

# INDEX

## Symbols & Numbers

& (ampersand)  
  for address-of operator, 45  
  for background process, 347  
< > (angle brackets), for include file, 91  
= (assignment operator), 12  
\* (asterisk), for pointers, 43  
\ (backslash), for escaped character, 180  
{ } (curly braces), for set of instructions, 8, 9  
\$ (dollar sign qualifier), and direct parameter access, 180  
== (equal to operator), 14  
! (exclamation point), 14  
> (greater than operator), 14  
>= (greater than or equal to operator), 14  
< (less than operator), 14  
<= (less than or equal to operator), 14  
!= (not equal to operator), 14  
! (not operator), 14  
% (percent sign), for format parameter, 48  
" (quotation marks), for include files, 91  
; (semicolon), for instruction end, 8  
\$1 variable, 31  
8-by-8 S-box, 435  
32-bit addressing scheme, 22  
64-bit addressing scheme, 22  
404 HTTP response, 213

## A

accept() function, 199, 206  
access mode for file, 84

Accumulator (EAX) register, 24, 346  
  zeroing, 368  
ACK flag, 223  
  filter for, 260  
active sniffing, 239–251  
add instruction, 293  
Address Resolution Protocol (ARP),  
  219, 240  
  cache poisoning, 240  
  redirection, 240  
  reply messages, 219  
  spoofing, 243  
  request messages, 219  
address-of operator, 45, 47, 98  
addressof.c program, 46  
addressof2.c program, 47  
addr\_struct.c file, 348–349  
administrator account, 88. *See also*  
  root, user  
AES (Rijndael), 398  
AF\_INET, socket address structure  
  for, 201–202  
aircrack, 448–449  
AirSnort, 439  
algorithm, efficiency of, 398  
algorithmic run time, 397–398  
ampersand (&)  
  for address-of operator, 45  
  for background process, 347  
amplification attacks, 257  
AND bitwise operation, 366  
and instruction, 293  
AND operator, 14–15  
< > (angle brackets), for include file, 91  
application layer (OSI), 196  
argument vector, 59  
arithmetic operators, 12–14

ARP. *See* Address Resolution Protocol (ARP)  
 arp\_cmdline() function, 246  
 ARPhdr structure, 245–246  
 arp\_initdata() function, 246  
 arp\_send() function, 249  
 arspoof.c program, 249–250, 408  
 arp\_validatedata() function, 246  
 arp\_verbose() function, 246  
 arrays in C, 38  
 artistic expression, programming as, 2  
 ASCII, 33–34  
     function for converting to  
         integer, 59  
         for IP address, conversion, 203  
 ASLR, 379–380, 385, 388  
 aslr\_demo.c program, 380  
 aslr\_execl.c program, 389  
 aslr\_execl\_exploit.c program,  
     390–391  
 assembler, 7  
 assembly language, 7, 22, 25–37  
     GDB examine command to display  
         instructions, 30  
     if-then-else structure in, 32  
     Linux system calls in, 284–286  
     for shellcode, 282–286  
     syntax, 22  
 assignment operator (=), 12  
 asterisk (\*), for pointers, 43  
 asymmetric encryption, 400–405  
 asymptotic notation, 398  
 AT&T syntax for assembly  
     language, 22  
 atoi() function, 59  
 auth\_overflow.c program, 122–125  
 auth\_overflow2.c program, 126–133

**B**

backslash (\), for escaped  
     character, 180  
 backtrace  
     of nested function calls, 66  
     of stack, 40, 61, 274  
 bandwidth, ping flood to  
     consume, 257  
 Base (EBX) register, 24, 344–345  
     saving current values, 342  
 Base Pointer (EBP) register, 24, 31,  
     70, 73, 344–345  
     saving current values, 342  
 BASH shell, 133–150, 332  
     command substitution, 254  
     investigations with, 380–384  
     for loops, 141–142  
     script to send ARP replies, 243–244  
 BB84, 396  
 bc calculator program, 30  
 beauty, in mathematics, 3  
 Bennett, Charles, 396  
 Berkeley Packet Filter (BPF), 259  
 big-endian byte order, 202  
 big-oh notation, 398  
 bind call, host\_addr structure for, 205  
 bind() function, 199  
 bind\_port.c program, 303–304  
 bind\_port.s program, 306–307  
 bind\_shell.s program, 312–314  
 bind\_shell1.s program, 308  
 /bin/sh, 359  
     system call to execute, 295  
 birthday paradox, 437  
 bitwise operations, 84  
 bitwise.c program, 84–85  
 block cipher, 398  
 Blowfish, 398  
 Bluesmack, 256  
 Bluetooth protocol, 256  
 bootable LiveCD. *See* LiveCD  
 botnet, 258  
 bots, 258  
 BPF (Berkeley Packet Filter), 259  
 Brassard, Gilles, 396  
 breakpoint, 24, 27, 39, 342, 343  
 broadcast address, for amplification  
     attacks, 257  
 brute-force attacks, 436–437  
     exhaustive, 422–423  
 bss segment, 69, 77  
     for C variable storage, 75  
 bt command, 40  
 buffer overflows, 119–133, 251  
     command substitution and Perl to  
         generate, 134–135  
     in memory segments, 150–167  
     notesearch.c program vulner-  
         ability to, 137–142  
     stack-based vulnerabilities, 122–133

- buffer overrun, 119
- buffers, 38
  - program restrictions on, 363–376
- buildarp() function, 246
- byte, 21
- byte counter, incrementing, 177
- byte order of architecture, 30
  - conversion, 238

