Symbols & Numerics
(-functions, 468
32-bit versions of Windows, 71–72
64-bit arithmetic, 528–534
64-bit versions of Windows, 71–72
3DES encryption algorithm, 200

A
Accolade game developer, 18
activation records (MSIL), 430
ADC instruction, 529
ADD instruction (IA-32)
  configuration, 49–50
  operands, 522
  64-bit integers, 529
add instruction (MSIL), 432
address spaces, 72
Advanced Compiler Design and Implementation, Steven S. Muchnick, 54
adware, 276–277
aggregation transformations, 346
Aleph1, 245
algorithms
  binary search algorithm, 177
  Cipher Block Chaining (CBC), 415
cryptographic, 6
DES (Data Encryption Standard) algorithm, 200
MD5 cryptographic hashing algorithm, 213
password transformation algorithm, 210–213
ripping, 365–370
3DES encryption algorithm, 200
XOR algorithm, 416
alignment of data structures, 547–548
alldiv function, 530–534
allmul function, 530
AND logical operator, 492–493, 498–499
Andrews, Gregory, Disassembly of Executable Code Revisited, 111
Andromeda IA-32 decompiler, 477
anti-reverse-engineering clauses, 23
antireversing
  antidebugger code, 329, 331–336
  benefits, 327–328
  control flow transformations, 346
decompliers, 348
disassemblers, 336–343
cryptography, 330
antireversing (continued)
inlining, 353
interleaving code, 354–355
OBFUSCATE macro, 343–344
obfuscation, 328–329, 344–345
opaque predicates, 346–347
outlining, 353
symbolic information, 328–330
table interpretation, 348–353
APIs (application programming interfaces)
defined, 88
generic table API
callbacks prototypes, 195
definition, 145–146, 194–196
function prototypes, 196
internal data structures, 195
RtlDeleteElementGenericTable function, 193–194
RtlGetElementGenericTable function, 153–168
RtlInitializeGenericTable function, 146–151
RtlInsertElementGenericTable function, 168–170
RtlIsGenericTableEmpty function, 152–153
RtlLocateNodeGenericTable function, 170–178
RtlLookupElementGenericTable function, 188–193
RtlNumberGenericTableElements function, 151–152
RtlRealInsertElementWorker function, 178–186
RtlSplay function, 185–188
IsDebuggerPresent Windows API, 332–333
native API, 90–91
NtQuerySystemInformation native API, 333–334
undocumented Windows APIs, 142–144
Win32 API, 88–90
Apple Macintosh, 423
applications of reverse engineering, 4–5
Applied Cryptography, Second Edition,
Bruce Schneier, 312, 415
“Architectural Support for Copy and Taper Resistant Software”,
David Lie et al., 319
architecture
compilers, 55–58
decompilers, 459
Windows operating system, 70–71
arithmetic flags
carry flag (CF), 520–521
defined, 519
EFLAGS register, 519–520
overflow flag (OF), 520–521
parity flag (PF), 521
sign flag (SF), 521
zero flag (ZF), 521
arithmetic operations
ADC instruction, 529
ADD instruction, 522, 529
DIV/IDIV instruction, 524
LEA instruction, 522
modulo, 527–528
MUL/IMUL instruction, 523–524
reciprocal multiplication, 524–527
SBB instruction, 529
64-bit arithmetic, 528–534
SUB instruction, 522, 529
arithmetic (pure), 510–512
array restructuring, 356
arrays, 31, 548–549
The Art of Computer Programming —
Volume 2: Seminumerical Algorithms (Second Edition), Donald E. Knuth, 251
The Art of Computer Programming —
Volume 3: Sorting and Searching (Second Edition), Donald E. Knuth, 177, 187
assembler program, 11
assemblies (.NET), 426, 453
assembly language
  AT&T Unix notation, 49
code examples, 52–53
defined, 10–11, 44
flags, 46–47
instructions, 47–51
Intel notation, 49
machine code, 11
operation code (opcode), 11
platforms, 11
registers, 44–46
AT&T Unix assembly language
  notation, 49
attacks
  copy protection technologies, 324
  DoS (Denial-of-Service) attacks, 280
  power usage analysis attacks, 319
audio, 321
Automatic Detection and Prevention of Buffer-Overflow Attacks, Crispin Cowan, Calton Pu, David Maier, Heather Hinton, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang, 252

