ESI register, 45–46	firmware malware, 279–280
ESP register, 45–46	flags
European Union's Directive on the	carry flag (CF), 520–521
Legal Protection of Computer	defined, 519
Programs, 23	EFLAGS register, 519–520
evaluation stack (MSIL), 430	overflow flag (OF), 520–521
events, 86	parity flag (PF), 521
exception handlers, 105–107	sign flag (SF), 521
exceptions, 105–107	status flags, 46–47
EXECryptor (StrongBit Technology),	system flags, 46–47
345	zero flag (ZF), 521
executable data sections, 43	flow analysis
executable formats	data propagation, 468–470
directories, 99–102	data type propagation, 471–474
exports, 99	defined, 466–467
file alignment, 95	register variable identification,
headers, 97–98	470–471
image sections, 95	single static assignment (SSA),
imports, 99	467–468
relative virtual address (RVA), 95	flow control
relocations, 93–95	conditional blocks, 32
section alignment, 95–96	defined, 32
executable-dumping tools, 133–138	loops, 33
execution environments	low-level implementation, 43–44
defined, 60	switch blocks, 33
microprocessors, 63–68	front end of decompilers
virtual machines, 60–63	basic block (BB), 464–466
expression trees, 461–462	function of, 463
expressions, 461–462	semantic analysis, 463–464
_	function calls
F	assembly language instructions, 51
fastcall calling convention, 541	stack, 42
faults (pages), 73–74	"A Functional Taxonomy for Soft-
Felten vs. RIAA case, 22	ware Watermarking", J. Nagra, C.
file formats	Thomboroson, and C. Colberg,
.crx file format, 202–204	322
Microsoft Word file format, 200	function-level working-set tuning,
reversing, 202–204	515–517
file-backed section object, 78	functions
FileMon system-monitoring tool,	alldiv,530-534
130	allmul,530
finding crackmes, 420	calling, 487

G functions (continued) Cryptex command-line data GCC (GNU C Compiler) and G++ encryption tool, 205–207 (GNU C++ Compiler) version defined, 486 3.3.1 compiler, 59 epilogues, 486 General Method of Program Code (-functions, 468 Obfuscation, Gregory Wroblewski, imported, 487–488 347 internal, 487 generic data structures, 547–548 intrinsic string-manipulation funcgeneric data type arrays, 548 tions, 249–250 generic table API library functions, 475–476 callbacks prototypes, 195 prologues, 486 definition, 145–146, 194–196 RtlDeleteElementGeneric function prototypes, 196 Table, 193-194 internal data structures, 195 RtlGetElementGenericTable RtlDeleteElementGeneric disassembly, 153–155 Table function, 193-194 initialization, 155–159 RtlGetElementGenericTable logic and structure, 159–161 function search loop 1, 161–163 disassembly, 153–155 search loop 2, 163–164 initialization, 155–159 search loop 3, 164–165 logic and structure, 159–161 search loop 4, 165 search loop 1, 161–163 setup, 155–159 search loop 2, 163–164 source code, 165–168 search loop 3, 164–165 RtlInitializeGenericTable, search loop 4, 165 146-151 setup, 155–159 RtlInsertElementGeneric source code, 165-168 Table, 168-170 RtlInitializeGenericTable RtlIsGenericTableEmpty, function, 146-151 152 - 153RtlInsertElementGeneric RtlLocateNodeGenericTable, Table function, 168–170 170-178 RtlIsGenericTableEmpty RtlLookupElementGeneric function, 152–153 Table, 188-193 RtlLocateNodeGenericTable RtlNumberGenericTable function, 170-178 Elements, 151-152 RtlLookupElementGeneric RtlRealInsertElement Table function, 188–193 Worker, 178-186 RtlNumberGenericTable RtlSplay, 185-188 Elements function, 151-152 virtual functions, 557–560 RtlRealInsertElementWorker (-functions, 468 function, 178-186 RtlSplay function, 185–188

Genesis gaming console (Sega Enter-	DIV, 49-50, 524
prises), 18	DIV/IDIV, 524
GLOBAL?? directory, 83	ENTER, 538-540
global variables, 542	IDIV, 49–50, 524
GNU C Compiler (GCC) and GNU	IMUL, 49-50, 523
C++ Compiler (G++) compilers,	int 3,331
59	Jcc, 51
Grier, Aaron, Automatic Detection and	LEA, 522
Prevention of Buffer-Overflow	LEAVE, 538, 540
Attacks, 252	MOV, 49
ground rules for reversing sessions,	MOVSX, 535
142–143	MOVZX, 534-535
	MUL, 49-50, 523
Н	opcode (operation code), 47
Hacarmy.D, Trojan/Backdoor pro-	operands, 47–48
gram, 285–305	RET, 51, 540
Hack SDMI challenge, 22	SBB, 52 9
handles, 81	Set Byte on Condition (SETcc),
hardware breakpoints, 331–332	513–514
hardware exceptions, 105	SUB, 49–50, 522, 529
hardware-based copy protection	SYSENTER, 394
technologies, 316–317	IA-32 Intel Architecture Software
heap, 42	Developer's Manual, Volume 2A and
heap overflows, 255–256	Volume 2B reference manuals, 48
Hex Workshop (BreakPoint Soft-	IA-32 registers
ware, Inc.), 131–132	defined, 39, 44–45
high-level data management, 38	EAX, 45–46
high-level languages, 33–37	EBP, 45-46
Hinton, Heather, Automatic Detection	EBX, 45-46
and Prevention of Buffer-Overflow	ECX, 45-46
Attacks, 252	EDI, 45-46
,	EDX, 45–46
1	EFLAGS, 46, 519–520
IA-32 decompilers, 477	ESI, 45-46
IA-32 instructions	ESP, 45–46
ADC, 529	IDA (Interactive Disassembler),
ADD, 49–50, 522, 529	112–115, 121
CALL, 51, 487, 540	IDC, BSA and IDC Global Software
CDQ, 535	Piracy Study, 310
CMP, 50, 480–483	IDIV instruction, 49–50, 524
Conditional Move (CMOVcc),	IIS Indexing Service Vulnerability,
514–515	262–271
01T 010	<u> </u>