## C

- C compilers, 19
  - free, 20
  - variable data types and, 58
- C programming language
  - address-of operator, 45
  - arithmetic operators shorthand, 13
  - vs. assembly language, 282
  - Boolean operations, 15
  - comments, 19
  - control structures, 309–314
  - file access in, 81–86
  - functions in, 16
  - memory segments, 75–77
  - programmer responsibility for data integrity, 119
- call instruction, 287
  - null bytes from, 290
- callback function, 235
- carriage return, for line termination in HTTP, 209
- caught\_packet() function, 236, 237
- CD with book. *See* LiveCD
- cdq instruction, 302
- char data type, 12, 43
- character array (C), 38
- char\_array executable binary, 38
- char\_array.c program, 38
- check\_authentication() function, 122, 125
  - stack frame for, 128–129
- child process, spawning root shell with, 346
- chmod command, 88
- chown command, 90
- chsh command, 89
- cleanup() function, 184
- client\_addr\_ptr, 348, 349
  - and crash, 353

- close() function, file descriptor for, 82
- closed ports, response with SYN/ACK packets, 268
- cmp operation, 26, 32, 310, 311
- code segment, 69
- CodeRed worm, 117, 319
- command line, Perl to execute instructions, 133
- command prompt, indicator of background jobs, 332
- command-line arguments, 58–61
- commandline.c program, 58–59
- commands
  - running single as root user, 88
  - substitution and Perl to generate buffer overflows, 134–135
- comments, in C program, 19
- comparison operators, 14–15
- compiled code, 20
- compiler, 7
- computational power, vs. storage space, 424
- computational security, 396
- conditional probability, 114
- conditional statements, variables in, 14
- confusion, 399
- connect() function, 199, 213, 314
- connect-back shellcode, 314–318
- connectback-shell.s program, 314–315
- connectivity, ICMP to test for, 221
- constants, 12
- constructors (.ctors), table sections for, 184–188
- convert.c program, 59–60
- Copyright Act, 118
- core dump, 289
- Counter (ECX) register, 24
- countermeasures
  - for attack detections, 320
  - buffer restrictions, 363–376
  - hardening, 376
  - log files and, 334–336
  - nonexecutable stack, 376–379
  - overlooking obvious, 336–347
  - system daemons, 321–328
  - tools, 328–333
- crackers, 3

- crash, 61, 128
  - from buffer overflow, 120
  - and `client_addr_ptr`, 353
  - by DoS attacks, 251
  - from out-of-bound memory
    - addresses, 60
- CRC32 (cyclic redundancy checksum)
  - function, 434
- criminal activity, 451–452
- `crypt()` function, 153, 418
  - salt values, 423
- cryptanalysis, 393
- `crypt_crack.c` program, 420
- cryptography, 393
  - laws restricting, 3
- cryptology, 393
- `crypt_test.c` program, 418
- `.ctors` (constructors), table sections
  - for, 184–188
- curly braces (`{ }`), for set of
  - instructions, 8, 9
- `current_time` variable, 97
- custom signal handlers, 322
- `cut` command, 143–144
- cyclic redundancy checksum
  - (CRC32) function, 434
- Cynosure, 118

**D**

- `daemon()` function, 321
- daemons, 321
- Data (EDX) register, 24, 361
- data integrity, programmer responsibility for, 119
- data segment, 69
  - for C variable storage, 75
- data types, of variables, 12
- datafile buffer, 151–152
- datagram socket, 198
- data-link layer (OSI), 196, 197
  - for web browser, 217, 218–219
- `datatype_sizes.c` program, 42–43
- DCMA (Digital Millennium Copyright Act) of 1998, 3
- debuggers, 23–24
- declaring
  - destructor function, 184
  - functions with data type of return value, 16–17
  - heap variable, 76
  - stack variable, 76
  - variables, 12
- `decode_ethernet()` function, 237
- `decode_ip()` function, 237
- `decode_sniff.c` file, 235–239
- `decode_tcp()` function, 236, 237
- decoherence, 399
- default gateway, ARP redirection
  - and, 241
- Denial of Service (DoS), 251–258
  - amplification attacks, 257
  - distributed DoS flooding, 258
  - ping flooding, 257
  - ping of death, 256
  - SYN flooding, 252–256
  - teardrop, 256
- dereference operator, 47
  - loading address of, 297
- DES, 398
- Destination Index (EDI) register, 24
- destructors (`.dtors`)
  - displaying contents, 185
  - overwriting section with address of injected shellcode, 190
  - table sections for, 184–188
- Deutsch, Peter, 2
- dictionary attacks, 419–422
- dictionary tables, IV-based
  - decryption, 438
- diffusion, 399
- Digital Millennium Copyright Act (DCMA) of 1998, 3
- direct parameter access, 180–182
- directory, for include files, 91
- Disassembler, 454
- distributed DoS flooding, 258
- division, remainder after, 12
- DNS (Domain Name Service), 210
- dollar sign qualifier (`$`), and direct parameter access, 180
- DoS. *See* Denial of Service (DoS)
- dotted-number notation, 203
- double word (DWORD), 29
  - converting to quadword, 302
- `drop_privs.c` program, 300
- `dsniff` program, 226, 249, 454
- `.dtors` (destructors)
  - displaying contents, 185
  - overwriting section with address of injected shellcode, 190
  - table sections for, 184–188

- dtors\_sample.c program, 184
- dump() function, 204
- dup2 system call, 307
- DWORD (double word), 29
  - converting to quadword, 302

