.9	The <code>longjmp()</code> Function	134
3.10	Exception Handling	136
	Structured Exception Handling	137
	System Default Exception Handling	139
3.11	Mitigation Strategies	139
	Stack Canaries	140
	W ^X	140
	Encoding and Decoding Function Pointers	140
3.12	Summary	142
3.13	Further Reading	143
Chapter 4	Dynamic Memory Management	145
4.1	C Memory Management	146
	C Standard Memory Management Functions	146
	Alignment	147
	<code>alloca()</code> and Variable-Length Arrays	149

4.2	Common C Memory Management Errors	151
	Initialization Errors	151
	Failing to Check Return Values	153
	Dereferencing Null or Invalid Pointers	155
	Referencing Freed Memory	156
	Freeing Memory Multiple Times	157
	Memory Leaks	158
	Zero-Length Allocations	159
	DR #400	161
4.3	C++ Dynamic Memory Management	162
	Allocation Functions	164
	Deallocation Functions	168
	Garbage Collection	169
4.4	Common C++ Memory Management Errors	172
	Failing to Correctly Check for Allocation Failure	172
	Improperly Paired Memory Management Functions	172
	Freeing Memory Multiple Times	176
	Deallocation Function Throws an Exception	179
4.5	Memory Managers	180
4.6	Doug Lea's Memory Allocator	182
	Buffer Overflows on the Heap	185
4.7	Double-Free Vulnerabilities	191
	Writing to Freed Memory	195
	RtlHeap	196
	Buffer Overflows (Redux)	204
4.8	Mitigation Strategies	212
	Null Pointers	212
	Consistent Memory Management Conventions	212
	phkmalloc	213
	Randomization	215
	OpenBSD	215
	The jemalloc Memory Manager	216
	Static Analysis	217
	Runtime Analysis Tools	218
4.9	Notable Vulnerabilities	222
	CVS Buffer Overflow Vulnerability	222
	Microsoft Data Access Components (MDAC)	223
	CVS Server Double-Free	223
	Vulnerabilities in MIT Kerberos 5	224
4.10	Summary	224
Chapter 5	Integer Security	225
5.1	Introduction to Integer Security	225
5.2	Integer Data Types	226
	Unsigned Integer Types	227

	Wraparound	229
	Signed Integer Types	231
	Signed Integer Ranges	235
	Integer Overflow	237
	Character Types	240
	Data Models	241
	Other Integer Types	241
5.3	Integer Conversions	246
	Converting Integers	246
	Integer Conversion Rank	246
	Integer Promotions	247
	Usual Arithmetic Conversions	249
	Conversions from Unsigned Integer Types	250
	Conversions from Signed Integer Types	253
	Conversion Implications	256
5.4	Integer Operations	256
	Assignment	258
	Addition	260
	Subtraction	267
	Multiplication	269
	Division and Remainder	274
	Shifts	279
5.5	Integer Vulnerabilities	283
	Vulnerabilities	283
	Wraparound	283
	Conversion and Truncation Errors	285
	Nonexceptional Integer Logic Errors	287
5.6	Mitigation Strategies	288
	Integer Type Selection	289
	Abstract Data Types	291
	Arbitrary-Precision Arithmetic	292
	Range Checking	293
	Precondition and Postcondition Testing	295
	Secure Integer Libraries	297
	Overflow Detection	299
	Compiler-Generated Runtime Checks	300
	Verifiably In-Range Operations	301
	As-If Infinitely Ranged Integer Model	303
	Testing and Analysis	304
5.7	Summary	307
Chapter 6	Formatted Output	309
6.1	Variadic Functions	310
6.2	Formatted Output Functions	313
	Format Strings	314