IL (Intermediate Language)	inherited classes, 555–556
activation records, 430	inlining, 353, 419
add instruction, 432	input/output system (Windows
beg instruction, 432	operating system), 103–104
bge instruction, 432	instruction sets for decompilers, 460
bgt instruction, 432	instructions (IA-32)
ble instruction, 432	ADC, 529
blt instruction, 432	ADD, 49–50, 522, 529
bne instruction, 432	CALL, 51, 487, 540
box instruction, 432	CDQ, 535
br instruction, 432	CMP, 50, 480-483
C#, 36–37	Conditional Move (CMOVcc),
call instruction, 431	514–515
code samples	DIV, 49-50, 524
counting items, 433–435	DIV/IDIV,524
linked lists, 436–443	ENTER, 538-540
details, 424	IDIV, 49-50, 524
div instruction, 432	IMUL, 49-50, 523
evaluation stack, 430	int 3,331
ldarg instruction, 431	Jcc, 51
1dc instruction, 431	LEA, 522
ldfld instruction, 431	LEAVE, 538, 540
ldloc instruction, 431	MOV, 49
mul instruction, 432	MOVSX, 535
.NET executables, 429	MOVZX, 534-535
newarr instruction, 433	MUL, 49-50, 523
newobj instruction, 433	opcode (operation code), 47
ret instruction, 431	operands, 47–48
starg instruction, 431	RET, 51, 540
stfld instruction, 431	SBB, 529
stloc instruction, 431	Set Byte on Condition (SETcc),
sub instruction, 432	513–514
switch instruction, 432	SUB, 49–50, 522, 529
unbox instruction, 432	SYSENTER, 394
ILDasm, 115–116	instructions (MSIL)
imported functions, 487–488	add, 432
imported variables, 544–546	beg, 432
IMUL instruction, 49–50, 523–524	bge, 432
information theft, 281	bgt, 432
information-stealing worms,	ble, 432
278–279	blt, 432
inheritance, 29	bne, 432
micrimice, 27	D11C, 102

box, 432	call instruction, 431
br, 432	code samples
call, 431	counting items, 433–435
div, 432	linked lists, 436–443
ldarg, 431	details, 424
ldc, 431	div instruction, 432
ldfld,431	evaluation stack, 430
ldloc,431	ldarg instruction, 431
mul, 432	1dc instruction, 431
newarr,433	ldfld instruction, 431
newobj,433	ldloc instruction, 431
ret,431	mul instruction, 432
starg,431	.NET executables, 429
stfld,431	newarr instruction, 433
stloc,431	newobj instruction, 433
sub, 432	ret instruction, 431
switch,432	starg instruction, 431
unbox, 432	stfld instruction, 431
int 3 instruction, 331	stloc instruction, 431
integer overflows, 256–260	sub instruction, 432
Intel	switch instruction, 432
assembly language notation, 49	unbox instruction, 432
C++ Compiler version 8.0, 59-60	intermediate representations, 55–56,
LaGrande Technology Architectural	459–460
Overview, 319	internal functions, 487
NetBurst microarchitecture, 65–67	interoperability, 8, 17, 142
intellectual property, 310	interpreters, 61–62
Interactive Disassembler (IDA),	intrinsic string-manipulation func-
112–115, 121	tions, 249–250
interleaving code, 354–355	I/O system (Windows operating
Intermediate Language (IL)	system), 103–104
activation records, 430	IsDebuggerPresent Windows
add instruction, 432	API, 332–333
beg instruction, 432	
bge instruction, 432	j
bgt instruction, 432	J#, 428
ble instruction, 432	Jaffe, Joshua, "Differential Power
blt instruction, 432	Analysis", 319
bne instruction, 432	Java, 36, 423
box instruction, 432	Java Virtual Machine (JVM), 60
br instruction, 432	Jcc instructions, 51
C#, 36–37	JiTs (just-in-time compilers), 62