**E**

- EAX (Accumulator) register, 24,
  - 312, 346
  - zeroing, 368
- EBP (Base Pointer) register, 24, 31,
  - 70, 73, 344–345
  - saving current values, 342
- EBX (Base) register, 24, 312, 344–345
  - saving current values, 342
- ec\_malloc() function, 91
- ECX (Counter) register, 24
- EDI (Destination Index) register, 24
- EDX (Data) register, 24, 361
- EFLAGS register, 25
- EIP register. *See* Instruction Pointer (EIP) register
- elegance, 2, 6
- encapsulation, 196
- encoded\_sockreuserestore\_dbg.s file,
  - 360–361
- encryption, 393
  - asymmetric, 400–405
    - maximum allowable key size in
      - exported software, 394
  - symmetric, 398–400
  - wireless 802.11b, 433–436
- env command, 142
- environment variables, 142
  - displaying location, 146
  - for exploiting, 148
  - PATH, 172
  - placing shellcode in, 188
  - randomization of stack
    - location, 380
  - for storing string, 378
- epoch, 97
- equal to operator (==), 14
- error checking, for malloc(), 79, 80–81
- errorchecked\_heap.c program, 80–81
- errors, off-by-one, 116–117
- escape sequences, 48
- escaped character, backslash (\)
  - for, 180
- ESI (Source Index) register, 24
- ESP (Stack Pointer) register, 24, 33,
  - 70, 73
  - shellcode and, 367
  - /etc/passwd file, 89, 153
  - /etc/services file, default ports in,
    - 207–208
- ETHERhdr structure, 245–246
- Ethernet, 218, 230
  - header for, 230
  - length of, 231
- Euclidean algorithm, 400–401
  - extended, 401–402
- Euler’s totient function, 400, 403
- examine command (GDB)
  - for ASCII table lookup, 34–35
  - to display disassembled
    - instructions, 30
  - display unit size for, 28–29
  - for memory, 27–28
- exclamation point (!), 14
- execl() function, 149, 389, 390
- execle() function, 149
- exec\_shell.c program, 296
- exec\_shell.s program, 297
- executable binaries, 21
  - creating from assembly code, 286
- execute permission, 87
- execution flow, controlling, 118
- execution of arbitrary code, 118
- execve() function, 295–296, 388–389
  - structure for, 298
- exhaustive brute-force attacks,
  - 422–423
- exit, automatically executing
  - function on, 184
- exit() function, 191, 286
  - address of, 192
- exploit buffer, 332
- exploit programs, 329
- exploit scripts, 328–333
- exploit tools, 329
- exploitation, 115
  - with BASH, 133–150
    - buffer overflows, 119–133
    - format strings, 167–193
      - direct parameter access,
        - 180–182
      - reading from arbitrary memory
        - addresses, 172

- exploitation, *continued*
  - format strings, *continued*
    - with short writes, 182–183
    - vulnerability, 170–171
  - writing to arbitrary memory
    - addresses, 173–179
  - general techniques, 118
  - heap-based overflow, 150–155
  - jackpot() function as target, 160–166
  - overflowing function pointers, 156–167
  - overwriting global offset table, 190–193
    - without log file, 352–354
- exploit\_notesearch.c program, 121
- exploit\_notesearch\_env.c program, 149–150
- extended Euclidian algorithm, 401–402

**F**

- fatal errors, displaying, 228
- fatal() function, 83, 91
- fcntl\_flags.c program, 85–86
- fcntl.h file, 84
- Feistel network, for DES, 399
- Felten, Edward, 3
- fencepost error, 116
- ffp, 454
- fg (foreground) command, 158, 332
- fgets() function, 419
- field-width option, for format parameter, 49
- file access, in C, 81–86
- file descriptors, 81
  - duplicating standard, 307–309
  - in Unix, 283
- File Not Found HTTP response, 213
- file permissions, 87–88
- File Transfer Protocol (FTP), 222
  - server, 226
- filestreams, 81
- FILO (first-in, last-out) ordering, 70
- filter, for packets, 259
- FIN scans, 264–265
  - after kernel modification, 268
  - before kernel modification, 267–268
- find\_jmpesp.c program, 386
- fingerprints
  - fuzzy, 413–417
  - host, for SSH, 410–413
- firewalls, and port-binding
  - shellcode, 314
- first-in, last-out (FILO) ordering, 70
- firstprog.c program, 19
- float data type, 12, 13, 43
- flood services, by DoS attacks, 251
- flow of execution, operations
  - controlling, 26
- Fluhrer, Mantin, and Shamir (FMS)
  - attack, 439–449
- fms.c program, 443–445
- fmt\_strings.c program, 48–49
- fmt\_uncommon.c program, 168
- fmt\_vuln.c program, 170–171
- fopen() function, 419
- for loops, 10–11
  - with assembly instructions, 309–310
  - to fill buffer, 138
- foreground (fg) command, 158, 332
- forging source address, 239
- fork() function, 149, 346
- format parameters, 48
- format strings, 167–193
  - memory for, 171
  - for printf() function, 48–51
  - short writes for exploits, 182–183
  - simplifying exploits with direct parameter access, 180–182
  - vulnerability, 170–171
- FP (frame pointer), 70
- fprintf() function, for error messages, 79
- fraggle attacks, 257
- fragmenting packets, 221
  - IPv6, 256
- frame pointer (FP), 70
- free() function, 77, 79, 152
- free speech, 4
- FTP (File Transfer Protocol), 222
  - server, 226
- funcptr\_example.c program, 100
- functionality, expansion, and errors, 117
- functions, 16–19
  - automatically executing on exit, 184
  - breakpoint in, 24