	GCC	318
	Visual C++	318
6.3	Exploiting Formatted Output Functions	319
	Buffer Overflow	320
	Output Streams	321
	Crashing a Program	321
	Viewing Stack Content	322
	Viewing Memory Content	324
	Overwriting Memory	326
	Internationalization	331
	Wide-Character Format String Vulnerabilities	332
6.4	Stack Randomization	332
	Defeating Stack Randomization	332
	Writing Addresses in Two Words	334
	Direct Argument Access	335
6.5	Mitigation Strategies	337
	Exclude User Input from Format Strings	338
	Dynamic Use of Static Content	338
	Restricting Bytes Written	339
	C11 Annex K Bounds-Checking Interfaces	340
	iostream versus stdio	341
	Testing	342
	Compiler Checks	342
	Static Taint Analysis	343
	Modifying the Variadic Function Implementation	344
	Exec Shield	346
	FormatGuard	346
	Static Binary Analysis	347
6.6	Notable Vulnerabilities	348
	Washington University FTP Daemon	348
	CDE ToolTalk	348
	Ettercap Version NG-0.7.2	349
6.7	Summary	349
6.8	Further Reading	351
Chapter 7	Concurrency	353
7.1	Multithreading	354
7.2	Parallelism	355
	Data Parallelism	357
	Task Parallelism	359
7.3	Performance Goals	359
	Amdahl's Law	361
7.4	Common Errors	362
	Race Conditions	362

	Corrupted Values	364
	Volatile Objects	365
7.5	Mitigation Strategies	368
	Memory Model	368
	Synchronization Primitives	371
	Thread Role Analysis (Research)	380
	Immutable Data Structures	383
	Concurrent Code Properties	383
7.6	Mitigation Pitfalls	384
	Deadlock	386
	Prematurely Releasing a Lock	391
	Contention	392
	The ABA Problem	393
7.7	Notable Vulnerabilities	399
	DoS Attacks in Multicore Dynamic Random-Access Memory (DRAM) Systems	399
	Concurrency Vulnerabilities in System Call Wrappers	400
7.8	Summary	401
Chapter 8	File I/O	403
8.1	File I/O Basics	403
	File Systems	404
	Special Files	406
8.2	File I/O Interfaces	407
	Data Streams	408
	Opening and Closing Files	409
	POSIX	410
	File I/O in C++	412
8.3	Access Control	413
	UNIX File Permissions	413
	Process Privileges	415
	Changing Privileges	417
	Managing Privileges	422
	Managing Permissions	428
8.4	File Identification	432
	Directory Traversal	432
	Equivalence Errors	435
	Symbolic Links	437
	Canonicalization	439
	Hard Links	442
	Device Files	445
	File Attributes	448
8.5	Race Conditions	450
	Time of Check, Time of Use (TOCTOU)	451

	Create without Replace	453
	Exclusive Access	456
	Shared Directories	458
8.6	Mitigation Strategies	461
	Closing the Race Window	462
	Eliminating the Race Object	467
	Controlling Access to the Race Object	469
	Race Detection Tools	471
8.7	Summary	472
Chapter 9	Recommended Practices	473
9.1	The Security Development Lifecycle	474
	TSP-Secure	477
	Planning and Tracking	477
	Quality Management	479
9.2	Security Training	480
9.3	Requirements	481
	Secure Coding Standards	481
	Security Quality Requirements Engineering	483
	Use/Misuse Cases	485
9.4	Design	486
	Secure Software Development Principles	488
	Threat Modeling	493
	Analyze Attack Surface	494
	Vulnerabilities in Existing Code	495
	Secure Wrappers	496
	Input Validation	497
	Trust Boundaries	498
	Blacklisting	501
	Whitelisting	502
	Testing	503
9.5	Implementation	503
	Compiler Security Features	503
	As-If Infinitely Ranged (AIR) Integer Model	505
	Safe-Secure C/C++	505
	Static Analysis	506
	Source Code Analysis Laboratory (SCALe)	510
	Defense in Depth	511
9.6	Verification	512
	Static Analysis	512
	Penetration Testing	513
	Fuzz Testing	513
	Code Audits	515
	Developer Guidelines and Checklists	516

Independent Security Review	516
Attack Surface Review	517
9.7 Summary	518
9.8 Further Reading	518
References	519
Acronyms	539
Index	545