Johnstone, Richard, Computer Soft- ware Security System patent, 311	Kocher, Paul, "Differential Power Analysis", 319
Journal of the ACM, Self-adjusting	Kruegel, Christopher, "Static Disas-
binary search trees, Robert Endre	sembly of Obfuscated Binaries",
Tarjan and Daniel Dominic	344
Sleator, 187	Kuhn, Markus G., "Cipher Instruc-
Jun, Benjamin, "Differential Power	tion Search Attack on the Bus-
Analysis", 319	Encryption Security
just-in-time compilers (JiTs), 62	Microcontroller", 319
JVM (Java Virtual Machine), 60	wherecontroller , 317
j v ivi (java v irtuar iviacimic), 00	L
К	LaGrande Technology Architectural
kernel memory, 74	Overview, Intel, 319
kernel memory space, 75–77	last in, first out (LIFO), 40
kernel mode, 72–73	layout
kernel-mode debuggers	doubly linked lists, 553
applications, 122–123	singly linked lists, 551
defined, 117–118	stack, 539
limitations, 123	stack frames, 539
SoftICE, 124–126	trees, 554
virtual machines, 127	ldarg instruction, 431
WinDbg, 123–124	1dc instruction, 431
Key ID (Windows Media Rights	ldfld instruction, 431
Manager), 321	ldloc instruction, 431
KeygenMe-3 crackme program,	LEA instruction, 522
358–363	LEAVE instruction, 538, 540
keygenning, 364–365	legality of reverse engineering,
keywords	17–23
class,547	lexical analysis or scanning, 55
register,545	libraries, 28
static,543	library functions, 475–476
struct,547	license agreements, 23
volatile,545	licenses for software, 311
kleptographic worms, 278	Lie, David, "Architectural Support
Knuth, Donald E.	for Copy and Taper Resistant
The Art of Computer Programming —	Software", 319
Volume 2: Seminumerical Algo-	LIFO (last in, first out), 40
rithms (Second Edition), 251	linear sweep disassemblers, 337–338
The Art of Computer Programming —	line-level working-set tuning, 516,
Volume 3: Sorting and Searching	518
(Second Edition), 177, 187	linked lists, 32, 549–553
	Linux, 423

listing files, 58–59 lists, 31 live code analysis, 110 local variables, 42, 542–544 logical operators, 492–499 loops break conditions, 506–507 defined, 33 posttested, 506 pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 Jenus de data caccess, 280 Denial-of-Service (DoS) attacks, 280 information theft, 281 resource theft, 280–281 vandalism, 280 viruses, 274 vulnerabilities, 281 worms, 274–275 malloc exploits, 255–256 malware. See malicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
local variables, 42, 542–544 logical operators, 492–499 loops break conditions, 506–507 defined, 33 posttested, 506 pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M M M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 long information theft, 281 resource theft, 280–281 vandalism, 280 viruses, 274 vulnerabilities, 281 worms, 274–275 mallicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
local variables, 42, 542–544 logical operators, 492–499 loops break conditions, 506–507 defined, 33 posttested, 506 pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M M M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 long information theft, 281 resource theft, 280–281 vandalism, 280 viruses, 274 vulnerabilities, 281 worms, 274–275 mallicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
logical operators, 492–499 loops break conditions, 506–507 defined, 33 posttested, 506 pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 malicious software adware, 276–277 break conditions, 596–507 vandalism, 280 viruses, 274 vulnerabilities, 281 worms, 274–275 malloc exploits, 255–256 malware. See malicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
loops break conditions, 506–507 defined, 33 posttested, 506 pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 resource theft, 280–281 vandalism, 280 viruses, 274 vulnerabilities, 281 worms, 274–275 malloc exploits, 255–256 malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
break conditions, 506–507 defined, 33 posttested, 506 pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 malicious code, 507–508 unrolling, 508–509 malloc exploits, 255–256 malloc exploits, 250–256 malloc exploits, 255–256 malloc exploits
defined, 33 posttested, 506 pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software vulnerabilities, 281 worms, 274–275 malloc exploits, 255–256 malloc exploits, 255–256 malloc exploits, 255–256 malloc exploits, 255–256 malware. See malicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, backdoors, 276
pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software worms, 274–275 malloc exploits, 255–256 malware. See malicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, backdoors, 276
pretested, 504–506 skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software worms, 274–275 malloc exploits, 255–256 malware. See malicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, backdoors, 276
skip-cycle statements, 507–508 unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 malloc exploits, 255–256 malware. See malicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
unrolling, 508–509 Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 malware. See malicious software Malware: Fighting Malicious Code, Ed Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
Low, Douglas "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M M Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 Manufacturing Cheap, Resilient, Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
"Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 Skoudis and Lenny Zeltser, 280 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
and Stealthy Opaque Constructs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 Managed C++, 428 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
structs", 346 A Taxonomy of Obfuscating Transformations, 348 low-level data management, 37–38 low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 managed code (.NET), 426 managed code (.NET), 426 managing data high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
mations, 348 low-level data management, 37–38 low-level software, 9–10, 25 low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
mations, 348 low-level data management, 37–38 low-level software, 9–10, 25 low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 high-level, 38 lists, 31–32 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
low-level software, 9–10, 25 M machine code, 11 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 low-level, 37–38 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
machine code, 11 variables, 30 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 Christian Collberg, Clark Thomborson, and Douglas Low, 346 adware, 276–277 MacCabe software complexity metric, backdoors, 276 registers, 39 user-defined data structures, 30–31 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 machine code, 11 variables, 30 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
machine code, 11 variables, 30 Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 Christian Collberg, Clark Thomborson, and Douglas Low, 346 adware, 276–277 backdoors, 276 McCabe software complexity metric, 445
Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252 malicious software adware, 276–277 backdoors, 276 "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
and Prevention of Buffer-Overflow Attacks, 252 Christian Collberg, Clark Thomborson, and Douglas Low, 346 adware, 276–277 backdoors, 276 and Stealthy Opaque Constructs", Christian Collberg, Clark Thomborson, and Douglas Low, 346 McCabe software complexity metric, 445
Attacks, 252 Christian Collberg, Clark Thommalicious software borson, and Douglas Low, 346 adware, 276–277 McCabe software complexity metric, backdoors, 276 445
malicious software borson, and Douglas Low, 346 adware, 276–277 McCabe software complexity metric, backdoors, 276 445
adware, 276–277 McCabe software complexity metric, backdoors, 276 445
backdoors, 276 445
•
7700/0
BIOS/firmware, 279–280 MD5 cryptographic hashing algo-
defined, 5–6, 273 rithm, 213
deleting, 277 media-based copy protection tech-
information-stealing worms, nologies, 314–316
278–279 Memon, Nasir, "Protecting Digital
metamorphism, 283–285 Media Content", 322
mobile code, 276 memory management in Windows
mobile code, 276 memory management in Windows polymorphism, 282–283 kernel memory, 74–75
mobile code, 276 memory management in Windows polymorphism, 282–283 kernel memory, 74–75 spyware, 276–277 kernel memory space, 75–77
mobile code, 276 memory management in Windows polymorphism, 282–283 kernel memory, 74–75 spyware, 276–277 kernel memory space, 75–77 Trojan/Backdoor.Hacarmy.D page faults, 73–74
mobile code, 276 memory management in Windows polymorphism, 282–283 kernel memory, 74–75 spyware, 276–277 kernel memory space, 75–77 Trojan/Backdoor.Hacarmy.D program, 285–305 paging, 73
mobile code, 276 memory management in Windows polymorphism, 282–283 kernel memory, 74–75 spyware, 276–277 kernel memory space, 75–77 Trojan/Backdoor.Hacarmy.D page faults, 73–74