- declaring as void, 17
- for error checking, 80–81
- libraries of, 19
- local variables for, 62
- memory, string pointer
  - referencing, 228
- pointers, 100–101
  - calling without overwriting, 157
  - overflowing, 156–167
- prologue, 27, 71, 132
  - saving current register values, 342
- prototype, 17
  - for string manipulation, 39
- fuzzy fingerprints, 413–417

## G

- game\_of\_chance.c program, 102–113, 156–167
- gateway, 241
- GCC. *See* GNU Compiler Collection (GCC)
- GCD (greatest common divisor), 401
- GDB debugger, 23–24
  - address-of operator, 45
  - analysis with, 273–275
  - to control running tinywebd process, 350–352
  - to debug daemon child process, 330–331
  - disassembly syntax, 25
  - displaying local variables in stack frame, 66
  - examine command
    - for ASCII table lookup, 34–35
    - to display disassembled instructions, 30
    - for memory, 27–28
  - investigating core with, 289–290
  - investigations with, 380–384
  - print command, 31
  - shorthand commands, 28
  - stepi command, 384
  - .gdbinit file, 25
  - general-purpose registers, 24
  - GET command (HTTP), 208
  - getenv() function, 146
  - getenvaddr.c program, 147–148, 172
  - getuid() function, 89

- gethostbyname() function, 210, 211
- getuid() function, 89, 92
- Glen, Peter, 454
- glibc, heap memory management, 152
- global offset table (GOT),
  - overwriting, 190–193
- global variables, 63, 64, 75
  - memory addresses, 69
  - memory segment for, 69
- GNU Compiler Collection (GCC), 20.
  - See also* GDB debugger
  - compiler, GDB access to source code, 26
  - objdump program, 21, 184, 185
- Goldberg, Ian, 394
- GOT (global offset table),
  - overwriting, 190–193
- greater than operator (>), 14
- greater than or equal to operator (>=), 14
- greatest common divisor (GCD), 401
- Greece, ancient, 3
- grep command, 21, 143–144
  - to find kernel code sending reset packets, 267
- Grimes, Mark, 242, 454
- groups, file permissions for, 87
- Grover, Lov, 399–400

## H

- Hacker Ethic, 2
- hacking, 272–280
  - analysis with GDB, 273–275
  - attitudes toward, 451
  - and compiled program, 21
  - cycle of innovation, 319
  - essence of, 1–2
  - origins, 2
  - port-binding shellcode, 278–280
  - as problem solving, 5
  - and program crash control, 121
- hacking.h file, adding to, 204
- hacking-network.h file, 209–210, 231, 232, 272–273
- hacks, 6
- half-open scan, 264
- handle\_connection() function, 216, 342
  - breakpoint in function, 274–275
- handle\_shutdown() function, 328

- hardware addresses, 218
- hash lookup table, 423–424
- head command, 143–144
- HEAD command (HTTP), 208
- heap, 70
  - allocation function for, 75
  - buffer overflows in, 150–155
  - growth of, 75
  - memory allocation, 77
  - variable
    - declaring, 76
    - space allocated for, 77
- heap\_example.c program, 77–80
- Heisenberg uncertainty principle, 395
- “Hello, world!”, program to print, 19
- helloworld1.s program, 287–288
- helloworld3.s program, 294
- helloworld.asm program, 285–286
- helloworld.c, rewrite in assembly, 285
- Herfurt, Martin, 256
- hexadecimal dump, of standard
  - shellcode, 368
- hexadecimal notation, 21
- high-level languages, conversion to
  - machine language, 7
- Holtmann, Marcel, 256
- host fingerprints, for SSH, 410–413
- host key, retrieving from servers, 414
- host\_addr structure, for bind call, 205
- hostent structure, 210–211
- host\_lookup.c file, 211–212
- hton1() function, 202
- htons() function, 203, 205
- HTTP (Hypertext Transfer Protocol),
  - 197, 207–208, 222
- hybrid ciphers, 406–417
- Hypertext Transfer Protocol (HTTP),
  - 197, 207–208, 222

**I**

- ICMP. *See* Internet Control Message Protocol (ICMP)
- id command, 88
- idle scanning, 265–266
- IDS (intrusion detection systems),
  - 4, 354
- if statement, in BASH, 381
- ifconfig command, 316
  - for promiscuous mode setting, 224
- if-then-else structure, 8–9
  - in assembly language, 32
- in\_addr structure, 203
  - connection IP address in, 315–316
- inc operation, 25, 36
- include file, for functions, 91
- incoming connection
  - C function to accept, 199
  - listening for, 316
- incrementing variable values, 13–14
- inet\_aton() function, 203
- inet\_ntoa() function, 203, 206
- info register eip command, 28
- information theory, 394–396
- initialization vector (IV)
  - gathering, 449
  - for WEP, 434, 437, 440
    - decryption dictionary tables
      - based on, 438
- input, length check or
  - restriction on, 120
- input size, for algorithm, 397
- input validation, 365
- input.c program, 50
- input\_name() function, 156
- Instruction Pointer (EIP) register, 25,
  - 27, 40, 43, 69, 73
    - assembly instructions and, 287
    - crash from attempt to restore, 133
    - examining memory for, 28
      - as pointer, 43
      - program execution and, 69
      - shellcode and, 367
- int data type, 12
- int instruction, 285
- integers, function for converting
  - ASCII to, 59
- Intel syntax for assembly language,
  - 22, 23, 25
- Internet Control Message Protocol (ICMP), 220–221
  - amplification attacks with
    - packets, 257
  - echo messages, 256
  - Echo Request, 221
- Internet Datagram header, 232
- Internet Explorer, zero-day VML
  - vulnerability, 119
- Internet Information Server (Microsoft IIS), 117



