NUMERICS
0day, definition, 4
0day kernel vulnerabilities, Unix, 636–642

A
AAAS (ASCII Armored Address Space), 394–395
abusing frame-based exception handlers, 161–166
eexisting handlers, 162–164
alphanumeric filters, exploits, writing, 205–209
application layer attacks, database software, 618–619
arbitrary free vulnerabilities, 271
arbitrary size overflow, stack overflows, 232–233
architectural failures
asymmetry and, 511–512
authentication and, 512
authorization and, 512
boundaries and, 508–509
data translation and, 509–511
archives, paper, 438
arrays, 12
ASCII, converting to Unicode, 210–211
ASCII Venetian implementation, 214–217
ASLR (Address Space Layout Randomization), 313, 396–399
asm( ) statements, 132
assembler references, 430
assembly language, 6
C code constructs, 7–10
C++ code constructs, 7–10
registers and, 6
asymmetry, problems, 511–512
attack detection, bypassing
alternate encodings, 514
attack signatures, 517
file-handling features, 515–517
length limitations, 517–520
stripping bad data, 513–514
auditing binaries. See binary auditing
auditing source code
automated analysis tools, 484
auditing source code (continued)
methodology
  bottom-up approach, 485
  selective approach, 485–486
  top-down approach, 485
tools
  Cbrowser, 484
  Cscope, 482–483
  Ctags, 483
  editors, 483
vulnerabilities
  different-sized integer conversions, 497–498
double free, 498
  extinct bug classes, 487
  format strings, 487–489
generic logic errors, 486
  incorrect bounds-checking, 489–490
  integer-related vulnerabilities, 495–497
  loop constructs, 490
  multithreading and, 500–501
  non-null termination issues, 492
  off-by-one, 490–492
  out-of-scope memory usage, 499
  signed comparison vulnerabilities, 494–495
  skipping null-termination issues, 493
  uninitialized variable usage, 499–500
  use after free vulnerabilities, 499–500
authentication, problems, 512
Authentication Tokens, 116
authorization, problems, 512
automated source code analysis tools, 484

B
binary auditing
  C++ code constructs, 561
calling conventions, 554–555
  C calling convention, 555
  Stdcall calling convention, 555
  compiler-generated code for loops, 557–558
  function layouts, 556
  if statements, 556–557
  switch statements, 558–560
  while loops, 557–558
IDA Pro, 550–552
manual, 563–566
memcpy-like code constructs, 560
source code auditing, 550
stack frames, 552–553
  BP-based, 553, 554
  frame pointer, 553–554
strlen-like code constructs, 560–561
this pointer, 561–562
bit flipping, 469–470
boundaries, problems, 518–509
bounds-checking, 489–490
bridge building, 206
BSD, OS X and, 314
buffer overflows
  exploiting, 197–202
  heap-based, 173
  exploiting, 178–194
  stack-based, 156
buffers, 12–13
  arrays, 12
  length, 95
overflowing on stack, 18–23
bug discovery, exploitation and, 6

C
C, code constructs, 7–10
C++, code constructs, 7–10
canaries, 166, 388–389
Cbrowser, 484
chained ret2code, 379
Check Heaps, 359
Cisco IOS, 339–340
crash dumps, 354–355
exploiting
heap overflows, 359–361
stack overflows, 357–359
GDB agent, 355–356
hardware platforms, 340
images
diffing, 350–351
taking apart, 349–350
partial attacks
global variable overwrite, 363–364
NVRAM invalidation, 362
reverse engineering, 348–356
runtime analysis, 351–356
ROMMON, 351–354
shellcode, bind shell, 370–372
shellcodes
configuration changing, 364–370
runtime image patching, 370
software packages, 340–343
system architecture
IO memory, 346
IOS heap, 344–346
memory layout, 343–344
vulnerabilities, 346–347
command-line interface, 348
protocol parsing code, 347
security, 347–348
services, router, 347
class definitions, reconstructing
vtables, 562–563
configuration related shellcode, 599
continuation of execution, 441–442
control registers, 7
cookies, 403
countermeasures, 601–602
brute forcing, 602–603
information leaks, 605–606
local exploits, 603
OS/application fingerprinting,
603–605
preparation, 602
cross-platform shellcode (OS X),
332–333
Cscope, 482–483
Ctags, 483

data, instructions and, 4
data translation, problems, 509–511
database software
application layer attacks, 618–619
network layer attacks, 608–618
operating system commands
IBM DB2, 621–623
Oracle, 620–621
SQL Server, 619–620
DB (define byte) directive, 56
DCE-RPC, 116
DCE-RPC tools, 118
recon, 118–120
SPIKE, 118–120
DCOM (Distributed Component
Object), 111
DCE-RPC and, 116–123
exploitation, 120
debugging
debug trick, 433–434
OllyDbg, 112–114
unhandled exception filter and, 186
debugging (continued)