memory management in Windows	mul instruction, 432
(continued)	.NET executables, 429
user-mode allocations, 78-79	newarr instruction, 433
VAD (Virtual Address Descriptor)	newobj instruction, 433
tree, 78	ret instruction, 431
virtual memory, 72–73	starg instruction, 431
Virtual Memory Manager, 79–80	stfld instruction, 431
working sets, 74	stloc instruction, 431
memory mapped files, 78	sub instruction, 432
metadata (.NET), 426	switch instruction, 432
metamorphism, 283–285	unbox instruction, 432
methodologies of reversing, 110	Microsoft (MS)
methods, 556–557	C/C++ Optimizing Compiler ver-
microcode, 65	sion 13.10.3077, 59
Microprocessor for Executing Enciphered	cryptographic service providers
Programs patent, Robert M. Best,	(CSPs), 207
311, 318	DUMPBIN executable-dumping
microprocessors, 63–68	tool, 133–136
Microsoft Intermediate Language	IIS Indexing Service Vulnerability
(MSIL)	262–271
activation records, 430	ILDasm, 115–116
add instruction, 432	Next-Generation Secure Comput-
beq instruction, 432	ing Base (NGSCB), 323-324
bge instruction, 432	Virtual PC, 128
bgt instruction, 432	WinDbg debugger, 119-121,
ble instruction, 432	123–124
blt instruction, 432	Microsoft .NET platform
bne instruction, 432	assemblies, 426, 453
box instruction, 432	C# programming language, 428
br instruction, 432	class library, 426
C#, 36–37	Common Intermediate Language
call instruction, 431	(CIL), 429
code samples	Common Language Runtime
counting items, 433–435	(CLR), 426–427
linked lists, 436–443	Common Type System (CTS),
details, 424	428–429
div instruction, 432	comparison with Java, 423
evaluation stack, 430	compilation stages, 429
ldarg instruction, 431	decompilers, 424-425, 443
1dc instruction, 431	IL (Intermediate Language), 424,
ldfld instruction, 431	429–430
1dloc instruction, 431	J# programming language, 428