B
back end of decompilers, 476–477
backdoor access (with malicious software), 280
backdoors, 276
Bakke, Peat, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252
base object, 29
BaseNamedObjects directory, 83
basic block (BB), 464–466
Beattie, Steve, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252
beq instruction, 432
Best, Robert M., Microprocessor for Executing Enciphered Programs
  patent, 311, 318
bge instruction, 432
bgt instruction, 432
binary code, 11
binary file comparison programs, 242
binary search algorithm, 177
binary searching, 32
binary trees, 32, 552, 554
BIOS/firmware malware, 279–280
ble instruction, 432
blt instruction, 432
bne instruction, 432
Boomerang IA-32 decompiler, 477
box instruction, 432
br instruction, 432
branch prediction, 67–68
branchless logic
  conditional instructions, 513–515
defined, 509
pure arithmetic, 510–512
break conditions in loops, 506–507
breaking copy protection technologies attacks, 324
challenge response, 315–316
class breaks, 312–313
cracking, 357–358
crypto-processors, 318–319
Defender crackme program, 415–416
dongle, 316–317
encryption, 318
hardware-based, 316–317
media-based, 314–316
objectives, 312
online activation, 315–316
requirements, 313
ripping algorithms, 365–370
serial numbers, 315
breaking copy protection
technologies (continued)
server-based software, 317
StarForce suite (StarForce Technologies), 345
trusted components, 312
Uncrackable Model, 314
breakpoint interrupt, 331
BreakPoint Software Hex Workshop, 131–132
breakpoints, 331–332
brute-forcing the Defender crackme program, 409–414
BSA and IDC Global Software Piracy Study, Business Software Alliance and IDC, 310
bugs (overflows)
heap overflows, 255–256
integer overflows, 256–260
stack overflows, 245–255
string filters, 256
Business Software Alliance, BSA and IDC Global Software Piracy Study, 310
Byte magazine, 311
bytecodes
defined, 12
difference from binary code, 61
interpreters, 61–62
just-in-time compilers (JiTs), 62
reversing strategies, 62–63
virtual machines, 12–13, 61
C
C programming language, 34–35
C# programming language, 36–37, 428
C++ programming language, 35
CALL instruction, 51, 487, 540
call instruction, 431
calling conventions
cdecl, 540
defined, 540
fastcall, 541
stdcall, 541
thiscall, 541
calling functions, 487
carry flag (CF), 520–521
cases
Felten vs. RIAA, 22
US vs. Sklyarov, 22
CBC (Cipher Block Chaining), 415
cdecl calling convention, 540
CDQ instruction, 535
CF (carry flag), 520–521
CFGs (control flow graphs), 462
challenge response, 315–316
checksums, 335–336
Cifuentes, Christina, Reverse Compilation Techniques, 477
CIL (Common Intermediate Language). See Common Intermediate Language (CIL)
Cipher Block Chaining (CBC), 415
“Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller”, Markus G. Kuhn, 319
class breaks, 312–313
class keyword, 547
class library (.NET), 426
classes
constructors, 559–560
data members, 555–556
defined, 555
inherited classes, 555–556
methods, 556–557
virtual functions, 557–560
CLR (Common Language Runtime), 36, 60, 426–427
CMOVcc (Conditional Move), 514–515
CMP instruction, 50, 480–483
code
analysis with decompilers, 466–468
compiler-generated, 53–54
constructs, 28–29
code checksums, 335–336
code interleaving, 354–355
Code Red Worm, 262
code-level reversing, 13–14
Collberg, Christian
“A Functional Taxonomy for Software Watermarking”, 322
“Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs”, 346
A Taxonomy of Obfuscating Transformations, 348
Common Intermediate Language (CIL)
activation records, 430
add instruction, 432
beq instruction, 432
bge instruction, 432
bgt instruction, 432
ble instruction, 432
blt instruction, 432
bne instruction, 432
box instruction, 432
br instruction, 432
C#, 36–37
call instruction, 431
code samples
    counting items, 433–435
    linked lists, 436–443
details, 424
div instruction, 432
evaluation stack, 430
ldarg instruction, 431
ldc instruction, 431
ldfld instruction, 431
ldloc instruction, 431
mul instruction, 432
.NET executables, 429
newarr instruction, 433
newobj instruction, 433
ret instruction, 431
starg instruction, 431
stfld instruction, 431
stloc instruction, 431
sub instruction, 432
switch instruction, 432
unbox instruction, 432
Common Language Runtime (CLR), 36, 60, 426–427
Common Type System (CTS), 428–429
comparing operands, 50, 480–483
competing software, 8–9, 18–19
compilation
    lexical analysis or scanning, 55
    redundancy elimination, 57
compiler-generated code, 53–54
compilers
    architecture, 55–58
    bytecodes, 12
    compiler-readable form, 458
    defined, 11–12, 54
    GCC and G++ version 3.3.1, 59
    Intel C++ Compiler version 8.0, 59–60
    intermediate representations, 55–56
    just-in-time compilers (JiTs), 62
    listing files, 58–59
    Microsoft C/C++ Optimizing Compiler version 13.10.3077, 59
    optimizations, 54, 56–57
complex data types, 473–474
compound conditionals, 491–492
computation transformations, 346
Computer Software Security System patent, Richard Johnstone, 311
conditional blocks, 32
conditional branches, 51
conditional codes
    signed, 483–485
    unsigned, 485–486
conditional instructions, 513–515
Conditional Move (CMOVcc), 514–515
conditionals
  compound, 491–492
  logical operators, 492–499
loops
  break conditions, 506–507
  posttested, 506
  pretested, 504–506
  skip-cycle statements, 507–508
  unrolling, 508–509
multiple-alternative, 490–491
  single-branch, 488–489
switch blocks, 499–504
two-way, 489–490
constants, 546
constructors, 559–560
constructs for data
  constants, 546
  global variables, 542
  imported variables, 544–546
  local variables, 542–544
  thread-local storage (TLS), 546–547
context switching, 85–86
control flow
  conditional blocks, 32
  defined, 32
  loops, 33
  low-level implementation, 43–44
  switch blocks, 33
control flow analysis, 475
control flow graphs (CFGs), 462
control flow transformations, 346–347
conventions for calls
cdecl, 540
defined, 540
fastcall, 541
stdcall, 541
thiscall, 541
Copper, Keith D., Engineering a Compiler, 54
copy protection technologies
attacks, 324
challenge response, 315–316
class breaks, 312–313
cracking, 357–358
crypto-processors, 318–319
Defender crackme program, 415–416
dongle, 316–317
encryption, 318
hardware-based, 316–317
media-based, 314–316
objectives, 312
online activation, 315–316
requirements, 313
ripping algorithms, 365–370
serial numbers, 315
server-based software, 317
StarForce suite (StarForce Technologies), 345
trusted components, 312
Uncrackable Model, 314
copyright laws, 19
copyrights, 309–310
CopyWrite copy protection technology, 314
Cowan, Crispin, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252
cracking
class breaks, 312–313
defined, 309, 357–358
keygenning, 364–365
patching, 358–363
ripping algorithms, 365–370

Crackmes
Defender
  brute-forcing, 409–415
copy protection technologies, 415–416
decrypted code analysis, 387–395
decryption keys, 418–419
disappearance of SoftICE, 396
DUMPBIN, 372–376
Executable Modules window, 371–372
generic usage message, 370–371
initialization routine reversal, 377–387
inlining, 419
KERNEL32.DLL, 400–404
“killer” thread, 399–400
obfuscated interface, 416–417
parameter parsing, 404–406
PEiD program, 376–377
processor time-stamp verification thread, 417–418
running, 370
secondary thread reversal, 396–399
16-digit hexadecimal serial numbers, 371
usernames, 371, 406–407
validating user information, 407–408
defined, 358
finding, 420
KeygenMe-3, 358–363
critical sections, 87
.crx file format, 202–204
Cryptex command-line data encryption tool
clusters, 239–241
commands, 202
decrypting files, 235–236
decryption loop, 238–239
directory layout
directory processing code, 218–223
dumping, 227
file entries, 223–227
file decryption and extraction routine, 228–233
file entry format, 241
floating-point sequence, 236–238
functions, 205–207
header, 240
holes, 241
password verification process “Bad Password” message, 207–210
hashing the password, 213–218
password transformation algorithm, 210–213
scanning the file list, 234–235
3DES encryption algorithm, 200
verifying hash values, 239
welcome screen, 201
Windows Crypto API, 206–207
cryptographic service providers (CSPs), 207
cryptography algorithms, 6
information-stealing worms, 278
trusted computing, 322–324
crypto-processors, 318–319
CSPs (cryptographic service providers), 207
CTS (Common Type System), 428–429

