INDEX

Numerics
32-bit applications, WOW64, running in, 598-602
64-bit debugging, 603-624
  Application Verifier, 604
  current call stacks, discovering, 609-613
  current register values, finding, 605-607
  custom debugger extensions, writing, 629
  Debugging Tools for Windows, 603-604
  Ethereal, 605
  interprocess communication, 628-629
  Leak Diagnostic tool, 603
  local variable values, discovering, 613-614
  memory corruption, 625-626
  process memory, inspecting, 615
  processors, code processing, 609
  security, 626-628
  WOW64 commands, 607-609
64-bit operating systems, 595-598
  32-bit applications, running in, 598-602
  64-bit processors, 595-596

A
access, breakpoints, 170-171
access control, Windows Vista, side effects, 712-714
access tokens, 325-330
  client server applications
    impersonation levels, 337-338
    remote authentication, 335-337
    security support provider interface support, 335-337
    token propagation, 334-335
  impersonation tokens, dumping, 351
  process access tokens, obtaining, 329
  thread impersonation tokens, displaying, 329

access violation, dump files, analysis, 646-647
Access violation enumerating registry values
  listing (5-12), 232-234
AccessCheck function, 323-324
accessing cell debugging information, 399-408
ACE (access control entry), 320
ACLs (Access Control Lists), 320-322
activation checks, DCOM, 355-363
activation failures, DCOM, Windows XP2, 371-376
AddEnd option (Heaps test setting), 752
Addr option (Heaps test setting), 752
!address command, 90-91
  listing (2-39), 91
!address debugger command example
  listing (2-38), 90
!address extension command, 473
ADDRESS objects, naming conventions, 390
address space layout, Windows Vista, 716-717
AddrStart option (Heaps test setting), 752
ADPlus, 632
  dump files, generating, 639-641
AeDebug key, 638
  auto registry value, 638
  debugger registry value, 638-639
agestore.exe, 8
algorithms, LIFO (last in first out)
  semantics, 209
aliases, events, 134
All thread stacks currently running in the
  process listing (10-7), 512
allocations
- heaps
  - back end allocators, 263-270, 272-281
  - front end allocators, 261-263
- memory blocks, 269
allocators, LeakDiag tool, 4-5
analysis
- custom analysis scripts, authoring, 697-699
- dump files
  - access violation, 646-647
  - handle leaks, 647-652
- handle leaks, Debug Diagnostic tool, 693-697
- memory leaks, Debug Diagnostic tool, 693-697
- network traffic, 413-421
analysis process
- memory corruption, 201
  - avoidance strategies, 209
  - detection tools, 208
  - instrument source code, 208
  - source code analysis, 202-208
  - state analysis, 201-202
- problem synchronization, 505-509
- resource leaks, 428-433
analysis scripts, authoring, Debug Diagnostic Tool, 697-699
analyze extension command, 691, 699
analyse extension command, 691, 699
fault follow-up, 706-708
faulty applications, analyzing, 700
output, 701, 703
results, analyzing, 701-706
anti-debugging techniques, 159
Application crash seen under the debugger listing (6-5), 283-286
Application Errors section (Dr. Watson dialog box), 656-657
Application manifest requesting a high integrity level listing (15-9), 724
Application option (Dr. Watson dialog box), 656-657
Application that calls a function in a DLL listing (5-17), 245-247
Application that establishes a connection to a data source listing (5-9), 223
Application Verifier, 10-16, 604
test settings, 747
  - DangerousAPIs, 774-775
  - DirtyStacks, 775
  - Exceptions, 747
  - FilePaths, 764-765
  - Handles, 747-748
  - Heaps, 749-757
  - HighVersionLie, 765-767
  - InteractiveServices, 767-768
  - KernelModeDriverInstall, 768-769
  - Locks, 757-759
  - Low Resource Simulation, 769-770
  - LuaPrior, 771-774
  - Memory, 760-762
  - PrintAPI, 776
  - PrintDriver, 776
  - ThreadPool, 762-763
  - TimeRollOver, 775
  - TSL, 764
Application verifier reported heap block corruption listing (6-8), 295-300
Application Verifier, heap block corruption reports, 295-299
applications
- crashes, 200
deadlocks, 510-516, 519
exceptions, 516-522
handle leaks, 436-460
multithreading, 550
resource leaks, memory leaks, 460-492
unpredictable behavior, 200
architecture, Windows Error Reporting, 662-682
assembly code
  - ProcA procedure, 214-216
  - Sum function, 217-219
Assembly code for ProcA listing (5-5), 215-216
Assembly code of the ThreadProcedure function listing (5-4), 213
Assembly code representing the Function5 function listing (12-7), 610
Assembly instructions right after our call to Sum listing (5-7), 219
Assembly listing generated for the function from Listing 324 listing (3-26), 163 asynchronous operations, stack corruptions, 231-240

attaching debuggers
to running processes, 35
to running processes nonintrusively, 36 processes to debuggers, 281

Attempt at manually constructing the stack listing (5-18), 248-255

authentication, remote authentication, 423 client server applications, 335-337

authoring custom analysis scripts, Debug Diagnostic Tool, 697-699

auto registry value, AeDebug key, 638 avoidance strategies memory corruption, 209 memory leaks, 491-492 resource leaks, 433 stack corruptions, 255-258

B ba command, 89
back end allocators, heaps, 263-281
Backward option (Heaps test setting), 751
BCD (Boot Configuration Data) objects, 715-716

binary files, debug information, finding in, 51
Binary folder tree listing (4-3), 182
Binary representation of a security descriptor listing (7-3), 324
binary trees, C++ binary trees, implementing, 556, 559-562
Binplace file content listing (4-1), 181
Binplace.exe file, 181
Binplacing the symbol files listing (4-2), 182 blocks (memory) allocating, 209 freeing, 209
PEB (process environment block), locating, 270-272
boundaries (system), security checks, 338-340

breaking cell paths, 421-422 breakpoints (code), 168-170 access, 170-171
conditional breakpoints, 107 reference releases, detecting, 107 setting, 79-81 on access, 88-89

breaks, events adjusting, 136-140 inspecting, 135

BSTREE.EXE Using the Binary Tree Implementation listing (11-2), 559-562

BUFFER_OVERFLOW_CHECKS environment variable, 210
building debugger extensions, 593

C
-c parameter, 137
-c2 parameter, 137

C++ binary trees, implementing, 556-562

Caches, symbol caches, 54-55
call access checks, DCOM, 364-367
call stacks

current call stacks
discovering, 609-613
displaying, 73-79
function calls, 210
threads, displaying, 212

Call-stack function parameter listing (12-8), 612
calling conventions, 243
cdecl, 243
fastcall, 243
functions

cdecl, 240-243
stdcall, 240-243

stack corruptions, 240-255
stdcall, 243
thiscall, 243

CallProc function, 250-255
calls

client call information, attaining, 407-408
DCOM calls, impersonating, 395-396
information, attaining, 404-407
LRPC calls, impersonating, 395-396
cancellations, commands, implementing, 589-590
CCALL objects, naming conventions, 389
CDB (cdb.exe) tool (Debugging Tools for Windows), 31
cdb.exe, 8
cdecl calling convention, 240-243
cell debugging
configuring, 396-398
information, 398-399, 408-412
accessing, 399-408
cell paths, breaking, 421-422
Changing file integrity level using built in icacls.exe tool listing (15-12), 731
Changing kernel flags using command line gflags.exe listing (3-21), 150
Changing the current thread listing (3-36), 176
Changing the default register mask listing (2-20), 69
Client stack containing the activation call listing (7-33), 372
Client's thread waiting on LPC request to complete listing (8-1), 384
code
execution, 96-97
Microsoft Detours library, 7
processors, execution, 72
source code, Fibonacci, 75-76
code breakpoints
conditional breakpoints, 107
reference releases, detecting, 107
setting, 79-81
code execution, tracing, 94-95
Code exercising the exception dispatching logic listing (3-22), 154
code organization, debugger extensions, 567-570
COM interfaces, initializing, 572-573
Command Line mode (gflags), 19-21
Command to delete the symbol files from the symbol servers listing (4-8), 187
Command to store the symbols on the symbol servers listing (4-6), 184
commands
!address, 90
!address extension command, 473
!analyze extension, 691, 699-701, 703-705
ba, 89
cancellations, implementing, 589-590
!cs, 500
d, 85
d**, 87
dump, 635-636
dumptree command, implementing, 580-585
dv, 82
dumps, 553-554
building, 593
C++ binary tree implementation, 556-562
code organization, 567-570
command cancellations, 589-590
DebugExtensionNotify function, 577
dumptree command, 580-585
evaluating, 554
general extensions, 554
header files, 567-570
Help command, 579-580
initialization, 570-576
KnownStructOutput function, 578, 585-589
models, 563, 565
requirements, 565-570
session state changes, 577
uninitializing, 578
user mode extensions, 554
versioning, 592
!getcallinfo extension, 404-406
!getdbgcell extension, 406-407
!getendpointinfo extension, 400-403
!handle, 496-497, 502-503
!heap extension commands, 474-491
Help command, implementing, 579-580
!htrace extension command, 455-459
Contents of the stack at the point of crash listing (5-13), 235
Context, changing, 98-103
context changing commands, debuggers, 94-104
CONTEXT structure (exceptions), 146
CONTEXT structure, as defined in MSDN listing (3-17), 146
controlling
  events, user mode debuggers, 133-144
  exceptions, user mode debuggers, 144-166
  targets, debuggers, 168-177
Corporate Error Reporting, 633, 682-683
configuring, 683-687
errors, reporting, 687-690
GP (Group Policy) settings, 683-687
corruption (memory), 199-201, 259
application crashes, 200
detection process, 201
  avoidance strategies, 209
detection tools, 208
  instrument source code, 208
  source code analysis, 202-208
  state analysis, 201-202
heap corruptions, 251
  handle mismatches, 300-305
  heap overruns, 286-300
  heap reuse after deletion, 306-314
  heap underruns, 286-300
  uninitiated state, 282-286
stack corruptions, 209-222
  asynchronous operations, 231-240
  avoidance strategies, 255-255
calling conventions mismatch, 240-255
  stack overruns, 223-231
  stack pointers, 231-240
  unpredictable behavior, 200
cpr (create a process) event, creating, 141
Crash Dump section (Dr. Watson dialog box), 655
Crash Dump Type section (Dr. Watson dialog box), 655
Crash reproduced in the debugger listing (5-10), 225
crash rule (Debug Diagnostic Tool), 692
Creating a kernel mode dump file
listing (13-2), 644
Critical section class that handles the
lifetime of a critical section
listing (10-10), 521
critical sections
fields, direct usage, 545
orphaned critical sections, 516-529
synchronization, 497-502
managing, 545-550
unlocked critical sections, 543-545
CritSecInfo object, 697-698
!cs extension command, 500
c (create a thread) event, creating, 143
current call stacks
  discovering, 609-613
  displaying, 73-79
current register values
  debuggers, displaying, 68-71
  finding, 64-bit debuggers, 605-607
current threads, changing, 175
current time stamps, attaining, 400
custom analysis scripts, authoring with
  Debug Diagnostic tool, 697-699
custom debugger extensions, writing
  64-bit debuggers, 629
  Windows Vista, 741