Windows
- kernel debugger, 124
- Microsoft tool chain, 124
- OllyDbg, 124
- SetDefaultExceptionHandler, 126
- shellcode, writing, 125
- SoftICE, 124
- TlsSetValue(), 126
- VirtualProtect(), 126
- Win XP, 127
- Win2K, 126
- Win32, 124–125
- Win9X/ME, 126
- WinDbg, 124
- Windows 2003 Server, 127
- Windows Vista, 127
- WinNT, 126
- WSASocket(), 126

decoder, Unicode, 218–221
delay slot, SPARC, 227
device drivers, 683
  - I/O control code components, 693–694
  - IOCTL handlers, 694–695
directives, DB (define byte), 56
dlmalloc, 89
DOS attacks, 521
double free vulnerabilities, 270, 498
dynamic analysis, fuzzing and, 470
dynamic heaps, 173
dynamic linking
  - single stepping dynamic linker, 281–296
  - SPARC ABI, 279
dynamic string table, 280

e
EAX register, 177
EBP register, 15
editors, source code, 483
EFLAGS (Extended Flags), 7
EIP (Extended Instruction Pointer), 7
controlling, 22–23
ellipsis syntax, 84
encryption, end-to-end, 299
end-to-end encryption, 299
exception filter, unhandled,
  - overwrite pointer to, 185–191
exception handlers
  - abusing existing, 162–164
  - frame-based, 156–161
  - triggering, continuation and, 441
exception handling
  - signal() system call, 122
  - vectored, 181
  - Vectored Exception Handling, 123
  - Win32 and, 122–123
  - Windows, searches and, 148–153
  - Exception Registration Record, 392
  - EXCEPTION_REGISTRATION
    - structure, 156
execution
  - continuation of, 441–442
  - controlling for exploitation, 75–84
execve function, 54
execve syscall, 52
exit() syscall, shellcode for, 44–48
exploitation
  - Cisco IOS
    - heap overflows, 359–361
    - stack overflows, 357–359
  - execution, controlling, 75–84
  - heaps, OS X, 333–335
  - overruns, SQL level, SQL
    - functions, 623–625
  - Solaris/SPARC, methodology,
    - 263–270
  - stack overflows, 236–241
exploits
  - countermeasures, 601–606
  - definition, 4
  - one-factor, 598
  - planning, 439
exploits (continued)
root privileges and, 25–35
stabilization and, 442–443
writing
   alphanumeric filters, 205–209
   Unicode filters, 209–211
Extended Flags (EFLAGS), 7
extended stack pointer (ESP) register, 6
extinct bug classes, 487
extproc overflow, 504–508

F
fault injection, 445–446
design, 447–456
fault delivery, 455
fuzzing and, 461
heuristics, 455–456
input generation, 447–448
   automated, 449
   fuzz generation, 449–450
   live capture, 449
   manual, 448
modification engines, 450–451
   delimiting logic, 451–453
   input sanitization, 453–454
Nagel algorithm, 455
state-based protocols, 456
stateless protocols, 456
timing, 455–456
fault monitoring, 456–457
debuggers and, 457
FaultMon, 457–458
fault testing, 446
FIFO (first in first out), 5
filters
   alphanumeric, exploits, writing, 205–209
   Unicode, exploits, writing, 209–211
   Windows, 129–130
foo( ) function, strcpy( ) call, 176
format strings, 61–63, 487–489
   bugs, 63–67
   reasons for, 84–85
exploits, 68–69
   crashing services, 69–70
   information leakage, 70–75
techniques, 85–88
frame pointers, 15
frame-based exception handlers, 156–161
   abusing, 161–166
   existing, 162–164
fstat (BSD), 435
functions
   execve, 54
   HeapAllocate( ), 173
   KiUserExceptionDispatcher, 161
   MyExceptionHandler, 158
   printf, 63–64
   RtlImageNtHeader, 161
   stack and, 15–18
fuzzers, 433
   bit flipping, 469–470
   Blackhat, 480
   CHAM, 480
   definition, 4
   Hailstorm, 480
   open source, modifying, 470
   weaknesses in, 468
fuzzing, 99
dynamic analysis and, 470
fault injection and, 461
introduction, 461–465
scalability, 466–467
sharefuzz, 462
static analysis versus, 466