data constructs
constants, 546
global variables, 542
imported variables, 544–546
local variables, 542–544
thread-local storage (TLS), 546–547
Data Encryption Standard (DES) algorithm, 200
data encryption tool
clusters, 239–241
commands, 202
decrypting files, 235–236
decryption loop, 238–239
directory layout
directory processing code, 218–223
dumping, 227
file entries, 223–227
data encryption tool (continued)
file decryption and extraction routine, 228–233
file entry format, 241
floating-point sequence, 236–238
functions, 205–207
header, 240
holes, 241
password verification process
“Bad Password” message, 
207–210
hashing the password, 213–218
password transformation algorithm, 210–213
scanning the file list, 234–235
3DES encryption algorithm, 200
verifying hash values, 239
welcome screen, 201
Windows Crypto API, 206–207
data management
defined, 29–30
high-level, 38
lists, 31–32
low-level, 37–38
registers, 39
user-defined data structures, 30–31
variables, 30
data members (classes), 555–556
data (programs)
defined, 537
stack
defined, 538
layout, 539
stack frames
defined, 538
ENTER instruction, 538–540
layout, 539
LEAVE instruction, 538, 540
data reverse engineering
Cryptex command-line data encryption tool, 200–202
defined, 199
file formats, 202–204
Microsoft Word file format, 200
networking protocols, 202
uses, 199–200
data structure arrays, 549
data structures
alignment, 547–548
arrays, 31, 548–549
classes
constructors, 559–560
data members, 555–556
defined, 555
inherited classes, 555–556
methods, 556–557
virtual functions, 557–560
defined, 547
generic data structures, 547–548
linked lists, 32, 549–553
lists, 31
trees, 32, 552, 554
user-defined data structures, 30–31
variables, 30
data transformations, 355–356
data type conversions
defined, 534
sign extending, 535
zero extending, 534–535
data types
complex, 473–474
primitive, 472–473
data-flow analysis
data propagation, 468–470
data type propagation, 471–474
defined, 466–467
register variable identification,
470–471
single static assignment (SSA),
467–468
DataRescue Interactive Disassembler (IDA), 112–115
dead-listing, 110
Debray, Saumya, *Disassembly of Executable Code Revisited*, 111

debuggers

- breakpoint interrupt, 331
- breakpoints, 15–16, 331–332
- code checksums, 335–336
- defined, 15–16, 116
- detecting, 334–336
- features, 117

- hardware breakpoints, 331–332
- *int 3* instruction, 331
- Interactive Disassembler (IDA), 121

- IsDebuggerPresent Windows API, 332
- kernel-mode debuggers, 117–118, 122–126
- NtQuerySystemInformation native API, 333–334
- OllyDbg, 118–120
- PEBrowse Professional Interactive, 122
- single-stepping, 16
- SoftICE, 124–126, 334
- tracing code, 15–16
- trap flag, 335
- user-mode debuggers, 117–122
- WinDbg
  - command-line interface, 119
  - disassembler, 119
  - extensions, 129
  - features, 119
  - improvements, 121
  - kernel-mode, 123–124
  - user-mode, 119–121
- debugging virtual machines, 127–128

decompilers

- antireversing, 348
- architecture, 459
- back end, 476–477
- code analysis, 466
- control flow analysis, 475
- control flow graphs (CFGs), 462
- data-flow analysis
  - data propagation, 468–470
  - data type propagation, 471–474
  - defined, 466–467
  - register variable identification, 470–471
  - single static assignment (SSA), 467–468
- defined, 16, 129
- expression trees, 461–462
- expressions, 461–462
- front end
  - basic block (BB), 464–466
  - function of, 463
  - semantic analysis, 463–464
  - IA-32 decompilers, 477
  - instruction sets, 460
  - intermediate representations, 459–460
  - library functions, 475–476
  - native code, 458–459
  - .NET, 424–425, 443
- Defender crackme program
  - brute-forcing, 409–415
- copy protection technologies, 415–416
- decrypted code analysis, 387–395
- decryption keys, 418–419
- disappearance of SoftICE, 396
- DUMPBIN, 372–376
- Executable Modules window, 371–372
- generic usage message, 370
- initialization routine reversal, 377–387
- inlining, 419
- KERNEL32.DLL, 400–404
- “killer” thread, 399–400
- obfuscated interface, 416–417
- parameter parsing, 404–406
- PEiD program, 376–377
Defender crackme program
(continued)
processor time-stamp verification
thread, 417–418
running, 370
secondary thread reversal, 396–399
16-digit hexadecimal serial num-
bers, 371
usernames, 371, 406–407
validating user information,
407–408
deleting malicious software, 277
Denial-of-Service (DoS) attacks, 280
deobfuscators, 345
DES (Data Encryption Standard)
algorithm, 200
detecting debuggers, 334–336
Devices directory, 83
“Differential Power Analysis”, Paul
Kocher, Joshua Jaffe, and Ben-
jamin Jun, 319
Digital Millennium Copyright Act
(DMCA), 20–22
digital rights management (DRM), 7,
319–321
Directive on the Legal Protection of
Computer Programs (European
Union), 23
directories (Windows operating sys-
tem), 83
disassemblers
antireversing, 336–343
decompilers, 463
defined, 15, 110–112
ILDasm, 115–116
Interactive Disassembler (IDA),
112–115
linear sweep, 111, 337–338
recursive traversal, 111, 338–343
Disassembly of Executable Code Revis-
ited, Benjamin Schwarz, Saumya
Debray, and Gregory Andrews,
111
dispatcher (Windows operating sys-
tem), 84
DIV instruction (IA-32), 49–50, 524
div instruction (MSIL), 432
DLLs (Dynamic Link Libraries), 28,
96–97
DMCA (Digital Millennium Copy-
right Act), 20–22
dongle, 316–317
DoS (Denial-of-Service) attacks, 280
DotFuscator obfuscator, 444, 448–451
doubly linked lists, 552–553
DRM (digital rights management), 7,
319–321
DUMPBIN executable-dumping tool,
133–136
Dynamic Link Libraries (DLLs), 28,
96–97

E
EAX register, 45–46
EBP register, 45–46
EBX register, 45–46
ECX register, 45–46
EDI register, 45–46
EDX register, 45–46
EFLAGS register, 46, 519–520
ElcomSoft software company, 22
capsulation, 27
encrypted assemblies (.NET), 453
encryption
antireversing, 330
Cipher Block Chaining (CBC), 415
copy protection technologies, 318
DES (Data Encryption Standard)
algorithm, 200
3DES encryption algorithm, 200
XOR algorithm, 416
Engineering a Compiler, Keith D. Cop-
pier and Linda Torczon, 54
ENTER instruction, 538–540
epilogues in functions, 486


Index 577

ESI register, 45–46
ESP register, 45–46
European Union’s Directive on the Legal Protection of Computer Programs, 23
evaluation stack (MSIL), 430
events, 86
exception handlers, 105–107
exceptions, 105–107
EXECryptor (StrongBit Technology), 345
executable data sections, 43
executable formats
directories, 99–102
exports, 99
file alignment, 95
headers, 97–98
image sections, 95
imports, 99
relative virtual address (RVA), 95
relocations, 93–95
section alignment, 95–96
executable-dumping tools, 133–138
execution environments
defined, 60
microprocessors, 63–68
virtual machines, 60–63
expression trees, 461–462
expressions, 461–462