N	n-way conditionals, 33, 499–500,
Nagra, J., "A Functional Taxonomy	502-504
for Software Watermarking", 322	
named objects, 81-83	0
native API, 90–91	OBFUSCATE macro, 343–344
native code decompilers, 457–459	obfuscation, 328-329, 344-345
Nebbett, Gary, Windows NT/2000	obfuscators
Native API Reference, 91, 389	defined, 63
.NET	DotFuscator, 444, 448–451
assemblies, 426, 453	.NET, 424, 444–455
C# programming language, 428	Remotesoft Obfuscator, 451-452
class library, 426	Remotesoft Protector, 452–455
Common Intermediate Language	Spices.Net, 444
(CIL), 429	XenoCode, 444, 446–447
Common Language Runtime	object code, 11
(CLR), 426–427	object-oriented design (OOD), 29
Common Type System (CTS),	objects
428–429	base object, 29
comparison with Java, 423	clients, 29
compilation stages, 429	defined, 29
decompilers, 424–425, 443	inheritance, 29
IL (Intermediate Language), 424,	named objects, 81-83
429–430	object-oriented design (OOD), 29
J# programming language, 428	polymorphism, 29, 35
Managed C++ programming lan-	Windows operating system, 80–83
guage, 428	OF (overflow flag), 520–521
managed code, 426	offline code analysis, 110
metadata, 426	OllyDbg debugger, 118–120
.NET Framework environment, 426	OOD (object-oriented design), 29
obfuscators, 424, 444–455	opaque predicates, 338-340, 346-347
Visual Basic .NET programming	opcode (operation code), 11, 47
language, 428	operand comparison, 50
NetBurst microarchitecture, 65–67	operands
networking protocols, 202	comparing, 480–483
newarr instruction, 433	instructions, 47–48
newobj instruction, 433	signed, 480–481
Next-Generation Secure Computing	unsigned, 482–483
Base (NGSCB), 323–324	operating systems
nonexecutable memory, 254–255	defined, 13
NtQuerySystemInformation	Windows
native API, 333–334	application programming inter-
NuMega SoftICE debugger, 124–126,	faces (APIs), 88–91
334	architecture, 70–71

compatibility, 71	VAD (Virtual Address Descriptor)
context switching, 85–86	tree, 78
critical sections, 87	virtual memory, 70, 72
directories, 83	Virtual Memory Manager, 79–80
dispatcher, 84	Win32 subsystem, 104–105
dynamically linked libraries	working sets, 74
(DLLs), 96–97	operation code (opcode), 11, 47
events, 86	operators, 492–499
exception handlers, 105–107	optimizers (compilers), 56–57
exceptions, 105–107	OR logical operator, 492, 494–498
executable formats, 93–102	ordering transformations, 346, 355
features, 70–71	outlining, 353
handles, 81	overflow bugs
history, 70	heap overflows, 255–256
I/O system, 103–104	integer overflows, 256–260
kernel memory, 74	stack overflows, 245–255
kernel memory space, 75–77	string filters, 256
kernel mode, 72–73	overflow flag (OF), 520–521
multiprocessor capability, 71	
multithreaded, 71	P
mutexes, 87	page faults, 73–74
object manager, 80–81	page tables (virtual memory), 72
objects, 80–83	pagefile-backed section object, 78
page faults, 73–74	pages (virtual memory), 72
paging, 73	paging, 73
portability, 71	parity flag (PF), 521
process initialization sequence,	password verification process
87–88	"Bad Password" message, 207–210
processes, 84	hashing the password, 213–218
scheduler, 84	password transformation algo-
section objects, 77–78	rithm, 210–213
security, 71	patching
semaphores, 87	Hex Workshop, 131–132
64-bit versions, 71–72	KeygenMe-3 crackme program,
supported hardware, 71	358–363
synchronization objects, 86–87	patents, 20, 311, 318
system calling mechanism, 91–93	PE (Portable Executable)
32-bit versions, 71–72	directories, 99–102
threads, 84–85	exports, 99
user memory, 74	file alignment, 95
user mode, 72–73	headers, 97–98
user-mode allocations, 78–79	image sections, 95

PE (Portable Executable) (continued)	PortMon system-monitoring tool, 130
imports, 99 relative virtual address (RVA), 95	posttested loops, 506
relocations, 93–95	power usage analysis attacks, 319
section alignment, 95–96	precompiled assemblies (.NET), 453
PEBrowse Professional Interactive	± ±
	PreEmptive Solutions DotFuscator
debugging, 122	obfuscator, 444, 448–451
executable dumping, 137–138	pretested loops, 504–506
PEID program, 376–377	primitive data types, 472–473
PEView executable-dumping tool, 137	procedures
	alldiv,530-534
PF (parity flag), 521	allmul,530
Phrack paper, Aleph1, 245	calling, 487
pipelines, 65–67	Cryptex command-line data
piracy	encryption tool, 205–207
class breaks, 312–313	defined, 486
copy protection schemes, 313	epilogues, 486
copy protection technologies,	(, 468
311–313	imported, 487–488
copyrights, 309–310	internal, 487
digital rights management (DRM), 319–321	intrinsic string-manipulation, 249–250
intellectual property, 310	library, 475–476
magnitude of, 309	prologues, 486
software, 310–311	RtlDeleteElementGener-
software piracy, 312	icTable, 193–194
trusted computing, 322–324	RtlGetElementGenericTable
watermarking, 321–322	disassembly, 153–155
polymorphism, 29, 35, 282–283	initialization, 155–159
portability of Windows operating	logic and structure, 159–161
system, 71	search loop 1, 161–163
Portable Executable (PE)	search loop 2, 163–164
directories, 99–102	
	search loop 4, 165
exports, 99	search loop 4, 165
file alignment, 95	setup, 155–159
headers, 97–98	source code, 165–168
image sections, 95	RtlInitializeGenericTable,
imports, 99	146–151
relative virtual address (RVA), 95	RtlInsertElementGener-
relocations, 93–95	icTable, 168–170
section alignment, 95–96	RtlIsGenericTableEmpty, 152-153