D
d command, 85
d*s command, 87
DACLs (discretionary access control
  lists), 318
DangerousAPIs test setting (Application
  Verifier), 774-775
dbengprx.exe, 8
dbgpc.exe, 8
DCE/RPC (DCE Remote Procedure
  Call), 380
DCOM (Distributed Common Object
  Model), 381
  activation checks, 355-363
  activation failures, Windows XP SP2,
    371-376
call access checks, 364-367
calls, impersonating, 395-396
events, 354-367
local communication, debugging, 388-396
deadlocks, 510-516, 519
Debug Diagnostic Tool, 691, 693
  crash rule, 692
custom analysis scripts, authoring, 697-699
handle leaks, analyzing, 693-697
hang rule, 692
Host component, 692
leak tracker component, 692
leaks rule, 692
memory leaks, analyzing, 693-697
Service component, 692
starting, 692
User Interface component, 692
Debug Diagnostics Tool, 691
Debug directories after bin place operation
listing (4-5), 183
Debug directories immediately after
  building the binaries listing (4-4), 183
Debug Flags field (!heap extension
  command), 478
deg debug information, binary files,
  finding in, 51
Debug registers on a normal processor
listing (3-31), 171
deg servers, 110-113
DebugDiag, 27, 697
DebugDiag custom analysis script metadata
listing (14-1), 697
DebugExtensionNotify function, 577
Debugger COM interfaces initialization
listing (11-3), 573
Debugger events generated a WOW64
  process execution (xcopy.exe)
listing (12-14), 620-622
Debugger events generated by a simple
  process execution (xcopy.exe)
listing (3-7), 131
degger extensions, 33
custom 64-bit debugger extensions,
  writing, 629
writing, Windows Vista, 741
degen registry value, AeDebug key,
  638-639
Debugger Tools for Windows
   KD (kd.exe) tool, 32
   WinDbg (windbg.exe) tool, 32
deguggers, 30, 45, 123-124
   64-bit debuggers, 605-624
code processing, 609
current call stack, 609-613
current register values, 605-607
custom debugger extensions, 629
   interprocess communication, 628-629
   local variable values, 613-614
   memory corruption, 625-626
   process memory, 615
   security, 626-628
   WOW64 commands, 607-609
address space layout, Windows Vista, 716-717
code breakpoints
   conditional breakpoints, 107
   setting, 79-81, 88-89
code execution, 96-97
   tracing, 94-95
commands, 32-33
   context changing commands, 94-104
   entering, 46
   exploratory commands, 66-94
   helper commands, 104-107
context, changing, 98-103
current call stacks, displaying, 73-79
current register values, displaying, 68-71
dump files, generating, 634-639
events, 140-144
   cpr (create a process) event, 141
   ct (create a thread) event, 143
   epr (exit a process) event, 142
   et (exit a thread) event, 144
   ibp (initial breakpoint) event, 141
   ld (load a module) event, 142
   ud (unload a module) event, 143
exceptions, structured exception dispatching
   mechanism, 144-153
extensions, 553-554
   building, 593
   C++ binary tree implementation, 556, 559-562
   code organization, 567-570
   command cancellations, 589-590
   DebugExtensionNotify function, 577
dumptree command, 580-585
   examining, 554
genral extensions, 554
   header files, 567-570
   Help command, 579-580
   initializing, 570-576
   KnownStructOutput function, 578, 585-589
   models, 563, 565
   requirements, 565-570
   session state changes, 577
   uninitializing, 578
   user mode extensions, 554
   versioning, 592
function execution, stepping over, 95-96
kernel mode debuggers, 32
   choosing, 44-45
   configuring, 37-41
   connecting, 39-41
   enabling Virtual PC for, 40
   event handling, 166-167
   output, 48-49
   thread suspension, 176-177
   Windows Vista, 715-716
last events, displaying, 66
memory, inspecting, 84-87
memory locations, contents, 89-90
postmortem setups, 637
processes
   attaching to, 281
   starting under, 281
   task trees, 34-35
   prompts, interpreting, 47-49
   reference releases, detecting, 107
   running processes
   attaching nonintrusively to, 36
   attaching to, 35
source files, 64-66
symbols, 49
   caches, 54-55
   loaded modules, 57-60
   paths, 52, 55-57
   reloading, 60-61
   symbol files, 49-51, 57-60
symbol servers, 52-54
utilizing, 62-64
validating, 61
target systems, discovering, 67
targets, controlling, 168-177
user mode debuggers, 30-31, 124
configuring, 34-36
event control, 133-144
event order, 131-133
event processing, 126-130
exception control, 144-166
loops, 125-126
operating system support, 124-130
output, 47
redirecting through kernel debuggers, 41-44
starting processes, 125
target creation, 124-125
u command, 72
version output, 67
Windows Vista, 712-714
value, entering, 103-104
variable values, displaying, 81-84
Windows Vista, 711
Debuggers for Windows, commands, 32-33
debbuging
64-bit debugging, 595-596, 603-624
Application Verifier, 604
custom debugger extensions, 629
Debugging Tools for Windows, 603-604
Ethereal, 605
interprocess communication, 628-629
Leak Diagnostic tool, 603
memory corruption, 625-626
security, 626-628
Application Verifier, test settings, 747
live debugging, thread state management, 172-176
LPC communication, 384-388
memory dumps, 36
multiple remote systems, 48
noninteractive processes, 118-119
without kernel mode debugger, 119-120
postmortem debugging, 631-632
Corporate Error Reporting, 682-690
dump files, 632-641, 645-652
kerneldumps, 642-644
Windows Error Reporting, 653-682
Windows Vista, 741-745
remote debugging, 109
debbug servers, 110-113
kernel servers, 113-114
process servers, 113-114
remote.exe, 109-110
source resolution, 117
symbol resolution, 115-116
scenarios, 117-118
security failures, 340-378
defered initation problems, 347-354
local security failures, 340-347
source files, managing for, 188-196
symbols, managing, 180-188
Windows Vista, security, 723-727, 729-735
Debugging a memory dump listing (2-4), 36
Debugging a service non-intrusive
listing (2-3), 36
Debugging a service process from an
elevated console listing (15-4), 714
Debugging a service process from the
command prompt listing (15-1), 712
Debugging a service process from
the normal command prompt
listing (15-2), 712
Debugging a service process from
the normal command prompt
listing (15-3), 714
Debugging Tools for Windows, 7-9, 29-30,
123, 603-604
CDB (cdb.exe) tool, 31
NTSD (ntsd.exe) tool, 31
remote.exe, 109-110
WinDbg (windbg.exe) tool, 31
DebugInfo field (RTL_CRITICAL_ SECTION structure), 498-500
Deciphering kernel global flags
listing (3-20), 150
declaring gGlobal variable, 88-89
Decoding a security descriptor using the lsd
extension command listing (7-4), 325
DeCommit option (Heaps test setting), 751
default process heaps, detailed views,
deferred initiation problems, security
problems during, 347-354
Detailed view of the default process heap
listing (6-3), 273-280
detection process, memory corruption, 201
avoidance strategies, 209
detection tools, 208
instrument source code, 208
source code analysis, 202-208
state analysis, 201-202
detection tools, memory corruption, 208
dialog boxes
Dr. Watson, 653-659
Options (LeakDialog), 6
Digital Rights Management (DRM) systems, 44, 139
directories, SDK directories, 568
Directory structure on the symbol servers
listing (4-7), 185
DirtyStacks test setting (Application
Verifier), 775
discretionary access control lists
(DACLs), 318
dispatching exceptions, 149-153
DisplayError function, 240
displaying
current call stacks, 73-79
current register values, debuggers, 68-71
debugger last events, 66
thread impersonation tokens, 329
variable values, debuggers, 81-84
Displaying information about a loaded
module listing (2-15), 58
Displaying primary token's security
descriptor listing (7-32), 370
Displaying the call stack listing (2-24), 76
Displaying the call stack of newly created
thread listing (5-3), 212
Displaying the current event handling state
listing (3-8), 135
Displaying the first three parameters used
by the five functions from the call
stack listing (2-26), 77
Displaying the module headers
listing (2-16), 59
Displaying the parameters used by the
last five functions from the call stack
listing (2-25), 76
Displaying the stack size used by the
five functions from the call stack
listing (2-27), 78
Displaying the thread from listing 736 using
a kernel mode debugger in local mode
listing (7-37), 375
Displaying the thread impersonation token
listing (7-7), 329
Displaying the thread in the RPCSS
service part of the activation path
listing (7-36), 374
Displaying the thread not impersonating
listing (7-8), 330
Distributed Common Object Model
(DCOM), 381
DllMain function, 529-537
DLLs (Dynamic Link Libraries), 529, 716-717
debugger extensions, 33
DLLs loaded in the 08cli.exe sample (after
reboot) listing (15-8), 717
DLLs loaded in the 08cli.exe sample (before
reboot) listing (15-7), 716
Dlls option (Heaps test setting), 750
double frees, memory blocks, 308-314
Dr. Watson dialog box, 199, 653-654
Application Errors section, 656-657
Application option, 656-657
Crash Dump section, 655
Crash Dump Type section, 655
Log File Path section, 654
Module List section, 658
Number of Errors to Save section, 655
Number of Instructions section, 655
Raw Stack Dump section, 660-662
Stack Back Trace section, 659
System Information section, 657
Task List section, 658
DRM (Digital Rights Management) systems, 44, 139
dt command, 82
  WOW 64 applications
  PEB, 617
  TEB, 617
dump command, 635-636
dump files, 645-646
  analysis
    access violation, 646-647
    handle leaks, 647-652
  generating
    Windows Vista, 743
    with Aplus, 639-641
    with debuggers, 634-639
  postmortem debugging, 632-634
dumpchk.exe, 8
dumping
  impersonation tokens, 351
  kernel thread information, 391
  thread state, 173
  threads, 506-507
  Dumping the impersonating token
    listing (7-22), 351
  Dumping the kernel thread information
    listing (8-9), 392
  Dumping the security descriptor for an
    object created while impersonating
    listing (7-23), 353
  Dumping the thread state listing (3-33), 173
dumptree command, implementing, 580-585
dv command, 82