F
fastcall calling convention, 541
faults (pages), 73–74
Felten vs. RIAA case, 22
file formats
.crx file format, 202–204
Microsoft Word file format, 200
reversing, 202–204
file-backed section object, 78
FileMon system-monitoring tool, 130
finding crackmes, 420
firmware malware, 279–280
flags
carry flag (CF), 520–521
defined, 519
EFLAGS register, 519–520
overflow flag (OF), 520–521
parity flag (PF), 521
sign flag (SF), 521
status flags, 46–47
zero flag (ZF), 521
flow analysis
data propagation, 468–470
data type propagation, 471–474
defined, 466–467
register variable identification,
470–471
single static assignment (SSA),
467–468
flow control
conditional blocks, 32
defined, 32
loops, 33
low-level implementation, 43–44
switch blocks, 33
front end of decompilers
basic block (BB), 464–466
function of, 463
semantic analysis, 463–464
function calls
assembly language instructions, 51
stack, 42
function-level working-set tuning, 515–517
functions
alldiv, 530–534
allmul, 530
calling, 487
functions (continued)
Cryptex command-line data encryption tool, 205–207
defined, 486
epilogues, 486
(-functions, 468
imported, 487–488
internal, 487
intrinsic string-manipulation functions, 249–250
library functions, 475–476
prologues, 486
RtlDeleteElementGeneric Table, 193–194
RtlGetElementGenericTable
disassembly, 153–155
initialization, 155–159
logic and structure, 159–161
search loop 1, 161–163
search loop 2, 163–164
search loop 3, 164–165
search loop 4, 165
setup, 155–159
source code, 165–168
RtlInitializeGenericTable, 146–151
RtlInsertElementGeneric Table, 168–170
RtlIsGenericTableEmpty, 152–153
RtlLocateNodeGenericTable, 170–178
RtlLookupElementGeneric Table, 188–193
RtlNumberGenericTable Elements, 151–152
RtlRealInsertElementWorker, 178–186
RtlSplay, 185–188
virtual functions, 557–560
(-functions, 468

G
GCC (GNU C Compiler) and G++ (GNU C++ Compiler) version 3.3.1 compiler, 59
General Method of Program Code Obfuscation, Gregory Wroblewski, 347
generic data structures, 547–548
generic data type arrays, 548
generic table API
callbacks prototypes, 195
definition, 145–146, 194–196
function prototypes, 196
internal data structures, 195
RtlDeleteElementGeneric Table function, 193–194
RtlGetElementGenericTable function
disassembly, 153–155
initialization, 155–159
logic and structure, 159–161
search loop 1, 161–163
search loop 2, 163–164
search loop 3, 164–165
search loop 4, 165
setup, 155–159
source code, 165–168
RtlInitializeGenericTable function, 146–151
RtlInsertElementGeneric Table function, 168–170
RtlIsGenericTableEmpty function, 152–153
RtlLocateNodeGenericTable function, 170–178
RtlLookupElementGeneric Table function, 188–193
RtlNumberGenericTable Elements function, 151–152
RtlRealInsertElementWorker function, 178–186
RtlSplay function, 185–188
Genesis gaming console (Sega Enterprises), 18
GLOBAL?? directory, 83
global variables, 542
GNU C Compiler (GCC) and GNU C++ Compiler (G++) compilers, 59
ground rules for reversing sessions, 142–143

H
Hacarmy.D, Trojan/Backdoor program, 285–305
Hack SDMI challenge, 22
handles, 81
hardware breakpoints, 331–332
hardware exceptions, 105
hardware-based copy protection technologies, 316–317
heap, 42
heap overflows, 255–256
Hex Workshop (BreakPoint Software, Inc.), 131–132
high-level data management, 38
high-level languages, 33–37

I
IA-32 decompilers, 477
IA-32 instructions
ADC, 529
ADD, 49–50, 522, 529
CALL, 51, 487, 540
CDQ, 535
CMP, 50, 480–483
Conditional Move (CMOVcc), 514–515
DIV, 49–50, 524
DIV/IDIV, 524
ENTER, 538–540
IDIV, 49–50, 524
IMUL, 49–50, 523
int 3, 331
Jcc, 51
LEA, 522
LEAVE, 538, 540
MOV, 49
MOVSX, 535
MOVZX, 534–535
MUL, 49–50, 523
opcode (operation code), 47
operands, 47–48
RET, 51, 540
SBB, 529
Set Byte on Condition (SETcc), 513–514
SUB, 49–50, 522, 529
SYSENTER, 394
IA-32 registers
defined, 39, 44–45
EAX, 45–46
EBP, 45–46
EBX, 45–46
ECX, 45–46
EDI, 45–46
EDX, 45–46
EFLAGS, 46, 519–520
ESI, 45–46
ESP, 45–46
IDA (Interactive Disassembler), 112–115, 121
IDC, BSA and IDC Global Software Piracy Study, 310
IDIV instruction, 49–50, 524
IIS Indexing Service Vulnerability, 262–271
IL (Intermediate Language)
activation records, 430
add instruction, 432
beq instruction, 432
bge instruction, 432
bgt instruction, 432
ble instruction, 432
blt instruction, 432
bne instruction, 432
box instruction, 432
br instruction, 432
C#, 36–37
call instruction, 431
code samples
  counting items, 433–435
  linked lists, 436–443
details, 424
div instruction, 432
evaluation stack, 430
ldarg instruction, 431
ldc instruction, 431
ldfld instruction, 431
ldloc instruction, 431
mul instruction, 432
.NET executables, 429
newarr instruction, 433
newobj instruction, 433
ret instruction, 431
starg instruction, 431
stfld instruction, 431
stloc instruction, 431
sub instruction, 432
switch instruction, 432
unbox instruction, 432
ILDasm, 115–116
imported functions, 487–488
imported variables, 544–546
IMUL instruction, 49–50, 523–524
information theft, 281
information-stealing worms, 278–279
inheritance, 29
inherited classes, 555–556
inlining, 353, 419
input/output system (Windows operating system), 103–104
instruction sets for decompilers, 460
instructions (IA-32)
  ADC, 529
  ADD, 49–50, 522, 529
  CALL, 51, 487, 540
  CDQ, 535
  CMP, 50, 480–483
  Conditional Move (CMOVcc), 514–515
  DIV, 49–50, 524
  DIV/IDIV, 524
  ENTER, 538–540
  IDIV, 49–50, 524
  IMUL, 49–50, 523
  int 3, 331
  Jcc, 51
  LEA, 522
  LEAVE, 538, 540
  MOV, 49
  MOVSX, 535
  MOVZ, 534–535
  MUL, 49–50, 523
  opcode (operation code), 47
  operands, 47–48
  RET, 51, 540
  SBB, 529
  Set Byte on Condition (SETcc), 513–514
  SUB, 49–50, 522, 529
  SYSENTER, 394
instructions (MSIL)
  add, 432
  beq, 432
  bge, 432
  bgt, 432
  ble, 432
  blt, 432
  bne, 432
Index  581