G
gcc (GNU Compiler Collection), 430
gdb (GNU Debugger), 430–431
genral-purpose registers, 6
generic logic errors, 486
GetSystemTimeAsFileTime, 167
GOT (Global Offset Table), 279
GPG 1.2.2 randomness patch, 583
GetDefaultHeap(), 115
HeapAllocate(), 115
HeapCreate(), 114
initialization, 5
malloc(), 91
overview, 91
process heap, 173–177
reallocate(), 91
repairing, 191–193
RtlHeapAllocate(), 115
RtlHeapFree(), 115
strcpy(), 176
threading and, 115–116
host IDS related shellcode, 599

H
hardware, Cisco IOS, 340
heap overflows, 91
advanced exploitation, 105–107
breakpoints, 96
buffer length, 95
Cisco IOS, 359–361
partial attacks, 362–364
dlmalloc, 96
every example, 271–276
intermediate, 98–105
limitations, Solaris, 266–267
ltrace output, 93
malloc, 95
Microsoft IIS, 92
repairing, 179
samba, 92
Solaris, 92
SPARC, 296–299
heap protections, 399–407
heap-based buffer overflows, 173
COM objects and, 193–194
exploiting, 178–194
logic program control data, 194
Solaris/SPARC, 241–263
HeapAlloc, 177
HeapAllocate() function, 173
HeapFree, 177
HeapRealloc, 177
heaps, 114–115
default, 115
dynamic, 173
exploitation, OS X, 333–335
GetDefaultHeap(), 115

I
IBM DB2, operating system
commands, 619–620
IDA Pro, 550–552
ideal stack layout, 389–394
impersonation, tokens and, 120–122
incorrect bounds-checking, 489–490
input validation, bypassing
alternate encodings, 514
attack signatures, evading, 517
file-handling features, 515–517
length limitations, 517–520
stripping bad data, 513–514
Instruction Pointer, 7
instructions, data and, 4
integer-related vulnerabilities, 495–497
Intel shellcode (OS X), 324–327
ret2libc, 327–329
ret2strncpy, 329–321
IPS (Intrusion Prevention System), 279
K
kernel, Unix
  exec_ibcs2_coff_prep_zmagic()
  vulnerability, 647–652
  breakpoints, calculating, 652–654
  execution, redirecting, 654–655
  kernel mode payload, 658–665
  offsets, calculating, 652–654
  process descriptor, 655–658
  return address, overwriting, 654–655
  root, 665–672
overflows
  0day kernel vulnerabilities, 636–642
  vulnerability types, 627–636
  Solaris vfs_getvfssw(), 642–344, 672–678
kernel, Windows
  introduction, 682–683
  kernel-mode payloads, 695
  rootkit, 703
  security, subverting, 701–703
  user-mode payload, 699–701
  user-mode processes, 696–698
mode flaws, 681–682
programming flaws, 683–684
  heap overflows, 688
  repurposing attacks, 689
  shared object attacks, 689
  stack overflows, 684–687
  user-mode addresses, insufficient validation, 688–689
  KiUserExceptionDispatcher function, 161

L
lazy binding, 280
LIFO (last in first out), 5
linking
  dynamic, 279
  unsafe unlinking, 402

Linux
  protections
    ASLR, 418–419
    heap, 420
    stack data, 419–420
    W^X, 417–418
    Windows comparison, 111–114
  loop constructs, 490
  ltrace (Unix), 434

M
malloc, 89
memory, Cisco IOS, 343–344
memory management, modem, 4
Microsoft toolchain, debugging and, 124
modeling network protocols, 469
SPIKE, 472–480
modem, memory management, 4
mount() system call, 645–646
MyExceptionHandler function, 158
MySQL, 1-bit patch, 578–580