errors
  DCOM (Distributed COM) errors, 354-367
  reporting, Corporate Error Reporting, 687-690
  et (exit a thread) event, creating, 144
  Ethereal, 26, 605
  Evaluating a ct event listing (3-14), 144
  Evaluating an et event listing (3-15), 144
  Evaluating an ud event listing (3-13), 143
  Event Log system, Windows Vista, 710-711
  event primitive, synchronization, 494-497
  event aliases, 134
  breakpoints, 168-170
  breaks
    adjusting, 136-140
    inspecting, 135
  controlling, user mode debuggers, 133-144
debuggers, 140-144
  epr (create a process) event, 141
et (create a thread) event, 143
  epr (exit a process) event, 142
et (exit a thread) event, 144
  ipb (initial breakpoint) event, 141
  ld (load a module) event, 142
  ud (unload a module) event, 143
  exceptions, compared, 135
  handling
    adjusting, 136-140
    inspecting, 135
  kernel mode debuggers, 166-167
  last events, displaying, 66
  order, user mode debuggers, 131-133
  processing, user mode debuggers, 126-130
  Examine explorer.exe process running
    at medium integrity level (UAC)
    listing (15-11), 729-730
  Examine one process running at system
    integrity level listing (15-10), 727, 729
  Examine the component specific
    AccessCheck performed by RPCSS
    listing (7-27), 360
  Examine the first AccessCheck performed
    by RPCSS listing (7-25), 357
  Examine the second AccessCheck performed
    by RPCSS listing (7-26), 359
Examining a registry key's object header listing (7-11), 333
Examining the process memory from a non-invasive debugger listing (3-30), 169
Examining the process object security descriptor listing (7-21), 330
Examining the thread and connection object info cell listing (8-27), 411
Example of heap handle mismatch listing (6-9), 300-305
Example of symbol paths with local cache listing (2-11), 54
Example of symbol server paths listing (2-10), 54
Example of the !handle extension command on an instance of notepad.exe listing (10-1), 495
Example run of registry enumeration application listing (5-11), 231-232
Examples of using the __cdecl and __stdcall calling conventions listing (5-15), 240-242
Exception dispatched to the user mode debugger listing (3-19), 147
exception events, processing, user mode debuggers, 129-130
exception handlers, frame-based exception handlers, 159-166
Exception handling code for a very simple function (tryexcept in 02sample.exe) listing (12-15), 622-625
exception models, Windows x64, 622-625
exceptions
   applications, utilization, 516-522
   breakpoints, 168-170
   controlling, user mode debuggers, 144-166
   dispatching, 149-153
   events, compared, 135
   life cycles, 147-149
   structured exception dispatching mechanisms, 144-153
   structures, 145-146
Exceptions test setting (Application Verifier), 747
EXCEPTION_DEBUG_EVENT, processing, 129-130
EXCEPTION_RECORD structure, 145
EXCEPTION_RECORD structure, as defined in winnt.h header listing (3-16), 145
Exception|Event|*, 137
exploratory commands, debuggers, 66-94
Exploring the impersonation token after SSPI impersonation listing (7-13), 337
extension commands, debuggers, 33
extensions
   custom 64-bit debugger extensions, writing, 629
ddebuggers, 553-554
   building, 593
   C++ binary tree implementation, 556, 559-562
code organization, 567-570
   command cancellations, 589-590
   DebugExtensionNotify function, 577
dumptree command, 580-585
   examining, 554
general extensions, 554
   header files, 567-570
   Help command, 579-580
   initialization, 570-576
   KnownStructOutput function, 578-589
   models, 563-565
   requirements, 565-570
   session state changes, 577
   uninitializing, 578
   user mode extensions, 554
   versioning, 592
F
  fastcall calling convention, 243
   fault follow-up, !analyze extension command, 706-708
FaultRate option (Heaps test setting), 752
Faults option (Heaps test setting), 752
faulty applications, analyzing, !analyze extension command, 700
Fibonacci function, source code, 75-76
fields, critical sections, direct usage, 545
file dumps, generating, tools, 632
file redirection, side effects, 601
file tree configuration, source servers, 194-196
file virtualization, Windows Vista, 732-735
FilePaths test setting (Application Verifier), 764-765
files
  binary files, debug information, 51
  Binplace.exe file, 181
  PDB files, stored information, 191
  source files, 64-66, 179
  managing, 188-196
  symbol files, 49-51, 179
  checking, 57-60
Final event for any process started under debugger listing (3-11), 142
Finding additional information about the LPC message listing (8-10), 393
Finding the PEB for a process listing (6-2), 270-272
Finding the stack that released a specific handle listing (2-43), 108
flags, KnownStructOutput function, 586
Flags field (!heap extension command), 478
Follow up ownership for scenario1.exe listing (14-4), 707
FPO (frame pointer omission)
  optimization, 74
frame pointer omission, 217
frame-based exception handlers, 159-166
freeing memory blocks, 269
FreeMem function, 305
freezing threads, 174
front end allocators, heaps, 261-263
Full option (Heaps test setting), 750
full pageheap (Heaps test setting), 750, 754, 757
function calls, call stacks, 210
function execution
  monitoring, 98
  tracing, 98
function prologs, 221
Function with five parameters, calling another function with five parameters listing (12-6), 610
functions
  AccessCheck, 323-324
  calling conventions, 243
  cdecl, 240-243
 fastcall, 243
  stdcall, 240-243
  thiscall, 243
  CallProc, 250-255
  DebugExtensionNotify, 577
  DisplayError, 240
  DllMain, 529-537
  execution, stepping over, 95-96
  FreeMem, 305
  InitModule, 245, 250-255
  InOrderTraversal, 582
  KnownStructOutput, 578
  KnownStructOutput function, implementing, 585-589
  Sum, 217, 219
  ThreadProcedure, 212-213
G
general extensions, debuggers, 554
Generated code for a simple function using _try/__except support listing (3-25), 162
generating
dump files
  Windows Vista, 743
  with ADPlus, 639-641
  with debuggers, 634-639
file dumps, tools, 632
public symbols, 180-183
Generating annotated assembly file from the source file listing (3-27), 165
!getcallinfo extension command, 404-406
!getdbgecell extension command, 406-407
!getendpointinfo extension command, 400-403
Getting more details about the call target listing (8-25), 410
Getting more details from the client cell info listing (8-24), 410
Getting the call info from the endpoint information listing (8-26), 411
gflags, 16
   Command Line mode, 19-21
   GUI mode, 16-18
   option abbreviations, 19-20

gflags.exe, 8

gGlobal declaration listing (2-36), 88

gGlobal variable, declaring, 88-89

Global Flags, 16
   Command Line mode, 19-21
   GUI mode, 16-18
   option abbreviations, 19-20

GP (Group Policy) settings, Corporate Error Reporting, 683-687

GUI mode (gflags), 16-18

GUI view (Process Explorer), 22

H
   -h parameter, 137
   !handle extension command, 496-497, 502-503

handle leaks
   analyzing, Debug Diagnostic tool, 693-697
dump files, analysis, 647-652

handles
   handle leaks, 434-460
   injection, 455-459
   mismatches, heaps, 300-305

Handles test setting (Application Verifyer), 747-748

handling events
   adjusting, 136-140
   inspecting, 135

   kernel mode debuggers, 166-167

hang rule (Debug Diagnostic Tool), 692

header files, debugger extensions, 567-570

headers
   kernel object headers, obtaining, 331
   registry key object headers, examining, 332

Heap block address field (!heap extension command), 477

heap coalescing, 268

Heap corruption analysis using the heap debugger command listing (6-7), 289-294

heap corruptions, 281
   handle mismatches, 300-305
   heap overruns, 286-300
   heap reuse after deletion, 306-314
   heap underruns, 286-300
   uninitiated state, 282-286

!heap extension command
   heap searching, 480-484
   heap statistics, 475-480
   leak detection, 484-485
   memory leaks, identifying, 474-491

Pageheap, 485-486

heap manager (Windows), 261, 717-723

heap searching, !heap extension command, 480-484

heap statistics, !heap extension command, 475-480

Heap-based string copy application
   listing (6-6), 287-289

heaps, 259-260
   back end allocators, 263-281
   default process heap, detailed view, 273-280
   front end allocators, 261-263
   heap coalescing, 268
   low fragmentation heaps, 718-721
   memory corruption, 625-626
   segments, layout, 266
   subsegments, 721

Heaps test setting (Application Verifyer), 749-757

   full pageheap, 750, 754, 757
   normal pageheap, 750, 753, 757

Help command, implementing, 579-580

helper commands, debuggers, 104-107

Heuristic used by debugger to find the symbol file listing (2-9), 52

HighVersionLie test setting (Application Verifyer), 765-767

Host component (Debug Diagnostic Tool), 692

How to Freeze or Unfreeze threads
   listing (3-35), 175

How to suspend and resume threads
   listing (3-34), 174

!htrace extension command, handle injection, 455-459
htrace leak detection tool, 432
HTTP servers, public symbols, storing on, 187-188