- box, 432
- br, 432
- call, 431
- div, 432
- ldarg, 431
- ldc, 431
- ldfld, 431
- ldloc, 431
- mul, 432
- newarr, 433
- newobj, 433
- ret, 431
- starg, 431
- stfld, 431
- stloc, 431
- sub, 432
- switch, 432
- unbox, 432
- int 3 instruction, 331
- integer overflows, 256–260

Intel
- assembly language notation, 49
- C++ Compiler version 8.0, 59–60
- LaGrande Technology Architectural Overview, 319
- NetBurst microarchitecture, 65–67
- intellectual property, 310
- Interactive Disassembler (IDA), 112–115, 121
- interleaving code, 354–355

Intermediate Language (IL)
- activation records, 430
- add instruction, 432
- beq instruction, 432
- bge instruction, 432
- bgt instruction, 432
- ble instruction, 432
- blt instruction, 432
- bne instruction, 432
- box instruction, 432
- br instruction, 432
- C#, 36–37
- call instruction, 431
- code samples
  - counting items, 433–435
  - linked lists, 436–443
  - details, 424
- div instruction, 432
- evaluation stack, 430
- ldarg instruction, 431
- ldc instruction, 431
- ldfld instruction, 431
- ldloc instruction, 431
- mul instruction, 432
- .NET executables, 429
- newarr instruction, 433
- newobj instruction, 433
- ret instruction, 431
- starg instruction, 431
- stfld instruction, 431
- stloc instruction, 431
- sub instruction, 432
- switch instruction, 432
- unbox instruction, 432
- intermediate representations, 55–56, 459–460
- internal functions, 487
- interoperability, 8, 17, 142
- interpreters, 61–62
- intrinsic string-manipulation functions, 249–250
- I/O system (Windows operating system), 103–104
- IsDebuggerPresent Windows API, 332–333

J

J#, 428
- Jaffe, Joshua, “Differential Power Analysis”, 319
- Java, 36, 423
- Java Virtual Machine (JVM), 60
- Jcc instructions, 51
- JiTs (just-in-time compilers), 62
Johnstone, Richard, *Computer Software Security System* patent, 311
Jun, Benjamin, “Differential Power Analysis”, 319
just-in-time compilers (JiTs), 62
JVM (Java Virtual Machine), 60

K
kernel memory, 74
kernel memory space, 75–77
kernel mode, 72–73
kernel-mode debuggers
applications, 122–123
defined, 117–118
limitations, 123
SoftICE, 124–126
virtual machines, 127
WinDbg, 123–124
Key ID (Windows Media Rights Manager), 321
KeygenMe-3 crackme program, 358–363
keygenning, 364–365
keywords
class, 547
register, 545
static, 543
struct, 547
volatile, 545
kleptographic worms, 278
Knuth, Donald E.
*The Art of Computer Programming — Volume 3: Sorting and Searching (Second Edition)*, 177, 187
Kocher, Paul, “Differential Power Analysis”, 319
Kruegel, Christopher, “Static Disassembly of Obfuscated Binaries”, 344

L
LaGrande Technology Architectural Overview, Intel, 319
last in, first out (LIFO), 40
layout
doubly linked lists, 553
singly linked lists, 551
stack, 539
stack frames, 539
trees, 554
ldarg instruction, 431
ldc instruction, 431
ldfld instruction, 431
ldloc instruction, 431
LEA instruction, 522
LEAVE instruction, 538, 540
legality of reverse engineering, 17–23
lexical analysis or scanning, 55
libraries, 28
library functions, 475–476
license agreements, 23
licenses for software, 311
Lie, David, “Architectural Support for Copy and Taper Resistant Software”, 319
LIFO (last in, first out), 40
linear sweep disassemblers, 337–338
line-level working-set tuning, 516, 518
linked lists, 32, 549–553
Linux, 423
Index