RtlLocateNodeGenericTable,	prologues in functions, 486
170–178	proprietary software, 7–8
RtlLookupElementGener-	"Protecting Digital Media Content",
icTable, 188–193	Nasir Memon and Ping Wah
RtlNumberGenericTableEle-	Wong, 322
ments, 151-152	protection technologies
RtlRealInsertElementWorker,	attacks, 324
178–186	challenge response, 315–316
RtlSplay, 185–188	class breaks, 312–313
Process Explorer system-monitoring	cracking, 357–358
tool, 130–131	crypto-processors, 318–319
process initialization sequence,	Defender crackme program,
87–88	415–416
processes, 84	dongle, 316–317
program data	encryption, 318 hardware-based, 316–317
program data defined, 537	media-based, 314–316
stack	objectives, 312
defined, 538	online activation, 315–316
layout, 539	requirements, 313
stack frames	ripping algorithms, 365–370
defined, 538	serial numbers, 315
ENTER instruction, 538–540	server-based software, 317
layout, 539	StarForce suite (StarForce Tech-
LEAVE instruction, 538, 540	nologies), 345
program structure	trusted components, 312
control flow	Uncrackable Model, 314
conditional blocks, 32	Protector (Remotesoft), 452–455
defined, 32	Pu, Calton, Automatic Detection and
loops, 33	Prevention of Buffer-Overflow
switch blocks, 33	Attacks, 252
data management, 29–32	pure arithmetic, 510–512
defined, 26–27	,
encapsulation, 27	R
modules, 28	reciprocal multiplication, 524–527
objects, 29	recursive traversal disassemblers,
procedures, 28	338–343
programming languages	redundancy elimination, 57
C, 34–35	register keyword, 545
C#, 36–37, 428	register transfer languages (RTL),
C++, 35	468
Java, 36, 423	register values, 42
.NET, 428	

registers	malicious software, 5–6
defined, 39, 44–45	proprietary software, 7–8
EAX, 45–46	software development, 8–9
EBP, 45–46	system-level reversing, 13–14
EBX, 45–46	reversing tools
ECX, 45-46	Cryptex command-line data
EDI, 45-46	encryption tool, 200, 202
EDX, 45–46	debuggers, 15–16, 116–126
EFLAGS, 46, 519-520	decompilers, 16, 129
ESI, 45-46	disassemblers, 15, 110–116
ESP, 45-46	executable dumping, 133–138
RegMon system-monitoring tool,	patching, 131–132
130	system monitoring, 15, 129–130
relative virtual address (RVA), 95	ripping algorithms, 365–370
Remotesoft	RTL (register transfer languages),
Obfuscator, 451–452	468
Protector, 452–455	RtlDeleteElementGener-
resource theft, 280-281	icTable function, 193-194
restructuring arrays, 356	RtlGetElementGenericTable
RET instruction, 51, 540	function
ret instruction, 431	disassembly, 153-155
Reverse Compilation Techniques,	initialization, 155–159
Christina Cifuentes, 477	logic and structure, 159–161
reverse engineering	search loop 1, 161–163
applications, 4–5	search loop 2, 163–164
code-level reversing, 13–14	search loop 3, 164–165
competing software, 8-9, 18-19	search loop 4, 165
data reverse engineering	setup, 155–159
Cryptex command-line data	source code, 165–168
encryption tool, 200–202	RtlInitializeGenericTable
defined, 199	function, 146–151
file formats, 202–204	RtlInsertElementGener-
Microsoft Word file format, 200	icTable function, 168–170
networking protocols, 202	RtlIsGenericTableEmpty func
uses, 199–200	tion, 152–153
defined, 3–4	RtlLocateNodeGenericTable
ground rules, 142–143	function, 170–178
legality, 17–23	RtlLookupElementGener-
live code analysis, 110	icTable function, 188-193
offline code analysis, 110	RtlNumberGenericTableEle-
security-related	ments function, 151-152
cryptographic algorithms, 6	RtlRealInsertElementWorker
digital rights management	function, 178–186
(DRM), 7	