I–J
ibp (initial breakpoint) event, creating, 141
identifying
memory leaks, 462-464
leak detection tools, 465-491
potential resource leaks, 425-430
Identifying Rprocess and DcomLaunch services on Windows XP SP2 listing (7-24), 356
Identifying the caller listing (7-29), 366
Identifying the owning thread of the problematic critical section listing (10-13), 527
Identifying the thread identity listing (7-34), 372
ImpersonateSelf invocation, debugger targets, simulating, 340-341
impersonation
DCOM calls, 395-396
LRPC calls, 395-396
security implications, 354
impersonation levels, client server applications, 337-338
impersonation tokens, 351
information
calls, attaining, 404-407
cell debugging, 398-412
accessing, 399-408
client calls, attaining, 407-408
endpoints, attaining, 400-402
source information gathering, 188-191
using, 192-193
threads, attaining, 402-403
information sources (security), 328
access tokens, 328-330
SDs (security descriptors), 330-333
Information stored in the PDB file listing (4-8), 191
Initial breakpoint stack trace for any process started under debugger listing (3-10), 141
initialization
COM interfaces, 572-573
debugger extensions, 570-576
type information, 574-575
version information, 572
WinDbg extension, extension data, 576
Initialization of type information listing (11-4), 575
Initializing the WinDbg extension data listing (11-5), 577
InitModule function, 245, 250-255
injection, handles, 455-459
InOrderTraversal function, 582
input parameters, local variables, compared, 84
Inspecting the command line and the identity of the server about to be started listing (7-28), 363
Inspecting the security descriptor of another thread object running in the same process listing (7-19), 346
Inspecting the security descriptor of the thread object running the failing code listing (7-18), 345
installation, WDK (Windows Driver Kit), 24
instrument source code, memory corruption, 208
integrity levels, Windows Vista, 724
InteractiveServices test setting (Application Verifier), 767-768
interprocess communication
64-bit debugging, 628-629
remote authentication, 423
Windows Vista, 736
interprocess communications, 379-380
communication mechanisms, 380-382
local communications, troubleshooting, 382-396
network traffic, analyzing, 413-421
remote communications, troubleshooting, 396-422
RPC extended error information, 424-425
Investigating x86 exception handler list listing (3-23), 160
K
k command, 78
KD (kd.exe) tool (Debugger Tools for Windows), 8, 32
kdbgctrl.exe, 8
kdsrv.exe, 8
kernel, thread information, dumping, 391
dumping, 391
kernel dumps, creating, 642-644
creating, 642-644
kernel mode debuggers, configuring, 37-41
configuring, 37-41
Kernel mode debugger output
output, 48-49
listing (2-7), 48
listing
kernel mode debuggers, 32
32
choosing, 44-45
code breakpoints, setting, 80
setting, 80
connecting, 39-41
connecting, 39-41
events, handling, 166-167
handling, 166-167
output, 48-49
output, 48-49
threads, suspending, 176-177
suspending, 176-177
user mode debuggers, redirecting through, 41-44
redirecting through, 41-44
Virtual PC, enabling for, 40
enabling for, 40

kernel objects
object headers, obtaining, 331
obtaining, 331
SDs (security descriptors), obtaining, 331
obtaining, 331
kernel servers, remote debugging, 113-114
remote debugging, 113-114
KernelModeDriverInstall test setting
setting (Application Verifier), 768-769