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
machine code, 11
Maier, David, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252
malicious software
  adware, 276–277
  backdoors, 276
  BIOS/firmware, 279–280
  defined, 5–6, 273
  deleting, 277
  information-stealing worms, 278–279
  metamorphism, 283–285
  mobile code, 276
  polymorphism, 282–283
  spyware, 276–277
  Trojan/Backdoor.Hacarmy.D program, 285–305
  Trojan horses, 275
  uses
  backdoor access, 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
MD5 cryptographic hashing algorithm, 213
media-based copy protection technologies, 314–316
Memon, Nasir, “Protecting Digital Media Content”, 322
memory management in Windows
  kernel memory, 74–75
  kernel memory space, 75–77
  page faults, 73–74
  paging, 73
  section objects, 77–78
  user memory, 74–75
memory management in Windows
(continued)
user-mode allocations, 78–79
VAD (Virtual Address Descriptor)
tree, 78
virtual memory, 72–73
Virtual Memory Manager, 79–80
working sets, 74
memory mapped files, 78
metadata (.NET), 426
metamorphism, 283–285
methodologies of reversing, 110
methods, 556–557
microcode, 65
Microprocessor for Executing Enciphered
Programs patent, Robert M. Best, 311, 318
microprocessors, 63–68
Microsoft Intermediate Language
(MSIL)
activation records, 430
add instruction, 432
beq instruction, 432
bge instruction, 432
bgt instruction, 432
ble instruction, 432
blt instruction, 432
bne instruction, 432
box instruction, 432
br instruction, 432
C#, 36–37
call instruction, 431
code samples
  counting items, 433–435
  linked lists, 436–443
details, 424
div instruction, 432
evaluation stack, 430
ldarg instruction, 431
ldc instruction, 431
ldfld instruction, 431
ldloc instruction, 431
mul instruction, 432
.NET executables, 429
newarr instruction, 433
newobj instruction, 433
ret instruction, 431
starg instruction, 431
stfld instruction, 431
stloc instruction, 431
sub instruction, 432
switch instruction, 432
unbox instruction, 432
Microsoft (MS)
C/C++ Optimizing Compiler version 13.10.3077, 59
cryptographic service providers (CSPs), 207
DUMPBIN executable-dumping tool, 133–136
IIS Indexing Service Vulnerability, 262–271
ILDasm, 115–116
Next-Generation Secure Computing Base (NGSCB), 323–324
Virtual PC, 128
WinDbg debugger, 119–121, 123–124
Microsoft .NET platform
assemblies, 426, 453
C# programming language, 428
class library, 426
Common Intermediate Language (CIL), 429
Common Language Runtime (CLR), 426–427
Common Type System (CTS), 428–429
comparison with Java, 423
compilation stages, 429
decomilers, 424–425, 443
IL (Intermediate Language), 424, 429–430
J# programming language, 428
Managed C++ programming language, 428
managed code, 426
metadata, 426
.NET Framework environment, 426
obfuscators, 424, 444–455
Visual Basic .NET programming language, 428
Microsoft Word file format, 200
Misra, Jayadeve, *Strategies to Combat Software Piracy*, 312
mobile code, 276
modules, 28
modulo, 527–528
monitoring tools
  defined, 15, 129–130
  FileMon, 130
  PortMon, 130
  Process Explorer, 130–131
  RegMon, 130
  TCPView, 130
  TDIMon, 130
  WinObj, 130
MOV instruction, 49
MOVSX instruction, 535
MOVZX instruction, 534–535
MS (Microsoft)
  C/C++ Optimizing Compiler version 13.10.3077, 59
cryptographic service providers (CSPs), 207
DUMPBIN executable-dumping tool, 133–136
IIS Indexing Service Vulnerability, 262–271
ILDasm, 115–116
Next-Generation Secure Computing Base (NGSCB), 323–324
Virtual PC, 128
WinDbg debugger, 119–121, 123–124
MSIL (Microsoft Intermediate Language)
  activation records, 430
  add instruction, 432
  beq instruction, 432
  bge instruction, 432
  ble instruction, 432
  blt instruction, 432
  bne instruction, 432
  box instruction, 432
  br instruction, 432
  C#, 36–37
  call instruction, 431
code samples
  counting items, 433–435
  linked lists, 436–443
details, 424
div instruction, 432
evaluation stack, 430
ldarg instruction, 431
ldc instruction, 431
ldfld instruction, 431
ldloc instruction, 431
mul instruction, 432
.NET executables, 429
newarr instruction, 433
newobj instruction, 433
ret instruction, 431
starg instruction, 431
stfld instruction, 431
stloc instruction, 431
sub instruction, 432
switch instruction, 432
unbox instruction, 432
Muchnick, Steven S., *Advanced Compiler Design and Implementation*, 54
MUL instruction, 49–50, 523–524
mul instruction, 432
multidimensional arrays, 31
multiple-alternative conditional, 490–491
mutexes, 87
Nagra, J., “A Functional Taxonomy for Software Watermarking”, 322
named objects, 81–83
native API, 90–91
native code decompilers, 457–459
Nebbett, Gary, Windows NT/2000 Native API Reference, 91, 389
.NET assemblies, 426, 453
C# programming language, 428
class library, 426
Common Intermediate Language (CIL), 429
Common Language Runtime (CLR), 426–427
Common Type System (CTS), 428–429
comparison with Java, 423
compilation stages, 429
decompilers, 424–425, 443
IL (Intermediate Language), 424, 429–430
J# programming language, 428
Managed C++ programming language, 428
managed code, 426
metadata, 426
.NET Framework environment, 426
obfuscators, 424, 444–455
Visual Basic .NET programming language, 428
NetBurst microarchitecture, 65–67
networking protocols, 202
newarr instruction, 433
newobj instruction, 433
Next-Generation Secure Computing Base (NGSCB), 323–324
nonexecutable memory, 254–255
NtQuerySystemInformation native API, 333–334
NuMega SoftICE debugger, 124–126, 334
n-way conditionals, 33, 499–500, 502–504