N
NASM (Netwide Assembler), 431
NetCat, 434
Network IDS (Intrusion Detection System), 279
network layer attacks, database software, 608–618
network related shellcode, 599
networks, protocols, modeling, 469
NEXTSTEP, OS X and, 314
NGS (Next Generation Security Software), 504
non-executable stacks
  exploiting, 197–202
  Return to libc, 35–38
  non-null termination issues, 492
NOP method
  offsets and, 33
SPARC, 231
null-termination issues, 493
nulls, 48–49
NVRAM validation, 362

O
objdump utility, 45
off-by-one vulnerabilities, 490–492
offset finder, 432–433
offsets, NOP Method and, 33
OllyDbg, 112–114, 124, 431
one-factor exploits, 598
opcodes, alphanumeric bytes, 206
open source
  modifying, fuzzers and, 470
OS X, 314–315
OpenBSD, protections
  ASLR, 421
heap, 423
stack data, 423
W^X, 422
OpenSSH RSA authentication patch, 580–581
operating system commands, database software
IBM DB2, 621–623
Oracle, 620–621
SQL Server, 619–620
Oracle
  extproc overflow, 504–508
operating system commands, 619–620
OS X, 313
  Aqua, 314
BSD and, 314
bugs, 335–337
cross-platform shellcode, 332–333
exploits, resources, 337
heap exploitation, 333–335
Intel shellcode, 324–327
  ret2libc, 327–329
  ret2str(l)cpy, 329–321
NEXTSTEP and, 314
open source, 314–315
passwords and, 316
PowerPC shellcode, 316–324
protections
  ASLR, 423
heap, 423
stack data, 423
W^X, 422
Unix and, 315–316
out-of-scope memory usage
  vulnerabilities, 499
overflows
buffer
  heap-based, 173
  stack-based, 156
.data section, 194–196
heap-based, Solaris/SPARC, 241–263
off-by-one, 270
stack-based buffer overflow
  methodologies, 232–236
static data overflows, 276
TEB/PEB, 196–197

P
Packetstorm, 599
paper archives, 438
passwords, OS X and, 316
PaX, 382
PE-COFF (Portable Executable-
  Common File Format), 111
Win32 and, 112–114
PIC (Position Independent Code), 133
planning exploits, 439
PLT (Procedure Linkage Table), 280
pointers, frame pointers, 15
Index

popping shells, Windows, 153–154
PowerPC shellcode (OS X), 316–324
printf functions, 63–64
privilege related shellcode, 599
process heap, 173–177
Proglot server, 584
ProPolice, 390
protections, 375–376
AAAS (ASCII Armored Address Space), 394–395
ASLR (Address Space Layout Randomization), 396–399
heap protections, 399–407
implementations
Linux, 417–420
OpenBSD, 421–422
OS X, 422–423
Solaris, 423–424
Windows, 413–417
kernel protections, 411–412
non-executable stack, 376–381
pointer protections, 412–413
stack data
canaries, 388–389
ideal stack layout, 389–394
W^X memory, 381–387
Windows SEH protections, 407–409
EEREAP, 409–410
pdest, 410–411
SEHInspector, 411
protocols, network, modeling, 469
Python, 432

ratifying, heaps, 191–193
ret2code, 378
ret2data, 377
ret2dl-resolve, 380–381
ret2gets, 378
ret2libc, 377
ret2plt, 380
ret2strcpy, 377–378
ret2syscall, 379
ret2text, 380
reverse engineering Cisco IOS, 348–356
Roman Exploit Writer, 214–217
ROMMON, Cisco IOS, 351–354
root exploits and, 25–35
shell, spawning, 26
RtlAcquirePebLock(), 178
RtlEnterCriticalSection(), 178
RtlImageNtheader function, 161
RtlLeaveCriticalSection(), 178
RtlReleasePebLock(), 178
runtime analysis, Cisco IOS, 351–356
runtime patching, 581–583

S
scalability, fuzzing, 466–467
sea monkey data, 518
searches, Windows exception handling and, 148–153
Security Cookies, 166
generating, 167
.set statements, 132–133
SetUnhandledExceptionFilter, 185
sharefuzz, 462
shellcode
Cisco IOS
bind shell, 370–372
configuration changing, 364–370
runtime image patching, 370
configuration related, 600