kill.exe, 8

KnownStructOutput function, 578
flags, 586
implementing, 585-589

L
LAL (look aside list) front end allocator,
front end allocator,
heaps, 261
heaps, 261
last events, debuggers, displaying, 66
displaying, 66
last in first out (LIFO) semantics,
semantics,
algorithms, 209
algorithms, 209
. lastevent command, 66
. lastevent output listing (2-17), 67
listing
layouts, heap segments, 266
segments, 266
ld (load a module) event, creating, 142
creating, 142
leak detection tools, 432-433, 465
432-433, 465
!heap extension command, 474-491
474-491
Leak Diag, 4-6, 432
4-6, 432
LeakDiag tool, 603
603
leak tracker component (Debug Diagnostic Tool), 692
Tool), 692
LeakDiag tool, 4-6, 432
4-6, 432
allocators, 4-5
4-5
memory leaks, identifying, 470-474
identifying, 470-474
Options dialog box, 6
6
Stat screen, 5
5
UMDH.exe, 4
4
leaks
detecting
 detecting
PUT MEMORY LEAKS and HANDLE LEAKS, 693
PUT MEMORY LEAKS and HANDLE LEAKS, 693
resource leaks, 430-431
430-431
analysis process, 428-433
428-433
avoidance strategies, 433
433
handle leaks, 434-460
434-460
leak detection tools, 432-433
432-433
memory leaks, 460-466, 468-492
460-466, 468-492
reproducibility, 433-434
reproducibility, 433-434
leaks rule (Debug Diagnostic Tool), 692
692
LF (low fragmentation) front end allocators,
front end allocators, 261
heaps, 261
libraries, Microsoft Detours, 7
7
life cycles, exceptions, 147-149
147-149
LIFO (last in first out) semantics,
semantics,
algorithms, 209
algorithms, 209
link.exe utility, binary files, finding debug information in, 51
51
listing
listing
processes, task trees, 34-35
34-35
thread summary information, 391
391
Listing all processes as task tree
Listing all processes as task tree
listing (2-1), 34
34
Listing all symbol files unused since a specific date listing (2-12), 55
55
Listing all the interfaces registered on the local system, identified by \.
Listing all the interfaces registered on the local system, identified by \.
listing (8-31), 426
426
Listing all the interfaces registered on \PIPE\winreg endpoint on the local syste
Listing all the interfaces registered on \PIPE\winreg endpoint on the local syste
listing (8-30), 425
425
Listing of all threads in the culprit process listing (10-16), 532
532
Listing thread summary information
listing (8-8), 391
listings
2-1 (Listing all processes as task tree), 34
2-2 (Options for attaching the debugger to a running process), 35
2-3 (Debugging a service non-intrusive), 36
2-4 (Debugging a memory dump), 36
2-5 (Switching from user mode to kernel mode debugger), 43
2-6 (User mode debugger output), 47
2-7 (Kernel mode debugger output), 48
2-8 (Using the link.exe utility to find debug information stored in the binary file), 51
2-9 (Heuristic used by debugger to find the symbol file), 52
2-10 (Example of symbol server paths), 54
2-11 (Example of symbol paths with local cache), 54
2-12 (Listing all symbol files unused since a specific date), 55
2-13 (Two methods of setting up the symbol path at debugger startup), 55
2-14 (Using the .sympath and .symfix commands), 56
2-15 (Displaying information about a loaded module), 58
2-16 (Displaying the module headers), 59
2-17 (.lastevent output), 67
2-18 (Version output from a user mode debugger), 67
2-19 (Registers value using the default register mask), 69
2-20 (Changing the default register mask), 69
2-21 (Pseudo-register used on user mode debugger break (x86)), 71
2-22 (u command used in user mode debugger (x86)), 72
2-23 (Source of Fibonacci function implemented in the 02sample.exe sample), 75
2-24 (Displaying the call stack), 76
2-25 (Displaying the parameters used by the last five functions from the call stack), 76
2-26 (Displaying the first three parameters used by the five functions from the call stack), 77
2-27 (Displaying the stack size used by the five functions from the call stack), 78
2-28 (Manual stack reconstruction using the k command), 78
2-29 (Using breakpoints in the user mode debugger), 79
2-30 (Using breakpoints in the kernel mode debugger), 80
2-31 (Using breakpoints in the user mode debugger), 80
2-32 (Use of dv command), 82
2-33 (Use of dt command), 83
2-34 (Use of d command), 85
2-35 (Use of d's command), 87
2-36 (gGlobal declaration), 88
2-37 (Typical use of the ba command), 89
2-38 (!address debugger command example), 90
2-39 (!address command), 91
2-40 (Obtaining the process PEB), 92
2-41 (Obtaining the thread TEB), 93
2-42 (Trace and watch function execution), 98
2-43 (Finding the stack that released a specific handle), 108
2-44 (Remoting the console using remote.exe), 109
2-45 (Starting the debugger server), 111
3-1 (Sample code used to start a process under user mode debugger), 125
3-2 (Standard user mode debugger loop), 126
3-3 (Simple debugger events processing), 127
3-4 (Processing output debug string event), 128
3-5 (Read a specific length string from the debugger target space), 128
3-6 (Processing exception debug event), 130
3-7 (Debugger events generated by a simple process execution (xcopy.exe)), 131
3-8 (Displaying the current event handling state), 135
3-9 (Tools.ini content), 140
3-10 (Initial breakpoint stack trace for any process started under debugger), 141
3-11 (Final event for any process started under debugger), 142
3-12 (stack trace after loading a dynamic link library), 142
3-13 (Evaluating an ud event), 143
3-14 (Evaluating a ct event), 144
3-15 (Evaluating an et event), 144
3-16 (EXCEPTION_RECORD structure, as defined in winnt.h header), 145
3-17 (CONTEXT structure, as defined in MSDN), 146
3-18 (x86 context flags values), 146
3-19 (Exception dispatched to the user mode debugger), 147
3-20 (Deciphering kernel global flags), 150
3-21 (Changing kernel flags using command line gflags.exe), 150
3-22 (Code exercising the exception dispatching logic), 154
3-23 (Investigating x86 exception handler list), 160
3-24 (Simple function using __try/__except constructs), 162
3-25 (Generated code for a simple function using __try/__except support), 162
3-26 (Assembly listing generated for the function from Listing 324), 163
3-27 (Generating annotated assembly file from the source file), 165
3-28 (Thread environment block on two different threads in the same process), 166
3-29 (Using kls flag for detecting a user mode module mapping), 167
3-30 (Examining the process memory from a noninvasive debugger), 169
3-31 (Debug registers on a normal processor), 171
3-32 (Simulating code tracing after attaching to a running project), 172
3-33 (Dumping the thread state), 173
3-34 (How to suspend and resume threads), 174
3-35 (How to Freeze or Unfreeze threads), 175
3-36 (Changing the current thread), 176
3-37 (Simulating a kernel32!Sleep call), 177
4-1 (Binplace file content), 181
4-2 (Binplacing the symbol files), 182
4-3 (Binary folder tree), 182
4-4 (Debug directories immediately after building the binaries), 183
4-5 (Debug directories after bin place operation), 183
4-6 (Command to store the symbols on the symbol servers), 184
4-7 (Directory structure on the symbol servers), 185
4-8 (Command to delete the symbol files from the symbol servers), 187
4-9 (Information stored in the PDB file), 191
4-10 (SourceServer information stored in the PDB file), 191
4-11 (Source server file tree configuration), 194
5-1 (Simple console-based application that simulates a memory corruption), 203-207
5-2 (Sample application showing the creation of a new thread), 210-211
5-3 (Displaying the call stack of newly created thread), 212
5-4 (Assembly code of the ThreadProcedure function), 213
5-5 (Assembly code for ProcA), 215-216
5-6 (Preamble assembly code for calling the Sum function), 217
5-7 (Assembly instructions right after our call to Sum), 219
5-8 (ProcA function epilog), 220
5-9 (Application that establishes a connection to a data source), 223
5-10 (Crash reproduced in the debugger), 225
5-11 (Example run of registry enumeration application), 231-232
5-12 (Access violation enumerating registry values), 232-234
5-13 (Contents of the stack at the point of crash), 235
| 5-14 | (Walking the stack back in time), 236-239 |
| 5-15 | (Examples of using the __cdecl and __stdcall calling conventions), 240-242 |
| 5-16 | (Simple application that declares a number of functions), 244-245 |
| 5-17 | (Application that calls a function in a DLL), 245-247 |
| 5-18 | (Attempt at manually constructing the stack), 248-255 |
| 6-1  | (Simple application that performs heap allocations), 270 |
| 6-2  | (Finding the PEB for a process), 270, 272 |
| 6-3  | (Detailed view of the default process heap), 273-270 |
| 6-4  | (Simple application that uses uninitialized memory), 282-283 |
| 6-5  | (Application crash seen under the debugger), 283-286 |
| 6-6  | (Heap-based string copy application), 287-289 |
| 6-7  | (Heap corruption analysis using the heap debugger command), 289-294 |
| 6-8  | (Application verifier reported heap block corruption), 295-300 |
| 6-9  | (Example of heap handle mismatch), 300, 302-305 |
| 6-10 | (Simple example of double free), 308-312, 314 |
| 7-1  | (SID structure definition), 319 |
| 7-2  | (Sample code exercising the AccessCheck function), 323-324 |
| 7-3  | (Binary representation of a security descriptor), 324 |
| 7-4  | (Decoding a security descriptor using the !sd extension command), 325 |
| 7-5  | (Using the !token extension command to display a token in the user mode debugger), 326 |
| 7-6  | (Obtaining the process access token), 329 |
| 7-7  | (Displaying the thread impersonation token), 329 |
| 7-8  | (Displaying the thread not impersonating), 330 |
| 7-9  | (Obtaining the object header for a kernel object), 331 |
| 7-10 | (Obtaining kernel objects security descriptor), 332 |
| 7-11 | (Examining a registry key’s object header), 333 |
| 7-12 | (Tracing the remote authentication from the server process), 336 |
| 7-13 | (Exploring the impersonation token after SSPI impersonation), 337 |
| 7-14 | (Sample function calling GetComputerNameEx at different impersonation levels), 338 |
| 7-15 | (Simulating ImpersonateSelf invocation in the debugger target), 341 |
| 7-16 | (Obtaining the process object security descriptor), 343 |
| 7-17 | (Obtaining the primary token object security descriptor), 344 |
| 7-18 | (Inspecting the security descriptor of the thread object running the failing code), 345 |
| 7-19 | (Inspecting the security descriptor of another thread object running in the same process), 346 |
| 7-20 | (Sample initialization function), 348 |
| 7-21 | (Examining the process object security descriptor), 350 |
| 7-22 | (Dumping the impersonating token), 351 |
| 7-23 | (Dumping the security descriptor for an object created while impersonating), 353 |
| 7-24 | (Identifying Rpccss and DcomLaunch services on Windows XP SP2), 356 |
| 7-25 | (Examine the first AccessCheck performed by RPCSS), 357 |
| 7-26 | (Examine the second AccessCheck performed by RPCSS), 359 |
| 7-27 | (Examine the component specific AccessCheck performed by RPCSS), 360 |
| 7-28 | (Inspecting the command line and the identity of the server about to be started), 363 |
| 7-29 | (Identifying the caller), 366 |
7-30 (Watching the error code returned by
OpenThreadToken/
OpenProcessToken, 369
7-31 (Obtaining the token information using
local KD), 370
7-32 (Displaying primary token’s security
descraper), 370
7-33 (client stack containing the activation
call), 372
7-34 (Identifying the thread identity), 372
7-35 (Stopping the DcomLaunch code after
impersonating the client), 373
7-36 (Displaying the thread in the RPCSS
service part of the activation path), 374
7-37 (Displaying the thread from Listing
736 using a kernel mode debugger in
local mode), 375
8-1 (Client’s thread waiting on LPC request
to complete), 384
8-2 (Using !lpc extension to get message
information), 385
8-3 (Using !lpc extension to get port
information), 386
8-4 (Using !lpc extension to obtain the
entire LPC activity on the system), 386
8-5 (Starting the client and listing a partial
call stack for each thread), 388
8-6 (Typical stack of clients using DCOM
over LRPC), 389
8-7 (Typical stack of a server thread waiting
for a new request on DCOM over
LRPC), 390
8-8 (Listing thread summary
information), 391
8-9 (Dumping the kernel thread
information), 392
8-10 (Finding additional information about
the LPC message), 393
8-11 (Server’s thread processing the LPC
message), 394
8-12 (Reading ImpersonationInfo
ImpersonationInfo stored on the server
thread), 395
8-13 (Using !rptime to obtain the current
time stamp used by troubleshooting
infrastructure), 400
8-14 (Using !getendpointinfo to list all
domain known by RPC), 401
8-15 (Using !getendpointinfo to list all
domain known by RPC), 402
8-16 (Using !getthreadinfo to list all thread
from RPC thread pool), 402
8-17 (Using !getthreadinfo to obtain a
specific thread RPC information), 403
8-18 (Using !getcallinfo to obtain the
call information maintained by the
server), 405
8-19 (Using !getcallinfo to filter call
information to a specific process), 405
8-20 (Using !getdbgcell to obtain the
call information maintained by the
server), 406
8-21 (Using !getclientcallinfo to obtain the
call information maintained by the
client), 408
8-22 (Typical client stack waiting on remote
call made using a connection-based
protocol), 409
8-23 (Enumerating all the client call info
cells), 409
8-24 (Getting more details from the client
cell info), 410
8-25 (Getting more details about the call
target), 410
8-26 (Getting the call info from the
endpoint information), 411
8-27 (Examining the thread and connection
object info cell), 411
8-28 (Server thread call stack), 412
8-29 (Server breakpoints encountered using
SSPI), 423
8-30 (Listing all the interfaces registered
on \PIPE\winreg endpoint on the local
system), 425
8-31 (Listing all the interfaces registered on
the local system, identified by \\), 426
10-1 (Example of the !handle extension
command on an instance of
notepad.exe), 495
10-3 (Using the \es command to list all
critical sections of a process), 501
10-4 (Using the !handle extension command to find mutex object in notepad.exe), 502
10-5 (Using the *kb command to Dump all threads and associated stacks), 506
10-6 (Sample application that results in a deadlock), 510
10-7 (All thread stacks currently running in the process), 512
10-8 (Sample application that utilizes exceptions), 516
10-9 (Sample application thread state), 519
10-10 (Critical section class that handles the lifetime of a critical section), 521
10-11 (Simple application utilizing the TerminateThread API), 523
10-12 (Thread list and associated stacks of hung application), 525
10-13 (Identifying the owning thread of the problematic critical section), 527
10-16 (Listing of all threads in the culprit process), 532
10-17 (Sample application that suffers from lock convoys), 540
10-18 (Unlocked Critical Section), 543
10-19 (Locked Critical Section with no Waiters), 543
10-20 (Locked Critical Section with Waiters), 544
10-21 (Properly initialized critical section), 546
11-1 (Simple C++ Binary Tree Implementation), 556, 559
11-2 (BSTREE.EXE Using the Binary Tree Implementation), 559-562
11-3 (Debugger COM interfaces initialization), 573
11-4 (Initialization of type information), 575
11-5 (Initializing the WinDbg extension data), 577
12-1 (Native stack on a WOW64 process), 599
12-2 (x86-64-bit general purposes register), 606
12-3 (True 32 bit Stack in WOW64 Process (hidden in Listing 121)), 608
12-4 (Obtaining WOW64 structures), 608
12-5 (Unassembly code is dependent on the processor execution mode), 609
12-6 (Function with five parameters, calling another function with five parameters), 610
12-7 (Assembly code representing the Function5 function), 610
12-8 (Call-stack function parameter), 612
12-9 (Unassembled non-optimized function), 613
12-10 (Local variable), 614
12-11 (WOW64 applications PEB), 615, 617
12-12 (WOW64 application's PEB using the dt command), 617
12-13 (WOW64 threads' TEB using !teb extension command and dt commands), 618
12-14 (Debugger events generated a WOW64 process execution (xcopy.exe)), 620-622
12-15 (Exception handling code for a very simple function (tryexcept in 02sample.exe)), 622, 624-625
12-16 (Security information on Windows x64), 626
13-1 (Simple crashing application), 633
13-2 (Creating a kernel mode dump file), 644
14-1 (DebugDiag custom analysis script metadata), 697
14-2 (Using the CritSec object), 698
14-3 (Output of the !analyze extension command), 701-703
14-4 (Follow up ownership for scenario1.exe), 707
15-1 (Debugging a service process from the command prompt), 712
15-2 (Debugging a service process from the normal command prompt), 712
15-3 (Debugging a service process from the normal command prompt), 714
15-4 (Debugging a service process from an elevated console), 714
15-5 (Enumerating all BCD objects from an elevated console), 715
15-6 (Configuring the kernel mode debugger for the running configuration), 716
15-7 {DLLs loaded in the 08cli.exe sample (before reboot)}, 716
15-8 {DLLs loaded in the 08cli.exe sample (after reboot)}, 717
15-9 (Application manifest requesting a high integrity level), 724
15-10 (Examine one process running at system integrity level), 727, 729
15-11 {Examine explorer.exe process running at medium integrity level (UAC), 729-730
15-12 (Changing file integrity level using built in icacls.exe tool), 731
15-13 (Setting and retrieving application settings from UAC administrator), 732
15-14 (Retrieving application settings from an elevated prompt), 733
live debugging, thread state management, 172-176
loaded modules, checking, 57-60
local communication, troubleshooting, 382-396
Local Procedure Call (LPC), 381
local security failures, investigating, 340-344, 347
Local variable listing (12-10), 614
local variables
input parameters, compared, 84
values, discovering, 613-614
lock contention, synchronization, 538-545
LockCount field (RTL_CRITICAL_SECTION structure), 498
Locked Critical Section with no Waiters listing (10-19), 543
Locked Critical Section with Waiters listing (10-20), 544
Locks test setting (Application Verifier), 757-759
LockSemaphore field (RTL_CRITICAL_SECTION structure), 499
Log File Path section (Dr. Watson dialog box), 654
logger.exe, 8
logviewer.exe, 8
look aside list (LAL) front end allocators, heaps, 261
loops, user mode debuggers, 125-126
low fragmentation (LF) front end allocator, heaps, 261
low fragmentation heaps, 718-721
Low Resource Simulation test setting (Application Verifier), 769-770
LPC (Local Procedure Call), 381
!lpc extension command, 385-386
LPC protocol, 382-383
debugging communication, 384-388
development of, 383-384
LRPC calls, impersonating, 395-396
LuaPriv test setting (Application Verifier), 771-774
M
managing critical sections, 545-550
Manual stack reconstruction using the k command listing (2-28), 78
manually constructing, stacks, 247, 249-252
mapping binaries to products, Windows Error Reporting, 673-677
memory
heaps
back end allocators, 263-281
front end allocators, 261-263
inspecting, 84-87
kernel dumps, creating, 642-644
memory leaks, 460-492
avoidance strategies, 491-492
identifying, 462-464
leak detection tools, 465-491
process memory, inspecting, 615
memory blocks
allocating, 269
double frees, 308-314
freeing, 269
memory corruption, 199-201, 259
application crashes, 200
detection process, 201
avoidance strategies, 209
detection tools, 208
instrument source code, 208
source code analysis, 202-208
state analysis, 201-202
heap corruptions, 281