O
OBfuscATE macro, 343–344
obfuscation, 328–329, 344–345
obfuscators
defined, 63
DotFuscator, 444, 448–451
.NET, 424, 444–455
Remotesoft Obfuscator, 451–452
Remotesoft Protector, 452–455
Spices.Net, 444
XenoCode, 444, 446–447
object code, 11
object-oriented design (OOD), 29
objects
base object, 29
clients, 29
defined, 29
inheritance, 29
named objects, 81–83
object-oriented design (OOD), 29
polymorphism, 29, 35
Windows operating system, 80–83
OF (overflow flag), 520–521
offline code analysis, 110
OllyDbg debugger, 118–120
OOD (object-oriented design), 29
opaque predicates, 338–340, 346–347
opcode (operation code), 11, 47
operand comparison, 50
operands
comparing, 480–483
instructions, 47–48
signed, 480–481
unsigned, 482–483
operating systems
defined, 13
Windows application programming interfaces (APIs), 88–91
architecture, 70–71
compatibility, 71
context switching, 85–86
critical sections, 87
directories, 83
dispatcher, 84
dynamically linked libraries (DLLs), 96–97
events, 86
exception handlers, 105–107
exceptions, 105–107
executable formats, 93–102
features, 70–71
handles, 81
history, 70
I/O system, 103–104
kernel memory, 74
kernel memory space, 75–77
kernel mode, 72–73
multiprocessor capability, 71
multithreaded, 71
mutexes, 87
object manager, 80–81
objects, 80–83
page faults, 73–74
paging, 73
portability, 71
process initialization sequence, 87–88
processes, 84
scheduler, 84
section objects, 77–78
security, 71
semaphores, 87
64-bit versions, 71–72
supported hardware, 71
synchronization objects, 86–87
system calling mechanism, 91–93
32-bit versions, 71–72
threads, 84–85
user memory, 74
user mode, 72–73
user-mode allocations, 78–79
VAD (Virtual Address Descriptor) tree, 78
virtual memory, 70, 72
Virtual Memory Manager, 79–80
Win32 subsystem, 104–105
working sets, 74
operation code (opcode), 11, 47
operators, 492–499
optimizers (compilers), 56–57
OR logical operator, 492, 494–498
ordering transformations, 346, 355
outlining, 353
overflow bugs
heap overflows, 255–256
integer overflows, 256–260
stack overflows, 245–255
string filters, 256
overflow flag (OF), 520–521
P
page faults, 73–74
page tables (virtual memory), 72
pagefile-backed section object, 78
pages (virtual memory), 72
paging, 73
parity flag (PF), 521
password verification process
“Bad Password” message, 207–210
hashing the password, 213–218
password transformation algorithm, 210–213
patching
Hex Workshop, 131–132
KeygenMe-3 crackme program, 358–363
patents, 20, 311, 318
PE (Portable Executable) directories, 99–102
exports, 99
file alignment, 95
headers, 97–98
image sections, 95
PE (Portable Executable) (continued)
imports, 99
relative virtual address (RVA), 95
relocations, 93–95
section alignment, 95–96
PEBrowse Professional Interactive debugging, 122
executable dumping, 137–138
PEiD program, 376–377
PEView executable-dumping tool, 137
PF (parity flag), 521
Phrack paper, Aleph1, 245
pipelines, 65–67
piracy
class breaks, 312–313
copy protection schemes, 313
copy protection technologies, 311–313
copyrights, 309–310
digital rights management (DRM), 319–321
intellectual property, 310
magnitude of, 309
software, 310–311
software piracy, 312
trusted computing, 322–324
watermarking, 321–322
polymorphism, 29, 35, 282–283
portability of Windows operating system, 71
Portable Executable (PE)
directories, 99–102
exports, 99
file alignment, 95
headers, 97–98
image sections, 95
imports, 99
relative virtual address (RVA), 95
relocations, 93–95
section alignment, 95–96
PortMon system-monitoring tool, 130
posttested loops, 506
power usage analysis attacks, 319
precompiled assemblies (.NET), 453
PreEmptive Solutions DotFuscator obfuscator, 444, 448–451
pretested loops, 504–506
primitive data types, 472–473
procedures
alldiv, 530–534
allmul, 530
calling, 487
Cryptex command-line data encryption tool, 205–207
defined, 486
epilogues, 486
(, 468
imported, 487–488
internal, 487
intrinsic string-manipulation, 249–250
library, 475–476
prologues, 486
RtlDeleteElementGenericTable, 193–194
RtlGetElementGenericTable
disassembly, 153–155
initialization, 155–159
logic and structure, 159–161
search loop 1, 161–163
search loop 2, 163–164
search loop 3, 164–165
search loop 4, 165
setup, 155–159
source code, 165–168
RtlInitializeGenericTable, 146–151
RtlInsertElementGenericTable, 168–170
RtlIsGenericTableEmpty, 152–153
RtlLocateNodeGenericTable, 170–178
RtlLookupElementGenericTable, 188–193
RtlNumberGenericTableElements, 151–152
RtlRealInsertElementWorker, 178–186
RtlSplay, 185–188
Process Explorer system-monitoring tool, 130–131
process initialization sequence, 87–88
processes, 84
program comprehension, 443
program data
defined, 537
stack
defined, 538
layout, 539
stack frames
defined, 538
ENTER instruction, 538–540
layout, 539
LEAVE instruction, 538, 540
program structure
task frame
conditional blocks, 32
defined, 32
loops, 33
switch blocks, 33
data management, 29–32
defined, 26–27
encapsulation, 27
modules, 28
objects, 29
procedures, 28
programming languages
C, 34–35
C#, 36–37, 428
C++, 35
Java, 36, 423
.NET, 428
prologues in functions, 486
proprietary software, 7–8
“Protecting Digital Media Content”, Nasir Memon and Ping Wah Wong, 322
protection technologies
attacks, 324
challenge response, 315–316
class breaks, 312–313
 cracking, 357–358
crypto-processors, 318–319
Defender crackme program, 415–416
donngle, 316–317
encryption, 318
hardware-based, 316–317
media-based, 314–316
objectives, 312
online activation, 315–316
requirements, 313
ripping algorithms, 365–370
serial numbers, 315
server-based software, 317
StarForce suite (StarForce Technologies), 345
trusted components, 312
Uncrackable Model, 314
Protector (Remotesoft), 452–455
Pu, Calton, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252
pure arithmetic, 510–512
R
reciprocal multiplication, 524–527
recursive traversal disassemblers, 338–343
redundancy elimination, 57
register keyword, 545
register transfer languages (RTL), 468
register values, 42
registers
  defined, 39, 44–45
  EAX, 45–46
  EBP, 45–46
  EBX, 45–46
  ECX, 45–46
  EDI, 45–46
  EDX, 45–46
  EFLAGS, 46, 519–520
  ESI, 45–46
  ESP, 45–46
RegMon system-monitoring tool, 130
relative virtual address (RVA), 95
Remotesoft
  Obfuscator, 451–452
  Protector, 452–455
resource theft, 280–281
restructuring arrays, 356
RET instruction, 51, 540
ret instruction, 431
Reverse Compilation Techniques,
  Christina Cifuentes, 477
reverse engineering
  applications, 4–5
  code-level reversing, 13–14
  competing software, 8–9, 18–19
data reverse engineering
  Cryptex command-line data encryption tool, 200–202
  defined, 199
  file formats, 202–204
  Microsoft Word file format, 200
  networking protocols, 202
  uses, 199–200
defined, 3–4
ground rules, 142–143
legality, 17–23
live code analysis, 110
offline code analysis, 110
security-related
  cryptographic algorithms, 6
digital rights management (DRM), 7
malicious software, 5–6
proprietary software, 7–8
software development, 8–9
system-level reversing, 13–14
reversing tools
  Cryptex command-line data encryption tool, 200, 202
d debuggers, 15–16, 116–126
decompilers, 16, 129
disassemblers, 15, 110–116
executable dumping, 133–138
patching, 131–132
system monitoring, 15, 129–130
ripping algorithms, 365–370
RTL (register transfer languages),
  468
RtlDeleteElementGenericTable function, 193–194
RtlGetElementGenericTable function
disassembly, 153–155
initialization, 155–159
logic and structure, 159–161
search loop 1, 161–163
search loop 2, 163–164
search loop 3, 164–165
search loop 4, 165
setup, 155–159
source code, 165–168
RtlInitializeGenericTable function, 146–151
RtlInsertElementGenericTable function, 168–170
RtlIsGenericTableEmpty function, 152–153
RtlLocateNodeGenericTable function, 170–178
RtlLookupElementGenericTable function, 188–193
RtlNumberGenericTableElements function, 151–152
RtlRealInsertElementWorker function, 178–186
RtlSplay function, 185–188
RVA (relative virtual address), 95