- Internet Protocol (IP), 220
  - addresses, 197, 220
  - conversion, 203
  - data-link layer and, 218–219
  - in logs, 348
  - redirection, 438–439
  - spoofing logged, 348–352
  - IDs, predictable, 265
  - structure, 231
- interrupt 0x80, 285
- intrusion detection systems (IDS), 4, 354
- intrusion prevention systems (IPS), 354
- intrusions
  - log files and detection, 334–336
  - overlooking obvious, 336–347
- IP. *See* Internet Protocol (IP)
- IPS (intrusion prevention systems), 354
- iptables command, 407
- IPv6 packets, fragmented, 256
- IV. *See* initialization vector (IV)

## J

- jackpot() function, as exploit target, 160–166
- jle operation, 32, 310
- jmp esp instruction, 385
  - predictable address for, 388
- jmp short instruction, 292
- jobs command, 332
- John the Ripper, 422, 454
- jumps in assembly language, 26
  - conditional, 310
  - unconditional, 36

## K

- Key Scheduling Algorithm (KSA), 435, 440–442
- keystream, 398
  - reuse, 437–438
- kill command, 323, 324
- knowledge, and morality, 4
- known\_hosts file, 410
- KSA (Key Scheduling Algorithm), 435, 440–442

## L

- LaMacchia, David, 118
- LaMacchia Loophole, 117–118
- Laurie, Adam, 256
- LB (local base) pointer, 70
- lea (Load Effective Address)
  - instruction, 35, 296
- least significant byte, 174, 178
- leave instruction, 132
- less than operator (<), 14
- less than or equal to operator (<=), 14
- libc, returning into, 376–377
- libc function, finding location, 377–378
- libnet library (C), 244
  - documentation for functions, 248–249
  - release, 254
  - structures, 263
- libnet\_build\_arp() function, 248–249
- libnet\_build\_ethernet() function, 248
- libnet\_close\_link\_interface()
  - function, 249
- libnet-config program, 254
- libnet\_destroy\_packet() function, 249
- libnet\_get\_hwaddr() function, 251
- libnet\_get\_ipaddr() function, 251
- libnet\_get\_prand() function, 252
- libnet\_host\_lookup() function, 251
- libnet\_init\_packet() function, 248
- libnet\_open\_link\_interface()
  - function, 248
- libnet\_seed\_prand() function, 252
- libpcap sniffer, 228–230, 235, 260
- libraries
  - documentation, 251
  - of functions, 19
- Linux environment, 19
  - booting from CD, 4
  - nonexecutable stack, 376
  - system calls in assembly, 284–286
- linux-gate
  - bouncing off, 384–388
  - execution jump to, 386
- linux/net.h include file, 304–305
- listen() function, 199, 206
- little-endian byte order, 29, 93, 316

- LiveCD, 4, 19
  - John the Ripper, 422
  - Nemesis, 242
    - /usr/src/mitm-ssh, 407
- Load Effective Address instruction
  - (lea), 35, 296
- local base (LB) pointer, 70
- local variables, 62
  - displaying in stack frame, 66
  - memory addresses, 69
  - memory saved for, 130
- localtime\_r() function, 97
- log files
  - exploitation without, 352–354
  - and intrusion detection, 334–336
- logic, as art form, 2
- long keyword, 42
- loopback address, 217, 317–318
- loopback\_shell\_restore.s file, 346–347
- loopback\_shell.s file, 318
- looping
  - for, 10–11
  - while/until, 9–10
- lseek() function, 95
- LSFR (stream cipher), 398

## M

- MAC (Media Access Control)
  - addresses, 218, 230
- machine language, 7
  - control structures, 309
  - converting assembly to, 288
  - viewing for main() function, 21
- main() function, 19
  - command-line argument
    - access in, 58
  - disassembly of, 27
  - viewing machine code for, 21
- malloc() function, 75, 76, 77, 79
  - error checking for, 80–81
- man page
  - for arpspoof, 249
  - for ASCII, 33–34
  - for daemon(), 321
  - for exec(), 388
  - for libnet, 248, 251
  - for write(), 283
- man-in-the-middle (MitM) attacks, 406–410

- mark\_break.s file, 342–343
- mark\_restore.s file, 345
- mark.s file, 339
- mathematics, beauty in, 3
- Maxwell, James, 321
- Media Access Control (MAC)
  - addresses, 218
- memcpy() function, 139
- memory, 21–22
  - addresses
    - hexadecimal notation for, 21
    - order of, 75
    - reading from arbitrary, 172
    - writing to arbitrary, 173–179
  - allocation for void pointer, 57
  - corruption, 118
  - efficiency, vs. time for coding, 6
  - for format string, 171
  - GDB debugger to examine, 27–28
  - instructions to set up, 27
  - for local variables, 130
  - predicting address, 147
  - segmentation, 69–81, 285
  - segments, 60
    - buffer overflows in, 150–167
    - in C, 75–77
    - for variables, 119
    - violation, 60
- memory\_segments.c program, 75–77
- memset() function, 138
- Microsoft, IIS webserver, 117
- MIT model railroad club, 2
- MitM (man-in-the-middle) attacks, 406–410
- mitm-ssh package, 407, 454
- modulo reduction, 12
- morality, and knowledge, 4
- mov instruction, 25, 33, 285
  - variations, 292

## N

- %n format parameter, 48, 168–169, 173
- nasm assembler, 286, 288, 454
- Nathan, Jeff, 242, 454
- nc program, 279
- ndisasm tool, 288
- negative numbers, 42
- Nemesis, 242–248, 454