handle mismatches, 300-305
heap overruns, 286-300
heap reuse after deletion, 306-314
heap underruns, 286-300
uninitiated state, 282-286
heaps, 625-626
stack corruptions, 209-222
asynchronous operations, 231-240
avoidance strategies, 255-258
calling conventions mismatch, 240-255
stack overruns, 223-231
stack pointers, 231-240
stacks, 625
unpredictable behavior, 200
memory dumps, debugging, 36
memory leak detection
LeakDiag (Leak Diagnosis) tool, 4-6
UMDH, 9
memory leaks, 460-492
analyzing, Debug Diagnostic tool, 693-697
avoidance strategies, 491-492
identifying, 462-464
leak detection tools, 465
heap extension command, 474-491
LeakDiag, 470-474
UMDH, 465-469
memory locations, contents, displaying, 89-90
Memory test setting (Application Verifier), 760-762
Microsoft Application Verifier, 10-16
Microsoft Detours, 7
mismatches, handles, heaps, 300-305
models, debugger extensions, 563-565
Module List section (Dr. Watson dialog box), 658
modules, loaded modules, checking, 57-60
monitoring function execution, 98
mov edi,edi instruction, 222
MSRPC, communication, debugging, 388-396
multiple remote systems, debugging, 48
multithreading, 493, 550
mutex kernel mode synchronization construct, 502-504
N
naming conventions
ADDRESS objects, 390
CCALL objects, 389
Native stack on a WOW64 process
listing (12-1), 599
network protocols, analyzing, Ethereal, 26
network traffic, analyzing, 413-421
NoLock option (Heaps test setting), 751
noninteractive processes, debugging, 118-119
without kernel mode debugger, 119-120
normal pageheap (Heaps test setting), 750-753, 757
npipe protocol, 111
NTSD (ntsd.exe) tool (Debugging Tools for Windows), 31
ntsd.exe, 8
Number of Errors to Save section (Dr. Watson dialog box), 655
Number of Instructions section (Dr. Watson dialog box), 655
NX enabled systems, Windows support for, 255
O
object headers
kernel objects, obtaining, 331
registry keys, examining, 332
object security, 317-319
access tokens, 325-328
ACLs (Access Control Lists), 320-322
client server applications
impersonation levels, 337-338
remote authentication, 335-337
security support provider interface support, 335-337
token propagation, 334-338
SDs (security descriptors), 322-325
security checks, 334
system boundaries, 338-340
security failures
deferred initiation problems, 347-354
investigating, 340-378
local security failures, 340-347
security information sources, 328
access tokens, 328-330
SDs (security descriptors), 330-333
SIDs (security identifiers), 319-320, 325-328
types, 327
well-known SIDs, 327
object security descriptors, process object
security descriptors, 342-344
objects
ADDRESS objects, naming conventions, 390
BCD (Boot Configuration Data), 715-716
CCALL objects, naming conventions, 389
CritSecInfo, 697-698
Obtaining kernel objects security descriptor
listing (7-10), 332
Obtaining the object header for a kernel
object listing (7-9), 331
Obtaining the primary token object security
descriptor listing (7-17), 344
Obtaining the process access token
listing (7-6), 329
Obtaining the process object security
descriptor listing (7-16), 343
Obtaining the process PEB listing (2-40), 92
Obtaining the thread TEB listing (2-41), 93
Obtaining the token information using local
KD listing (7-31), 370
Obtaining WOW64 structures
listing (12-4), 608
one-time initialization APIs,
synchronization, 740
operating systems
64-bit operating systems, 595-598
user mode debuggers, support, 124-130
Windows Vista, 709
custom debugger extensions, 741
debuggers, 711-717
debugging security, 723-735
Event Log system, 710-711
heap manager, 717-723
interprocess communication, 736
one-time initialization, 740
postmortem debugging, 741-745
resource leaks, 736
synchronization, 737-740
tools, 710
user access control side effects, 712-714
Options dialog box (LeakDialog), 6
Options for attaching the debugger to a
running process listing (2-2), 35
order, events, user mode debuggers, 131-133
orphaned critical sections, 516-529
output
!analyze extension command, 701-703
kernel mode debuggers, 48-49
user mode debuggers, 47
version output, 67
Output of the !analyze extension command
listing (14-3), 701-703
OUTPUT_DEBUG_STRING_EVENT,
processing, 127-129
overruns
heaps, 286-300
stacks, 223-231
OwningThread field (RTL_CRITICAL_ SECTION structure), 499
P
p command, 95-96
Pageheap, !heap extension command,
485-486
paths, symbols, 52
setting, 55-57
PDB files, stored information, 191
PEB (process environment block)
locating, 270, 272
obtaining, 93
WOW64 applications, 615
dt command, 617
pointers (stacks), 231-240
postmortem debugging, 631-632
Corporate Error Reporting, 682-683
configuring, 683-687
error reporting, 687-690
GP (Group Policy) settings, 683-687
dump files, 632-634, 645-646
access violation analysis, 646-647
generating, 634-641
handle leak analysis, 647-652
kernel dumps, creating, 642-644
Windows Error Reporting, 653-662
architecture, 662-682
Windows Vista, 741-745
Preamble assembly code for calling the Sum
function listing (3-6), 217
Previous size field (heap extension
command), 478
primary token object security descriptor,
obtaining, 344
primitives, synchronization, event primitive,
494-497
PrintAPI test setting (Application
Verifier), 776
PrintDriver test setting (Application
Verifier), 776
private symbols, 591
ProcA function epilog listing (5-8), 220
ProcA procedure, 214, 220
assembly code, 214-216
procedures, ProcA, 214, 220
assembly code, 214-216
process access tokens, obtaining, 329
Process Explorer, 22-23
process health, analyzing, Process Explorer,
22-23
process memory, inspecting, 615
Process Monitor, 23
process object security descriptors
examining, 350
obtaining, 342-344
process servers, remote debugging, 113-114
processes
64-bit processes, 595-596
debuggers, attaching to, 281
interprocess communication
64-bit debugging, 628-629
Windows Vista, 736
interprocess communications, 379-380
communication mechanisms, 380-382
local communications, 382-396
network traffic analysis, 413-421
remote communications, 396-422
RPC extended error information, 424-425
listing task trees, 34-35
noninteractive processes, 118-119
without kernel mode debugger, 119-120
PEB (process environment block)
locating, 270, 272
obtaining, 93
running processes
attaching debuggers nonintrusively to, 36
attaching debuggers to, 35
starting under debuggers, 281
user mode debuggers, starting, 125
Processing exception debug event
listing (3-6), 130
Processing output debug string event
listing (3-4), 128
processors
code execution, 72
code processing, discovering, 609
tracing, 172
product rollup page, Windows Error
Reporting, 672
programming errors, memory corruption,
199-200
application crashes, 200
detection process, 201-209
stack corruptions, 209-258
unpredictable behavior, 200
prologs, functions, 221
prompts, debuggers, interpreting, 47-49
propagation, token propagation, client server
applications, 334-338
Properly initialized critical section
listing (10-21), 546
Protect option (Heaps test setting), 751
protocols
communication protocols, 380-382
LPC protocol, 382-384
npipe protocol, 111
TCP, 112-113
Pseudo-register used on user mode
debugger break (x86) listing (2-21), 71
public symbols, 591
generating, 180-183
HTTP servers, storing on, 187-188
Q–R
Random option (Heaps test setting), 750
RandRate option (Heaps test setting), 751
Raw Stack Dump section (Dr. Watson dialog
box), 660-662
Read a specific length string from the debugger target space listing (3-5), 128
reader/writer locks, synchronization, 737-738
Reading ImpersonationInfo
ImpersonationInfo stored on the server thread listing (8-12), 395
RecursionCount field (RTL_CRITICAL_SECTION structure), 499
redirecting user mode debuggers through kernel debuggers, 41-44
redirection, files, side effects, 601
reference releases, detecting, 107
register values
current register values, finding, 605-607 debuggers, displaying, 68-71
Registers value using the default register mask listing (2-19), 69
registry keys, object headers, examining, 332
registry virtualization, Windows Vista, 732-735
reloading symbols, 60-61
remote authentication, 423
client server applications, 335-337
remote communication
cell paths, breaking, 421-422 remote authentication, 423
RPC extended error information, 424-425 troubleshooting, 396-422
remote debugging, 109
debug servers, 110-113
kernel servers, 113-114
process servers, 113-114
remote.exe, 108-110
source resolution, 117
symbol resolution, 115-116
remote systems, multiple remote systems, debugging, 48
remote.exe, 8
remote.exe utility, remote debugging, 109-110
Remoting the console using remote.exe listing (2-44), 109
reporting errors, Corporate Error Reporting, 687-690
reproducibility, resource leaks, 433-434
requirements, debugger extensions, 565-570
resolving sources, remote debugging, 117
symbols, remote debugging, 115-116
resource leaks, 427, 430-431
analysis process, 428-433 avoidance strategies, defining, 433
handle leaks, 434-460
leak detection tools, 432-433
memory leaks, 460-492
avoidance strategies, 491-492
identifying, 462-464
leak detection tools, 465-491
potential resource leaks, identifying, 428-430
PUT MEMORY LEAKS and HANDLE LEAKS, 693 reproducibility, 433-434
Windows Vista, 736
resources, 427-428
results, analyzing, !analyze extension command, 701, 703-706
resuming threads, 174
Retrieving application settings from an elevated prompt listing (15-14), 733
reuse, heaps after deletion, 306-314
rollup page, Windows Error Reporting, 672
RPC troubleshooting state information, 396
cell debugging
configuring, 396-398
information, 398-412
extended error information, 424-425
!rpctime extension command, 400
rtlist.exe, 8
RTL_CRITICAL_SECTION structure, 498
running processes, debuggers
attaching nonintrusively to, 36 attaching to, 35
S
Sample application showing the creation of a new thread listing (5-2), 210-211
Sample application that results in a deadlock listing (10-6), 510
Sample application that suffers from lock convoys listing (10-17), 540
Sample application that utilizes exceptions listing (10-8), 516
Sample application thread state listing (10-9), 519
Sample code exercising the AccessCheck function listing (7-2), 323-324
Sample code used to start a process under user mode debugger listing (3-1), 125
Sample function calling GetComputerNameEx at different impersonation levels listing (7-14), 338
Sample initialization function listing (7-20), 348
scenarios, debugging scenarios, 117-118
scripts, custom analysis scripts, authoring, 697-699
!sd extension command, 324
SDDs (security descriptor definition language), 323
SDK directories, 568
SDs (security descriptors), 322-325, 330-333
kernel object SD, obtaining, 331
primary token object security descriptor, obtaining, 344
process object security descriptors examining, 350
obtaining, 342-344
security, 317
!token extension commands, failures, 368-371
64-bit debugging, 626, 628
client server applications impersonation levels, 337-338
remote authentication, 333-337
security support provider interface support, 335-337
token propagation, 334-338
DCOM (Distributed COM) errors, 354-367
debugging, Windows Vista, 723-735
impersonation, implications, 354
information sources, 328
access tokens, 325-330
SDs (security descriptors), 330-333
object security, 317-319
access tokens, 325-328
ACLs (Access Control Lists), 320-322
SDs (security descriptors), 322-325
SDs (security identifiers), 319-320, 325-328
security checks, system boundaries, 338-340
security failures
deferred initiation problems, 347-354
investigating, 340-378
local security failures, 340-344, 347
security checks, 334
security checks, 334
security descriptor definition language (SDDL), 323
security failures, investigating, 340-378
deferred initiation problems, 347-354
local security failures, 340-347
Security information on Windows x64 listing (12-16), 626
Security Reference Monitor. See SRM (Security Reference Monitor)
security support provider interface support, client server applications, 335-337
segments, heaps, layout, 266
semaphore kernel mode synchronization object, 504
sending error information, importance of, 664
Server breakpoints encountered using SSPI listing (8-29), 423
Server thread call stack listing (8-28), 412
Server's thread processing the LPC message listing (8-11), 394
servers debug servers, 110-113
kernel servers, remote debugging, 113-114
process servers, remote debugging, 113-114
symbol servers, 52-54
Service component (Debug Diagnostic Tool), 692
session states, debuggers changes, 577
DebugExtensionNotify function, 577
setting code breakpoints, 88-89
Setting and retrieving application settings from UAC administrator listing (15-13), 732
SID structure definition listing (7-1), 319
side effects
access control, Windows Vista, 712-714
file redirection, 601
SIDs (security identifiers), 319-320, 325-328
types, 327
well-known SIDs, 327
Simple application that declares a number of
functions listing (5-16), 244-245
Simple application that performs heap
allocations listing (6-1), 270
Simple application that uses uninitialized
memory listing (6-4), 282-283