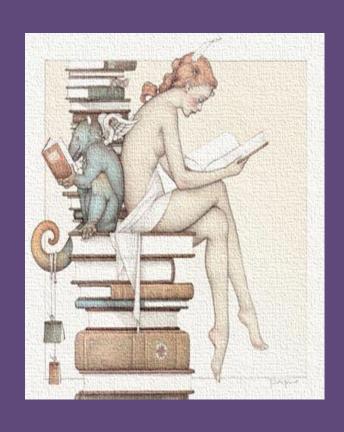
Rt1Splay function, 185–188	signed operands, 480–481
RVA (relative virtual address), 95	single static assignment (SSA), 467–468
S	single-branch conditionals, 488-489
SBB instruction, 529	single-stepping, 16
scheduler (Windows operating sys-	singly linked lists, 550–552
tem), 84	64-bit arithmetic, 528–534
Schneier, Bruce, Applied Cryptogra-	64-bit versions of Windows, 71–72
phy, Second Edition, 312, 415	skip-cycle statements in loops,
Schwarz, Benjamin, Disassembly of	507–508
Executable Code Revisited, 111	Sklyarov, Dmitry (Russian program
SDMI (Secure Digital Music Initia-	mer), 22
tive), 22	Skoudis, Ed, Malware: Fighting Mali
searching, 32	cious Code, 280
section objects, 77–78	Sleator, Daniel Dominic, Self-adjust-
Secure Audio Path, 321	ing binary search trees, Journal of
Secure Digital Music Initiative	the ACM (JACM), 187
(SDMI), 22	SoftICE debugger, 124–126, 334
security	software
defined, 243–244	anti-reverse-engineering
Windows operating system, 71	clauses, 23
security-related reverse engineering	assembly language, 10–11
cryptographic algorithms, 6	bytecodes, 12–13
digital rights management	competing software, 8–9, 18–19
(DRM), 7	compilers, 11–12
malicious software, 5–6	copy protection schemes, 313
proprietary software, 7–8	interoperability, 8, 17
Sega Enterprises, 18	license agreements, 23
self-adjusting binary search trees,	low-level, 9–10, 25
187–191	malicious, 5–6, 273–277
Self-adjusting binary search trees, Jour-	operating systems, 13
nal of the ACM (JACM), Robert	system, 9–10
Endre Tarjan and Daniel Dominic	Uncrackable Model, 314
Sleator, 187	virtual machines, 12–13
semaphores, 87	software development, 8–9
serial numbers, 315	software exceptions, 105
server-based software, 317	software licenses, 311
Set Byte on Condition (SETcc),	software piracy, 310–312
513–514	software watermarking, 322
sign extending, 535	Spices.Net obfuscator, 444
sign flag (SF), 521	splay tables, 187–191
signed conditional codes, 483–485	spyware, 276–277

SSA (single static assignment),	classes
467–468	constructors, 559–560
stack	data members, 555–556
defined, 40, 538	defined, 555
function calls, 42	inherited classes, 555-556
layout, 539	methods, 556-557
LIFO (last in, first out), 40	virtual functions, 557-560
local variables, 42	defined, 547
pop operations, 41	generic data structures, 547–548
push operations, 41	linked lists, 32, 549–553
register values, 42	lists, 31
stack checking, 250–254	trees, 32, 552, 554
stack frames	user-defined data structures, 30-31
defined, 538	variables, 30
ENTER instruction, 538–540	SUB instruction, 49–50, 522, 529
layout, 539	sub instruction, 432
LEAVE instruction, 538, 540	switch blocks, 33, 499-504
stack overflows, 245–255	switch instruction, 432
StarForce suite (StarForce Technolo-	symbolic information, 328–330
gies), 345	symbolic link directory, 83
starg instruction, 431	synchronization objects, 86-87
"Static Disassembly of Obfuscated	SYSENTER instruction, 394
Binaries", Christopher Kruegel, et	system calling mechanism (Win-
al., 344	dows operating system), 91–93
static keyword, 543	system flags, 46–47
static libraries, 28	system software, 9–10
status flags, 46–47	system-level reversing, 13–14
stdcall calling convention, 541	system-monitoring tools
stfld instruction, 431	defined, 15, 129–130
stloc instruction, 431	FileMon, 130
Strategies to Combat Software Piracy,	PortMon, 130
Jayadeve Misra, 312	Process Explorer, 130–131
string filters, 256	RegMon, 130
StrongBit Technology EXECryptor,	TCPView, 130
345	TDIMon, 130
struct keyword, 547	WinObj, 130
structured exception handling,	
105–106	T
structures for data	table API
alignment, 547–548	callbacks prototypes, 195
arrays, 31, 548–549	definition, 145-146, 194-196
	function prototypes, 196

internal data structures, 195 hardware-based, 316–317 RtlDeleteElementGenermedia-based, 314–316 icTable function, 193-194 objectives, 312 RtlGetElementGenericTable online activation, 315–316 function, 153–168 requirements, 313 RtlInitializeGenericTable ripping algorithms, 365–370 function, 146–151 serial numbers, 315 server-based software, 317 RtlInsertElementGenericTable function, 168-170 StarForce suite (StarForce Tech-RtlIsGenericTableEmpty nologies), 345 trusted components, 312 function, 152–153 Uncrackable Model, 314 RtlLocateNodeGenericTable function, 170–178 32-bit versions of Windows, 71–72 RtlLookupElementGenerthiscall calling convention, 541 icTable function, 188-193 Thomborson, Clark RtlNumberGenericTableEle-"A Functional Taxonomy for Softments function, 151-152 ware Watermarking", 322 RtlRealInsertElementWorker "Manufacturing Cheap, Resilient, function, 178–186 and Stealthy Opaque Constructs", 346 RtlSplay function, 185–188 table interpretation, 348–353 A Taxonomy of Obfuscating Transformations, 348 Tarjan, Robert Endre, Self-adjusting binary search trees, Journal of the thread information block (TIB), 106 ACM (JACM), 187 thread-local storage (TLS), 546–547 A Taxonomy of Obfuscating Transforthreads, 84–85 mations, Christian Collberg, Clark 3DES encryption algorithm, 200 Thomborson, and Douglas Low, tools 348 Cryptex command-line data TCPView system-monitoring tool, encryption tool, 200, 202 130 debuggers, 15–16, 116–126 TDIMon system-monitoring tool, decompilers, 16, 129 130 disassemblers, 15, 110–116 technologies for copy protection executable dumping, 133–138 attacks, 324 patching, 131–132 challenge response, 315–316 system monitoring, 15, 129–130 class breaks, 312–313 Torczon, Linda, Engineering a Comcracking, 357–358 piler, 54 crypto-processors, 318–319 trade secrets, 20 Defender crackme program, Transcopy copy protection technol-415–416 ogy, 314 dongle, 316–317 trap flag, 335 trees, 32, 552, 554 encryption, 318