R
register windows, stack overflows and, 233
registers, 6–7
control, 7
EBP, 15
ESP (extended stack pointer), 6
SPARC, 224–227
shellcode (continued)
exit( ) syscall, 44–48
Host IDS related, 600
injectable, 48–50
introduction, 41–42
libraries, 441
network related, 599
OS X, PowerPC, 316–324
Packetstorm, 599
privilege related, 599–600
Solaris, advanced, 299–311
SPARC
advanced, 299–311
exec, 229–230
self-location determination, 228–229
thread related, 600–601
Unix, pitfalls, 299
Windows
heapoverflow.c, 132–148
PEG, 132
setup, 131–132
Windows debugging, 125
writing in inline assembler, 439–441
shells
spawning, 50–59
Windows, popping, 153–154
signal() system call, 122
signed comparison vulnerabilities, 494–495
single stepping dynamic linker, 281–296
Snort IDS, 299
SoftICE, 124
software forced crash, 359
software packages, Cisco IOS, 340–343
Solaris
exploit methodology, 263–270
heap overflows
limitations, 266–267
methodology, 263
overflows, heap-based, 241–263
overwrite targets, 267–270
protections, 423
ASLR, 424
heap, 424
stack data, 424
W^X, 424
self-location determination, 228–229
shellcode, advanced, 299–311
stack frame, 231–232
static data overflows, 276
system calls, 230
vfs_getvfilsw( ), 642–344
source code
auditing
binary auditing, 550
Cbrowser, 484
Cscope, 482–483
Ctags, 483
editors, 483
methodology, 485–486
vulnerabilities, 486–501
automated analysis tools, 484
SPARC (Scalable Processor Architecture), 224
ABI (Application Binary Interface), 279
delay slot, 227
exploit methodology, 263–270
NOP instruction, 231
overflows, heap-based, 241–263
padding instructions, 231
register windows, 224–227
registers, 224–227
shellcode
advanced, 299–311
exec, 229–230
self-location determination, 228–229
stack frame, 231–232
stack overflow exploitation, 236–241
SPARC (continued)
  stack-based overflow
    methodologies, 232–236
  synthetic instructions, 228
  UltraSPARC processors, 224
  spawning shells, 50–59
  SPIKE, 471–472
    modeling network protocols,
    472–480
SQL (Structured Query Language), 504
  exploiting overruns, 623–625
  Transact-SQL, 504
SQL Server
  3-byte patch, 575–578
  operating system commands,
  619–620
SQL-UDP, 522
stack data
  canaries, 388–389
  ideal stack layout, 389–394
stack frame, Solaris/SPARC,
  231–232
stack overflows
  arbitrary size, 232–233
  Cisco IOS, 357–359
  exploitation, 236–241
  register windows and, 233
stack protection, Windows 2003
  Server and, 166–172
stack-based buffer overflows, 156
  methodologies, 232–236
StackGuard, 166
stacks, 13–15
  boundary, 13–14
  buffers, overflowing, 18–23
  EBP register, 15
  functions and, 15–18
  initialization, 5
  non-executable
    exploiting, 197–202
    Return to libc, 35–38
POP, 14
PUSH, 14
static analysis versus fuzzing, 466
static data overflows, 276
strace (Unix), 434
strcpy(), 176
symbol resolution, 280
syntax
  Windows, 129–130
  x86 AT&T versus Intel, 130
synthetic instructions, SPARC, 228
syscall proxies, 584–586
  problems with, 587–596
sysfs() system call, 644–645
Sysinternals Process Explorer,
  tokens and, 121
system calls, 42–44
  arguments, 43
  execve, 52
  exit(), 43
    shellcode, 44–48
  Solaris, 230
  Windows, attacking, 692

tTcpdump (Unix), 435
TEB (Thread Environment Block), 190
TEB/PEB overflows, 196–197
termination
  non-null, 492
  skipping null-termination issues, 493
thread related shellcode, 599
threading, heaps, 115–116
Threading and Process Architecture, 116
TNS (Transparent Network
  Substrate), 504
tokens
  impersonation and, 120–122
  Sysinternals Process Explorer, 121
tracing for vulnerabilities
component design
data collection, 537–538
function hooking, 534–537
machine-code analysis, 531–534
process injection, 530
fingerprint systems, 546–547
format bugs, 547
integer overflows, 547
VulnTrace, 538–546
Transact-SQL, 504
truss (Unix), 434
shellcode, pitfalls, 299
Solaris, vfs_getvfssw( ), 642–344
strace, 434
tcpdump, 435
truss, 434
Wireshark, 435
unlinking, unsafe, 402
unsafe unlinking, 402
use after free vulnerabilities,
499–500
utilities, objdump, 45