Simple application utilizing the
TerminateThread API
listing (10-11), 523
Simple C++ Binary Tree Implementation
listing (11-1), 556, 559
Simple console-based application that
simulates a memory corruption
listing (5-1), 203-207
Simple crashing application
listing (13-1), 633
Simple debugger events processing
listing (3-3), 127
Simple example of double free
listing (6-10), 308-312, 314
Simple function using __try/__except
constructs listing (3-24), 162
simulating ImpersonateSelf invocation,
debugger targets, 340-341
Simulating a kernel32!Sleep call
listing (3-37), 177
Simulating code tracing after attaching to a
running project listing (3-32), 172
Simulating ImpersonateSelf invocation in
the debugger target listing (7-15), 341
Size field (!heap extension command), 478
Size option (Heaps test setting), 750
SizeEnd option (Heaps test setting), 750
SizeStart option (Heaps test setting), 750
.sleepe command, 44
source code
Fibonacci function, 75-76
instrument source code, memory corruption
analysis, 208
source code analysis, memory corruption,
202-208
source files, 64-66, 179
information
gathering, 188-191
using, 192-193
managing for debugging, 188-196
Source of Fibonacci function implemented
in the 02sample.exe sample
listing (2-23), 75
Source server file tree configuration
listing (4-10), 194
source servers
file tree configuration, 194-196
without source revision control, 194-196
sources (security information), 328
access tokens, 328-330
resolving, remote debugging, 117
SDs (security descriptors), 330-333
SourceServer information stored in the PDB
file listing (4-9), 191
SpinCount field (RTL_CRITICAL_
SECTION structure), 499
SRM (Security Reference Monitor), 318
sse (single step exception), 138
ssec (single step exception
continuation), 138
Stack Back Trace section (Dr. Watson dialog
box), 659
stack corruptions, 209-222
asynchronous operations, 231-240
avoidance strategies, 235-255
calling conventions mismatch, 240-255
stack overruns, 223-231
stack pointers, 231-240
stack overruns, 223-231
stack pointers, stack corruptions, 231-240
Stack trace after loading a dynamic link
library listing (3-12), 142
stacks
call stacks
displaying, 212
function calls, 210
manually constructing, 247-252
memory corruption, 625
ProcA procedure, 214
assembly code, 214-216
Standard user mode debugger loop
listing (3-2), 126
starting
  Debug Diagnostic Tool, 692
  processes under debuggers, 281
Starting the client and listing a partial call
stack for each thread listing (8-5), 388
Starting the debugger server
listing (2-45), 111
Stat screen (LeakDiag), 5
state analysis, memory corruption, 201-202
State Dump for Thread ID X section (Dr.
Watson dialog box), 658
state transition, kernel mode prompts, 42
Status field (heap extension command), 478
stdcall calling convention, 240-243
stepping over, function execution, 95-96
stop codes
  DangerousAPIs test setting, 774
  FilePath test setting, 765
  Handles test setting, 748
  Heaps test setting, 756
  Locks test setting, 758-759
  LuaPriv test setting, 772-774
  Memory test setting, 761-762
  ThreadPool test setting, 763
  TLS test setting, 764
Stopping the DcomLaunch code
  after impersonating the client
listing (7-35), 373
storing
  public symbols, HTTP servers, 187-188
  symbols in symbol stores, 184-186
string events, processing, user mode
d debuggers, 127-129
stripped symbols. See symbols
structured exception dispatching methods,
144-153
structures
  exceptions, 145-146
  WOW64 structures, obtaining, 608
  subsegments, 721
Sum function, assembly code, 217-219
summary information, threads, listing, 391
suspending threads, 174
  kernel mode debuggers, 176-177
Switching from user mode to kernel mode
d debugger listing (2-5), 43
  sxd (set exceptions disable), 136
  sxe (set exceptions enable), 136
  sxi (set exceptions ignore), 137
  sxn (set exceptions notify), 137
  symbol caches, 54-55
  symbol files, 49-51, 179
    checking, 57-60
  symbol servers, 52-54
  symbol stores, 184-186
  symbols, 591
d debuggers, 49
  caches, 54-55
  loaded modules, 57-60
  paths, 52, 55-57
  reloading, 60-61
  symbol files, 49-51, 57-60
  symbol servers, 52-54
  utilizing, 62-64
  validating, 61
  managing, 180-188
  private symbols, 591
  public symbols, 591
  generating, 180-183
  sharing on HTTP servers, 187-188
  resolving, remote debugging, 115-116
  symbol stores, storing in, 184-186
symchk.exe, 8
symstore.exe, 8
synchronization, 493
  critical sections
    managing, 545-550
    orphaned critical sections, 516-529
    unlocked, 543-545
deadlocks, 510-519
DllMain function, 529-537
lock contention, 538-545
problem analysis process, 505-509
  threads, 493
  critical sections, 497-502
  event primitive, 494-497
  mutex, 502-504
  problem analysis, 507-509
  semaphore, 504
Windows Vista, 737
  condition variables, 739
  one-time initialization, 740
  reader/writer locks, 737-738
  thread pools, 740
system boundaries, security checks, 338-340
system health, analyzing, Process Explorer, 22-23
System Information section (Dr. Watson dialog box), 657
system targets, controlling, debuggers, 168-177
T
  t command, 94-95
target systems
  debuggers, discovering, 67
  kernel debuggers, connecting to, 39-41
targets
  controlling, debuggers, 168-177
  ImpersonateSelf invocation, simulating, 340-341
  user mode debuggers, creating, 124-125
Task List section (Dr. Watson dialog box), 658
task trees, processes, listing in, 34-35
TCP (Transmission Control Protocol), 112-113
TEB (thread environment block)
  obtaining, 93
  WOW64 applications, dt command, 617
TerminateThread API, 523, 525-528
termination, threads, 523-529
test settings, Application Verifier, 747
  Dangerous APIs, 774-775
  DirtyStacks, 775
  Exceptions, 747
  FilePaths, 764-765
  Handles, 747-748
  Heaps, 749-757
  HighVersionLie, 765-767
  InteractiveServices, 767-768
  KernelModeDriverInstall, 768-769
  Locks, 757-759
  Low Resource Simulation, 769-770
  LuaPriv, 771-774
Memory, 760-762
PrintAPI, 776
PrintDriver, 776
ThreadPool, 762-763
TimeRollOver, 775
TSL, 764
thiscall calling convention, 243
Thread environment block on two different threads in the same process listing (3-28), 166
thread impersonation tokens, displaying, 329
Thread list and associated stacks of hung application listing (10-12), 525
thread local storage (TLS), 764
thread pools, synchronization, 740
thread state, dumping, 173
thread state management, live debugging, 172-176
ThreadPool test setting (Application Verifier), 762-763
ThreadProcedure function, 212-213
threads
  call stacks, displaying, 212
  communication, 387
  current threads, changing, 175
  dumping out, 506-507
  freezing, 174
  information, attaining, 402-403
  kernel threads, dumping, 391
  multithreading, 493, 550
  resuming, 174
  summary information, listing, 391
  suspending, 174
    kernel mode debuggers, 176-177
  synchronization, 493
    critical sections, 497-502
    event primitive, 494-497
    mutex, 502-504
    semaphore, 504
  synchronization problems, analyzing, 507-509
  TEB (thread environment block), obtaining, 93
  termination, 523-529
  unfreezing, 174
time stamps, current time stamps, attaining, 400
TimeOut option (Heaps test setting), 752
TimeRollOver test setting (Application Verifyer), 775
tlist.exe, 8
TLS (thread local storage), 764
!token extension command, failures, 368-371
token propagation, client server applications, 334-338
impersonation levels, 337-338
remote authentication, 335-337
security support provider, 335-337
tools, 3-4
!analyze extension command, 691, 699-708
Application Verifier, 10-16
Debug Diagnostic Tool, 691-693
crash rule, 692
custom analysis script analysis, 697-699
handle leak analysis, 693-697
hang rule, 692
Host component, 692
leak tracker component, 692
leaks rule, 692
memory leak analysis, 693-697
Service component, 692
starting, 692
User Interface component, 692
Debug Diagnostics Tool, 691
DebugDiag, 27
Debugging Tools for Windows, 7-9, 29-30
CDB (cdb.exe) tool, 31
commands, 32-33
KD (kd.exe) tool, 32
NTSD (ntsd.exe) tool, 31
WinDbg (windbg.exe) tool, 31-32
Ethereal, 26
for Windows Vista, 710
Global Flags, 16
Command Line mode, 19-21
GUI mode, 16-18
option abbreviations, 19-20
leak detection tools, 432-433
LeakDiag (Leak Diagnosis) tool, 4-6
Process Explorer, 22-23
Process Monitor, 23
UMDH, 9
WDK (Windows Driver Kits), 23-26
installing, 24
Tools.ini content listing (3-9), 140
Trace and watch function execution listing (2-42), 98
Traces option (Heaps test setting), 751
tracing
code execution, 94-95
function execution, 98
processors, 172
Tracing the remote authentication from the server process listing (7-12), 336
tracing tools, security failures, investigating, 376-378
traffic (network), analyzing, 413-421
troubleshooting
local communication, 382-396
remote communication, 396-422
RPC troubleshooting state information, 396
cell debugging, 396-412
True 32 bit Stack in WOW64 Process (hidden in Listing 121) listing (12-3), 608
TSL test setting (Application Verifyer), 764
Two methods of setting up the symbol path at debugger startup listing (2-13), 55
type information, debugger extensions, initializing, 574-575
Typical client stack waiting on remote call made using a connection-based protocol listing (8-22), 409
Typical stack of a server thread waiting for a new request on DCOM over LRPC listing (8-7), 390
Typical stack of clients using DCOM over LRPC listing (8-6), 389
Typical use of the ba command listing (2-37), 89
U
u command, 72
u command used in user mode debugger (x86) listing (2-22), 72
UAC (User Access Control), Windows Vista, 725-731
uf command, 215
uld (unload a module) event, creating, 143
UMDH leak detection tool, 9, 432
    BSTRs, 469
    memory leaks, identifying, 465-469
UMDH.exe LeakDiag tool, 4, 8
Unalign option (Heaps test setting), 751
Unassembled non-optimized function
    listing (12-9), 613
Unassembly code is dependent on
    the processor execution mode
    listing (12-5), 609
underruns, heaps, 286-300
unfreezing threads, 174
uninitializing debugger extensions, 578
uninitiated state, heap corruptions, 282-286
Unlocked Critical Section
    listing (10-18), 543
unlocked critical sections, 543-545
Use of d command listing (2-34), 85
Use of ds command listing (2-35), 87
Use of dt command listing (2-33), 83
Use of dv command listing (2-32), 82
UseLFHGuardPages option (Heaps test
    setting), 752
User allocation size field (!heap extension
    command), 478
User Interface (Debug Diagnostic Tool), 692
user mode debuggers, configuring, 34-36
User mode debugger output listing (2-6), 47
user mode debuggers, 30-31, 124
    code breakpoints, setting, 79-81
    event processing, 120-130
    events
        controlling, 133-144
        order, 131, 133
exceptions
    controlling, 144-166
    structured exception dispatching
    mechanism, 144-153
kernel debuggers, redirecting through,
    41-44
loops, 125-126
operating systems, support for, 124-130
output, 47
processes, starting, 125
targets, creating, 124-125
u command, 72
version output, 67
Windows Vista, 712-716
user mode extensions, debuggers, 554
Using !getcallinfo to filter call information to
    a specific process listing (8-19), 405
Using !getcallinfo to obtain the call
    information maintained by the server
    listing (8-18), 405
Using !getclientcallinfo to obtain the call
    information maintained by the client
    listing (8-21), 408
Using !getdbgcell to obtain the cell
    information maintained by the server
    listing (8-20), 406
Using !getendpointinfo to list all endpoint
    known by RPC listing (8-15), 402
Using !getendpointinfo to list all endpoints
    known by RPC listing (8-14), 401
Using !getthreadinfo to list all thread from
    RPC thread pool listing (8-16), 402
Using !getthreadinfo to obtain a
    specific thread RPC information
    listing (8-17), 403
Using !lpc extension to get message
    information listing (8-2), 385
Using !lpc extension to get port information
    listing (8-3), 386
Using !lpc extension to obtain the
    entire LPC activity on the system
    listing (8-4), 386
Using !rpctime to obtain the current time
    stamp used by troubleshooting
    infrastructure listing (8-13), 400
Using breakpoints in the kernel mode
    debugger listing (2-30), 80
Using breakpoints in the user mode
    debugger listing (2-29), 79
Using breakpoints in the user mode
    debugger listing (2-31), 80
Using kls flag for detecting a user mode
    module mapping listing (3-29), 167
Using the !cs command to list all critical
    sections of a process listing (10-3), 501
Using the \!handle extension command to find mutex object in notepad.exe listing (10-4), 502
Using the \!token extension command to display a token in the user mode debugger listing (7-5), 326
Using the *kb command to Dump all threads and associated stacks listing (10-5), 506
Using the .sympath and .symfix commands listing (2-14), 56
Using the CritSec object listing (14-2), 698
Using the link.exe utility to find debug information stored in the binary file listing (2-8), 51