- nemesis\_arp() function, 245
- nemesis-arp.c file, 244–245
- nemesis.h file, 245–246
- nemesis-proto\_arp.c file, 246–248
- nested function calls, 62
- netcat program, 279, 309, 316, 332
- netdb.h file, 210
- netinet/in.h file, 201–202
- netstat program, 309
- Netwide Assembler (NASM), 454
- network byte order, 202–203, 316
- network layer (OSI), 196, 197
  - for web browser, 217, 220–221
- network sniffing, 224–251, 393
  - active sniffing, 239–251
  - decoding layers, 230–239
  - libpcap sniffer, 228–230
  - raw socket sniffer, 226–227
- networking, 195
  - abnormal traffic detection, 354–359
- Denial of Service, 251–258
  - amplification attacks, 257
  - distributed DoS flooding, 258
  - ping flooding, 257
  - ping of death, 256
  - SYN flooding, 252–256
  - teardrop, 256
- hacking, 272–280
  - analysis with GDB, 273–275
  - port-binding shellcode, 278–280
- network sniffing, 224–251
  - active sniffing, 239–251
  - decoding layers, 230–239
  - libpcap sniffer, 228–230
  - raw socket sniffer, 226–227
- OSI layers for web browser, 217–224
  - data-link layer, 218–219
  - network layer, 220–221
  - transport layer, 221–224
- OSI model, 196–198
- port scanning, 264–272
  - FIN, X-mas, and null scans, 264–265
  - idle scanning, 265–266
  - proactive defense, 267–272
  - spoofing decoys, 265
  - stealth SYN scan, 264
- sockets, 198–217
  - address conversion, 203
  - addresses, 200–202
  - functions, 199–200
  - network byte order, 202–203
  - server example, 203–207
  - tinyweb server, 213–217
  - web client, 207–213
- TCP/IP hijacking, 258–263
  - RST hijacking, 259–263
- newline character, for HTTP line termination, 209
- Newsham, Tim, 436–437
- nexti (next instruction) command, 31
- NFS (number field sieve), 404
- nm command, 159, 184, 185
- nmap (port scanning tool), 264
- No Electronic Theft Act, 118
- nonorthogonal quantum states, in photons, 395
- nonprintable characters, printing, 133
- NOP (no operation) sled, 140, 145, 275, 317, 332, 390
  - hiding, 362–363
  - between loader code and shellcode, 373
- not equal to operator (!=), 14
- not operator (!), 14
- notesearch.c program, 93–96
  - exploitation, 386–387
  - format string vulnerability, 189–190
  - vulnerability to buffer overflow, 137–142
- notetaker.c program, 91–93, 150–155
- note-taking program, 82
- ntohl() function, 203
- ntohs() function, 203, 206
- null bytes, 38–39, 290
  - and exploit buffer, 335
  - filling exploit buffer with, 275
  - removing, 290–295
- NULL pointer, 77
- null scans, 264–265
- number field sieve (NFS), 404
- numbers, pseudo-random, 101–102
- numerical values, 41–43
- Nyberg, Claes, 407, 454

## O

- O\_APPEND access mode, 84
- objdump program, 21, 184, 185
- O\_CREAT access mode, 84, 87
- off-by-one error, 116–117
- one-time pads, 395
- one-time password, 258
- one-way hashing algorithm, for password encryption, 153
- open files, file descriptor to reference, 82
- open() function, 87, 336–337
  - file descriptor for, 82
  - flags used with, 84
  - length of string, 83
- OpenBSD kernel
  - fragmented IPv6 packets, 256
  - nonexecutable stack, 376
- OpenSSH, 116–117
- openssh package, 414
- optimization, 6
- or instruction, 293
- OR operator, 14–15
  - for file access flags, 84
- O\_RDONLY access mode, 84
- O\_RDWR access mode, 84
- OSI model, 196–198
  - layers for web browser, 217–224
    - data-link layer, 218–219
    - network layer, 220–221
    - transport layer, 221–224
- O\_TRUNC access mode, 84
- outbound connections, firewalls and, 314
- overflow\_example.c program, 119
- overflowing function pointers, 156–167
- overflows. *See* buffer overflows
- O\_WRONLY access mode, 84
- owner, of file, 87

## P

- packet injection tool, 242–248
- packet-capturing programs, 224
- packets, 196, 198
  - capturing, 225
  - decoding layers, 230–239
  - inspecting, 359
  - size limitations, 221

- pads, 395
- password file, 153
- password probability matrix, 424–433
- passwords
  - cracking, 418–433
    - dictionary attacks, 419–422
    - exhaustive brute-force attacks, 422–423
    - hash lookup table, 423–424
    - length of, 422
    - one-time, 258
- PATH environment variable, 172
- payload smuggling, 359–363
- pcalc (programmer’s calculator), 42, 454
- pcap libraries, 229
- pcap\_fatal() function, 228
- pcap\_lookupdev() function, 228
- pcap\_loop() function, 235, 236
- pcap\_next() function, 235
- pcap\_open\_live() function, 229, 261
- pcap\_sniff.c program, 228
- percent sign (%), for format parameter, 48
- Perl, 133
- permissions for files, 87–88
- perrot() function, 83
- photons, nonorthogonal quantum states in, 395
- physical layer (OSI), 196, 197
  - for web browser, 218
- pigeonhole principle, 425
- ping flooding, 257
- ping of death, 256
- ping utility, 221
- plaintext, for protocol structure, 208
- play\_the\_game() function, 156–157
- PLT (procedure linkage table), 190
- pointer, to sockaddr structure, 201
- pointer arithmetic, 52–53
- pointer variables
  - dereferencing, 53
  - typecasting, 52
- pointer.c program, 44
- pointers, 24–25, 43–47
  - function, 100–101
  - to structs, 98
- pointer\_types.c program, 52
- pointer\_types2.c program, 53–54
- pointer\_types3.c program, 55