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

stack
- defined, 40, 538
- function calls, 42
- layout, 539
- LIFO (last in, first out), 40
- local variables, 42
- pop operations, 41
- push operations, 41
- register values, 42
- stack checking, 250–254
- stack frames
  - defined, 538
  - ENTER instruction, 538–540
  - layout, 539
  - LEAVE instruction, 538, 540
  - stack overflows, 245–255
- StarForce suite (StarForce Technologies), 345
- starg instruction, 431
- “Static Disassembly of Obfuscated Binaries”, Christopher Kruegel, et al., 344
- static keyword, 543
- static libraries, 28
- status flags, 46–47
- stdcall calling convention, 541
- stfld instruction, 431
  - stloc instruction, 431
- Strategies to Combat Software Piracy, Jayadeve Misra, 312
- string filters, 256
- StrongBit Technology EXECryptor, 345
- struct keyword, 547
- structured exception handling, 105–106
- structures for data alignment, 547–548
- arrays, 31, 548–549
- classes
  - constructors, 559–560
  - data members, 555–556
  - defined, 555
  - inherited classes, 555–556
  - methods, 556–557
  - virtual functions, 557–560
  - defined, 547
  - generic data structures, 547–548
  - linked lists, 32, 549–553
  - lists, 31
  - trees, 32, 552, 554
  - user-defined data structures, 30–31
  - variables, 30
- SUB instruction, 49–50, 522, 529
- sub instruction, 432
- switch blocks, 33, 499–504
- switch instruction, 432
- symbolic information, 328–330
- symbolic link directory, 83
- synchronization objects, 86–87
- SYSENTER instruction, 394
- system calling mechanism (Windows operating system), 91–93
- system flags, 46–47
- system software, 9–10
- system-level reversing, 13–14
- system-monitoring tools
  - defined, 15, 129–130
  - FileMon, 130
  - PortMon, 130
  - Process Explorer, 130–131
  - RegMon, 130
  - TCPView, 130
  - TDIMon, 130
  - WinObj, 130
- T
- table API
  - callbacks prototypes, 195
  - definition, 145–146, 194–196
  - function prototypes, 196
internal data structures, 195
RtlDeleteElementGenericTable function, 193–194
RtlGetElementGenericTable function, 153–168
RtlInitializeGenericTable function, 146–151
RtlInsertElementGenericTable function, 168–170
RtlIsGenericTableEmpty function, 152–153
RtlLocateNodeGenericTable function, 170–178
RtlLookupElementGenericTable function, 188–193
RtlNumberGenericTableElements function, 151–152
RtlRealInsertElementWorker function, 178–186
RtlSplay function, 185–188
table interpretation, 348–353
Tarjan, Robert Endre, Self-adjusting binary search trees, Journal of the ACM (JACM), 187
A Taxonomy of Obfuscating Transformations, Christian Collberg, Clark Thomborson, and Douglas Low, 348
TCPView system-monitoring tool, 130
TDIMon system-monitoring tool, 130
technologies for copy protection attacks, 324
challenge response, 315–316
class breaks, 312–313
cracking, 357–358
crypto-processors, 318–319
Defender crackme program, 415–416
dongle, 316–317
encryption, 318
hardware-based, 316–317
media-based, 314–316
objectives, 312
online activation, 315–316
requirements, 313
ripping algorithms, 365–370
serial numbers, 315
server-based software, 317
StarForce suite (StarForce Technologies), 345
trusted components, 312
Uncrackable Model, 314
32-bit versions of Windows, 71–72
thistcall calling convention, 541
Thomborson, Clark “A Functional Taxonomy for Software Watermarking”, 322
“Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs”, 346
A Taxonomy of Obfuscating Transformations, 348
thread information block (TIB), 106
thread-local storage (TLS), 546–547
threads, 84–85
3DES encryption algorithm, 200
tools
Cryptex command-line data encryption tool, 200, 202
debuggers, 15–16, 116–126
decomilers, 16, 129
disassemblers, 15, 110–116
executable dumping, 133–138
patching, 131–132
system monitoring, 15, 129–130
Torczon, Linda, Engineering a Compiler, 54
trade secrets, 20
Transscopy copy protection technology, 314
trap flag, 335
trees, 32, 552, 554
Trojan horses, 275
trusted computing, 322–324
tuning working sets
  function-level, 515–517
  line-level, 516, 518
two-way conditionals, 489–490
type conversion errors, 260–262
type conversions
  defined, 534
  sign extending, 535
  zero extending, 534–535

U
unbox instruction, 432
Uncrackable Model, 314
undocumented APIs, 142–144
unrolling loops, 508–509
unsigned conditional codes, 485–486
unsigned operands, 482–483
US vs. Sklyarov case, 22
user memory, 74
user mode, 72–73
user-defined data structures, 30–31
user-mode debuggers, 117–122

V
VAD (Virtual Address Descriptor)
  tree, 78
vandalism, 280
variables
  defined, 30
  global variables, 542
  imported variables, 544–546
  local variables, 542–544
verification process for passwords
  “Bad Password” message, 207–210
  hashing the password, 213–218
  password transformation algorithm, 210–213
Virtual Address Descriptor (VAD)
  tree, 78
virtual functions, 557–560
virtual machines
  bytecodes, 12–13, 60–63
  debugging, 127–128
Virtual Memory Manager, 79–80
virtual memory (Windows operating system), 70, 72
Virtual PC (Microsoft), 128
viruses, 274
Visual Basic .NET, 428
VMWare Workstation, 128
volatile keyword, 545
vulnerabilities
  defined, 245
  heap overflows, 255–256
  IIS Indexing Service Vulnerability, 262–271
  integer overflows, 256–260
  intrinsic string-manipulation functions, 249–250
  malicious software, 281
  stack overflows, 245–255
  string filters, 256
  type conversion errors, 260–262

W
watermarking, 321–322
Win32 API, 88–90
Win32 subsystem, 104–105
WinDbg debugger
  command-line interface, 119
  disassembler, 119
  extensions, 129
  features, 119
  improvements, 121
  kernel-mode, 123–124
  user-mode, 119–121
Windows APIs
  generic table API, 145–146
  IsDebuggerPresent, 332–333
  undocumented APIs, 142–144
Windows Media Rights Manager, 321
Windows NT/2000 Native API Reference, Gary Nebbett, 91, 389
Windows operating system
application programming interfaces (APIs), 88–91
architecture, 70–71
compatibility, 71
count context switching, 85–86
critical sections, 87
directories, 83
dispatcher, 84
dynamically linked libraries (DLLs), 96–97
events, 86
exception handlers, 105–107
exceptions, 105–107
executable formats, 93–102
features, 70–71
handles, 81
history, 70
I/O system, 103–104
kernel memory, 74
kernel memory space, 75–77
kernel mode, 72–73
multiprocessor capability, 71
multithreaded, 71
mutexes, 87
object manager, 80–81
objects, 80–83
page faults, 73–74
paging, 73
portability, 71
process initialization sequence, 87–88
processes, 84
scheduler, 84
section objects, 77–78
security, 71
semaphores, 87
64-bit versions, 71–72
supported hardware, 71
synchronization objects, 86–87
system calling mechanism, 91–93
32-bit versions, 71–72
threads, 84–85
user memory, 74
user mode, 72–73
user-mode allocations, 78–79
VAD (Virtual Address Descriptor) tree, 78
virtual memory, 70, 72
Virtual Memory Manager, 79–80
Win32 subsystem, 104–105
working sets, 74
WinObj system-monitoring tool, 130
Wong, Ping Wah, “Protecting Digital Media Content”, 322
working sets, 74
working-set tuning
function-level, 515–517
line-level, 516, 518
worms
Code Red Worm, 262
defined, 274–275
information-stealing worms, 278–279
Wroblewski, Gregory, General Method of Program Code Obfuscation, 347

X
XenoCode obfuscator, 444, 446–447
XOR algorithm, 416

Z
Zeltser, Lenny, Malware: Fighting Malicious Code, 280
zero extending, 534–535
zero flag (ZF), 521
Zhang, Qian, Automatic Detection and Prevention of Buffer-Overflow Attacks, 252