V
validating symbols, 61
values
   entering, 103-104
   local variables, discovering, 613-614
variable values, debuggers, displaying, 81-84
variables
   condition variables, synchronization, 739
   environment variables
      BUFFER_OVERFLOW_CHECKS environment variable, 210
      Windows x64, 601
   gGlobal variable, declaring, 88-89
   local variables, discovering, 613-614
version information, debugger extensions, initializing, 572
version output, user mode debuggers, 67
Version output from a user mode debugger listing (2-18), 67
versioning debugger extensions, 592
Virtual Memory Manager (VMM). See VMM (Virtual Memory Manager)
Virtual PC, kernel debugger, enabling for, 40
VMM (Virtual Memory Manager), 461

W
Walking the stack back in time listing (5-14), 236-239
Watching the error code returned by OpenThreadToken/OpenProcessToken listing (7-30), 369
WDbgExts APIs, 564-565
WDbgExts extension, 563-565
   APIs, 564
      code organization, 567-570
      header files, 567-570
      initialization, 570-575
         COM interfaces, 572-573
         extension data, 576
         type information, 574-575
         version information, 572
      requirements, 565-570
WDK (Windows Driver Kits), 23-26
   installing, 24
   well-known SIDs, 327
WinDbg (windbg.exe) tool (Debugging Tools for Windows), 31-32
Windbg GUI
   event breaks, adjusting, 138
   event handling, adjusting, 138
windbg.exe, 8
Windows Advanced Option menu, 38
Windows Driver Kits (WDK). See WDK (Windows Driver Kits)
Windows Error Reporting, 633
   architecture, 662-682
   binaries, mapping to products, 673-677
   Dr. Watson dialog box, 653-654
      Application Errors section, 656-657
      Application option, 656-657
      Crash Dump section, 655
      Crash Dump Type section, 655
      Log File Path section, 654
      Module List section, 658
      Number of Errors to Save section, 655
Number of Instructions section, 655
Raw Stack Dump section, 660-662
Stack Back Trace section, 659
State Dump for Thread ID X section, 658
System Information section, 657
Task List section, 658
postmortem debugging, 653-662
product rollup page, 672
software options, 672
web site, navigating, 671
Windows heap manager, 261
Windows Vista, 709
custom debugger extensions, writing, 741
d debuggers, 711
access control side effects, 712-714
address space layout, 716-717
kernel mode debuggers, 715-716
user mode debuggers, 712-714
debugging, postmortem debugging, 741-745
debugging security, 723-725
debugging security, 723-725
file virtualization, 732-735
registry virtualization, 732-735
UAC (User Access Control), 725-731
dump files, generating, 743
Event Log system, 710-711
heap manager, 717-723
integrity levels, 724
interprocess communication, 736
resource leaks, 736
synchronization, 737
condition variables, 739
one-time initialization, 740
reader/writer locks, 737-738
thread pools, 740
tools, 710
Windows x64
Application Verifier, 604
changes to, 602-605
Debugging Tools for Windows, 603-604
environment variables, 601
Ethereal, 605
exception models, 622-625
Leak Diagnostic tool, 603
working set sizes, adjustments, 464
WOW64
32-bit applications, running in, 598-602
applications, 615-617
commands, 607-609
structures, obtaining, 608
WOW64 application's PEB using the dt
command listing (12-12), 617
WOW64 applications PEB listing (12-11),
615-617
WOW64 threads’ TEB using !teb extension
class command and dt commands listing
(12-13), 618
writing custom 64-bit debugger
extensions, 629
wt command, 98
X–Z
x86 context flags values listing (3-18), 146
x86-64-bit general purposes register
listing (12-2), 606