- pointer\_types4.c program, 56
- pointer\_types5.c program, 57
- polymorphic printable ASCII shellcode, 366–376
- pop instruction, 287
  - and printable ASCII, 368
- popping, 70
- port scanning, 264–272
  - FIN, X-mas, and null scans, 264–265
  - idle scanning, 265–266
  - proactive defense, 267–272
  - spoofing decoys, 265
  - stealth SYN scan, 264
- port scanning tool (nmap), 264
- port-binding shellcode, 278–280, 303–314
- ports, root privileges for binding, 216
- position-independent code, 286
- PowerPC processor architecture, 20
- ppm\_crack.c program, 428–433
- ppm\_gen.c program, 426–428
- presentation layer (OSI), 196
- PRGA (Pseudo-Random Generation Algorithm), 435, 436
- print command (GDB), 31
- print error, 83
- printable ASCII shellcode, polymorphic, 366–376
- printable characters, program to calculate, 369
- printable\_helper.c program, 369–370
- printable.s file, 371–372
- printf() function, 19–20, 35, 37, 47
  - format strings for, 48–51, 167
- printing nonprintable characters, 133
- print\_ip() function, 254
- private key, 400
- privileges, 273, 299
- priv\_shell.s program, 301
- probability, conditional, 114
- problem solving
  - with hacking, 1–2
  - hacking as, 5
- procedure linkage table (PLT), 190
- procedure prologue, 71
- process, suspending current, 158
- process hijacking, 118
- processor, assembly language specificity for, 7

- product ciphers, 399
- programming
  - access to heap, 70
  - as artistic expression, 2
  - basics, 6–7
  - control structures, 8–11
    - if-then-else, 8–9
    - while/until loops, 9–10
  - variables, 11–12
- programs, results from, 116
- promiscuous mode, 224
  - capturing in, 229
- pseudo-code, 7, 9
- Pseudo-Random Generation Algorithm (PRGA), 435, 436
- pseudo-random numbers, 101–102
- public key, 400
- punch cards, 2
- push instruction, 287, 298
  - and printable ASCII, 368
- pushing, 70
- Pythagoreans, 3

**Q**

- quadword, converting
  - doubleword to, 302
- quantum factoring algorithm, 404–405
- quantum key distribution, 395–396
- quantum search algorithm, 399–400
- quotation marks ("), for include files, 91

**R**

- RainbowCrack, 433
- rand() function, 101
- rand\_example.c program, 101–102
- random numbers, 101–102
- randomization, execl() function and, 390, 391
- randomized stack space, 379–391
- raw socket sniffer, 226–227
- raw\_tcpsniff.c program, 226–227
- RC4 (stream cipher), 398, 434, 435–436
- read() function, file descriptor for, 82
- read permission, 87
- read-only permission, for text segment, 69

- Recording Industry Association of America (RIAA), 3
- recv() function, 199, 206
- recv\_line() function, 209, 273, 335, 342
- redirection attack, 240–241
- registers, 23, 285, 292
  - displaying, 24
  - for x86 processor, 23
  - zeroing, with polymorphic shellcode, 366
- relatively prime numbers, 400
- remainder, after division, 12
- remote access, to root shell, 317
- remote targets, 321
- Request for Comments (RFC)
  - 768, on UDP header, 224
  - 791, on IP headers, 220, 232
  - 793, on TCP header, 222–223, 233–234
- ret instruction, 132, 287
- ret2libc, 376–377
- return address, 70
  - finding exact location, 139
  - overwriting, 135
  - in stack frame, 131
- return command, 267
- Return Material Authorization (RMA), 221
- return value of function, declaring
  - function with data type of, 16–17
- RFC. *See* Request for Comments (RFC)
- RIAA (Recording Industry Association of America), 3
- Rieck, Konrad, 413, 454
- RMA (Return Material Authorization), 221
- Ronnick, Jose, 454
- root
  - privileges, 153, 273
  - to bind port, 216
  - shell to restore, 301
  - shell
    - obtaining, 188
    - overflow to open, 122
    - remote access, 317
    - socket reuse, 355–359
    - spawning, 192
    - spawning with child process, 346
    - user, 88
- RSA Data Security, 394, 400, 404
- RST hijacking, 259–263
- rst\_hijack.c program, 260–263
  - modification, 268
- run time of simple algorithm, 397

## S

- %s format parameter, 48, 172
- Sadmind worm, 117
- salt value, 153–154
  - for password encryption, 419
- Sasser worm, 319
- saved frame pointer (SFP), 70, 72–73, 130
- S-box array, 435
- scanf() function, 50
- scope of variables, 62–69
- scope.c program, 62
- scope2.c program, 63–64
- scope3.c program, 64–65
- script kiddies, 3
- Secure Digital Music Initiative (SDMI), 3
- Secure Shell (SSH)
  - differing host fingerprints, 410–413
  - protections against identity spoofing, 409–410
- Secure Sockets Layer (SSL), 393
  - protections against identity spoofing, 409–410
- security
  - changing vulnerabilities, 388
  - computational, 396
  - impact of mistakes, 118
  - unconditional, 394
- seed number, for random sequence
  - of numbers, 101
- segmentation fault, 60, 61
- semicolon (;), for instruction end, 8
- send() function, 199, 206
- send\_string() function, 209
- seq command, 141
- sequence numbers, for TCP, 222, 224
- server example, displaying packet data, 204