Trojan horses, 275	virtual machines
trusted computing, 322–324	bytecodes, 12–13, 60–63
tuning working sets	debugging, 127–128
function-level, 515-517	Virtual Memory Manager, 79–80
line-level, 516, 518	virtual memory (Windows operat-
two-way conditionals, 489–490	ing system), 70, 72
type conversion errors, 260–262	Virtual PC (Microsoft), 128
type conversions	viruses, 274
defined, 534	Visual Basic .NET, 428
sign extending, 535	VMWare Workstation, 128
zero extending, 534–535	volatile keyword, 545
0	vulnerabilities
U	defined, 245
unbox instruction, 432	heap overflows, 255–256
Uncrackable Model, 314	IIS Indexing Service Vulnerability,
undocumented APIs, 142-144	262–271
unrolling loops, 508–509	integer overflows, 256–260
unsigned conditional codes, 485–486	intrinsic string-manipulation func-
unsigned operands, 482–483	tions, 249–250
US vs. Sklyarov case, 22	malicious software, 281
user memory, 74	stack overflows, 245–255
user mode, 72–73	string filters, 256
user-defined data structures, 30–31	type conversion errors, 260–262
user-mode debuggers, 117–122	,
00 ,	W
V	Wagle, Perry, Automatic Detection and
VAD (Virtual Address Descriptor)	Prevention of Buffer-Overflow
tree, 78	Attacks, 252
vandalism, 280	watermarking, 321–322
variables	Win32 API, 88–90
defined, 30	Win32 subsystem, 104–105
global variables, 542	WinDbg debugger
imported variables, 544–546	command-line interface, 119
local variables, 542–544	disassembler, 119
verification process for passwords	extensions, 129
"Bad Password" message, 207–210	features, 119
hashing the password, 213–218	improvements, 121
password transformation algo-	kernel-mode, 123–124
rithm, 210–213	user-mode, 119–121
Virtual Address Descriptor (VAD)	Windows APIs
tree, 78	generic table API, 145–146
virtual functions, 557–560	IsDebuggerPresent, 332-333
,	undocumented APIs, 142–144

Windows Media Rights Manager,	supported hardware, 71
321	synchronization objects, 86–87
Windows NT/2000 Native API Refer-	system calling mechanism, 91–93
ence, Gary Nebbett, 91, 389	32-bit versions, 71–72
Windows operating system	threads, 84–85
application programming inter-	user memory, 74
faces (APIs), 88–91	user mode, 72–73
architecture, 70–71	user-mode allocations, 78–79
compatibility, 71	VAD (Virtual Address Descriptor)
context switching, 85–86	tree, 78
critical sections, 87	virtual memory, 70, 72
directories, 83	Virtual Memory Manager, 79–80
dispatcher, 84	Win32 subsystem, 104–105
dynamically linked libraries	working sets, 74
(DLLs), 96–97	WinObj system-monitoring tool, 130
events, 86	Wong, Ping Wah, "Protecting Digital
exception handlers, 105–107	Media Content", 322
exceptions, 105–107	working sets, 74
executable formats, 93–102	working-set tuning
features, 70–71	function-level, 515-517
handles, 81	line-level, 516, 518
history, 70	worms
I/O system, 103–104	Code Red Worm, 262
kernel memory, 74	defined, 274–275
kernel memory space, 75–77	information-stealing worms,
kernel mode, 72–73	278–279
multiprocessor capability, 71	Wroblewski, Gregory, General
multithreaded, 71	Method of Program Code Obfusca-
mutexes, 87	tion, 347
object manager, 80–81	
objects, 80–83	X
page faults, 73–74	XenoCode obfuscator, 444, 446-447
paging, 73	XOR algorithm, 416
portability, 71	
process initialization sequence,	Z
87–88	Zeltser, Lenny, Malware: Fighting
processes, 84	Malicious Code, 280
scheduler, 84	zero extending, 534–535
section objects, 77–78	zero flag (ZF), 521
security, 71	Zhang, Qian, Automatic Detection and
semaphores, 87	Prevention of Buffer-Overflow
64-bit versions, 71–72	Attacks, 252



##