U
UltraSPARC processors, 224
unhandled exception filter,
overwrite pointer to, 185–191
Unhandled Exception Filter
mechanism, 185
UnhandledExceptionFilter function,
168
Unicode, 210
converting from ASCII, 210–211
decoder, 218–221
exploits, instruction set, 212–213
Unicode filters, exploits, writing,
209–211
Unicode-based vulnerabilities,
exploiting, 211–213
uninitialized variable usage, 499–500
Unix
fstat (BSD), 435
kernel,
exec_ibcs2_coff_prep_zmagic( )
vulnerability, 647–672
kernel overflows
0day kernel vulnerabilities,
636–642
vulnerability types, 627–636
ltrace, 434
OS X and, 315–316

V
variables
local, placement, 168
uninitialized variable usage,
499–500
Vectored Exception Handling, 123,
181
Venetion technique, 213–217
ASCII implementation, 214–217
versioning, 598–599
Visual C++, 431
vtables, 562–563
vulnerabilities
arbitrary free, 271
binary
IIS WebDAV, 568–570
LSD’s RPC-DCOM, 567–568
Microsoft SQL Server bugs,
566–567
Cisco IOS, 346–347
command-line interface, 348
protocol parsing code, 347
security, 347–348
services, router, 347
definition, 4
different-sized integer conversions,
497–498
double free, 270, 498
extinct bug classes, 487
vulnerabilities (continued)
fingerprint systems, 546–547
format strings, 487–489
generic logic errors, 486
heap-related, 270–271
incorrect bounds-checking, 489–490
integer-related vulnerabilities, 495–497
loop constructs, 490
multithreading and, 500–501
non-null termination issues, 492
off-by-one, 490–492
out-of-scope memory usage, 499
signed comparison vulnerabilities, 494–495
skipping null-termination issues, 493
tracing for, 525–529
component design, 529–543
format bugs, 547
integer overflows, 547
VulnTrace, 538–546
Unicode-based, exploiting, 211–213
uninitialized variable usage, 499–500
use after free vulnerabilities, 499–500
vulnerable arguments, 391

W
W^X memory, 381–387
wide characters (Unicode), 210
Win32
exception handling, 122–123
PE-COFF and, 112–114
WinDbg, 124, 431
Windows
debugging
kernel debugger, 124
Microsoft tool chain, 124
OllyDbg, 124
SetDefaultExceptionHandler, 126
shellcode, writing, 125
SoftICE, 124
TlsSetValue(), 126
VirtualProtect(), 126
Win XP, 127
Win2K and, 126
Win32, 124–125
Win9X and, 126
WinDbg, 124
Windows 2003 Server, 127
Windows Vista, 127
WinME and, 126
WinNT and, 126
WSASocket(), 126
device drivers
I/O control code components, 693–694
IOCTL handlers, 694–695
exception handling, searches and, 148–153
FileMon, 435
filters, 129–130
HandleEx, 435
IDA pro disassembler, 436
kernel
introduction, 682–683
kernel-mode payloads, 695–703
mode flaws, 681–682
programming flaws, 683–689
Linux comparison, 111–114
ProcessExplorer, 435
protections
ASLR, 414–415
heap, 416
SEH, 416
stack data, 415–416
W^X, 413–414
RegMon, 435
Windows (continued)
shellcode
  heapoverflow.c, 132–148
  PEB, 132
  setup, 131–132
shells, popping, 153–154
SNMP DOS, 520–521
syntax, 129–130
system calls, 690–691
  attacking, 692
TCPView, 435

Windows 2003 Server, stack protection and, 166–172
Wireshark (Unix), 435

X
x86 AT&T syntax versus Intel syntax, 130