- session layer (OSI), 196
  - for web browser, 217
- set disassembly intel command, 25
- set user ID (setuid) permission, 89
- setuid() function, 299
- setresuid() system call, 300–301
- setsockopt() function, 205
- SFP (saved frame pointer), 70
- Shannon, Claude, 394
- shell command, executing like
  - function, 134
- shellcode, 137, 281
  - argument as placement option, 365
  - assembly language for, 282–286
  - connect-back, 314–318
  - creating, 286–295
  - jump to, 386
  - memcpy() function to copy, 139
  - memory location for, 142
  - overwriting .dtors section with
    - address of injected, 190
  - placing in environment
    - variable, 188
  - polymorphic printable ASCII,
    - 366–376
  - port-binding, 278–280, 303–314
  - proof of functioning, 336
  - reducing size, 298
  - restoring tinyweb daemon
    - execution, 345
  - shell-spawning, 295–303
    - and webserver, 332
    - zeroing registers, 294
- shellcode.s program, 302–303
- Shor, Peter, 404–405
- short keyword, 42
- short writes, for format string
  - exploits, 182–183
- shorthand expressions, for arithmetic operators, 13–14
- shroud.c program, 268–272
- sigint\_handler() function, 323
- SIGKILL signal, 324
- signal() function, 322
- signal\_example.c program, 322–323
- signal\_handler() function, 323
- signals, for interprocess communication in Unix, 322–324
- signed numerical values, 41
- Simple Mail Transfer Protocol (SMTP), 222
- simplenote.c program, 82–84
- simple\_server.c file, 204–207
- sizeof() function, 58
- sizeof() macro (C), 42
- Sklyarov, Dmitry, 3–4
- SMTP (Simple Mail Transfer Protocol), 222
- smurf attacks, 257
- sniffing packets
  - active, 239–251
  - in promiscuous mode, 225
- sockaddr structure, 200–202, 305, 306
  - pointer to, 201
- sockaddr\_in structure, 348
- socket() function, 199, 200, 205, 314
- socketcall() system call (Linux), 304
- socket\_reuse\_restore.s file, 357
- sockets, 198–217, 307
  - address conversion, 203
  - addresses, 200–202
  - file descriptor for accepted connection, 206
  - functions, 199–200
  - reuse, 355–359
  - server example, 203–207
  - tinyweb server, 213–217
  - web client, 207–213
- software piracy, 118
- Solar Designer, 422, 454
- Song, Dug, 226, 249, 454
- source address, manipulating, 239
- Source Index (ESI) register, 24
- Sparc processor, 20
- spoofing, 239–240
  - logged IP address, 348–352
  - packet contents, 263
- sprintf() function, 262
- srand() function, 101
- SSH. *See* Secure Shell (SSH)
- SSL (Secure Sockets Layer), 393
  - protections against identity spoofing, 409–410
- stack, 40, 70, 128
  - arguments to function call in, 339
  - assembly instructions using,
    - 287–289

- stack, *continued*
  - frame, 70, 74, 128
    - displaying local variables in, 66
    - instructions to set up and remove structures, 341
  - growth of, 75
  - memory in, 77
  - nonexecutable, 376–379
  - randomized space, 379–391
  - role with format strings, 169
  - segment, 70
  - variables
    - declaring, 76
    - and shellcode reliability, 356
- Stack Pointer (ESP) register, 24, 33, 70, 73
  - shellcode and, 367
- stack\_example.c program, 71–75
- Stallman, Richard, 3
- standard error, 307
- standard input, 307, 358
- standard input/output (I/O)
  - library, 19
- standard output, 307
- static function memory, string pointer
  - referencing, 228
- static keyword, 75
- static variables, 66–69
  - memory addresses, 69
  - memory segment for, 69
- static.c program, 67
- static2.c program, 68
- status flags, cmp operation to set, 311
- stderr argument, 79
- stdio header file, 19
- stealth, by hackers, 320
- stealth SYN scan, 264
- stepi command (GDB), 384
- storage space, vs. computational
  - power, 424
- strace program, 336–338, 352–353
- strcat() function, 121
- strcpy() function, 39–41, 365
- stream ciphers, 398
- stream sockets, 198, 222
- string.h, 39
- strings, 38–41
  - concatenation in Perl, 134
  - encoding, 359–362
- strlen() function, 83, 121, 209
- strncasecmp() function, 213
- strstr() function, 216
- structs, 96–100
  - access to elements, 98
- su command, 88
- sub instruction, 293, 294
- sub operation, 25
- sudo command, 88, 90
- superposition, 399–400
- suspended process, returning to, 158
- switched network environment,
  - packets in, 239
- symmetric encryption, 398–400
- SYN flags, 223
- SYN flooding, 252–256
  - preventing, 255
- SYN scan
  - preventing information leakage
    - with, 268
  - stealth, 264
- syncookies, 255
- synflood.c file, 252–254
- sys/stat.h file, 84
  - bit flags defined in, 87
- system calls, manual pages for, 283
- system daemons, 321–328
- system() function, 148–149
  - returning into, 377–379

## T

- TCP. *See* Transmission Control Protocol (TCP)
- tcpdump, 224, 226
  - BPFs for, 259
  - source code for, 230
- tcphdr structure (Linux), 234
- TCP/IP, 197
  - connection, telnet to
    - webserver, 208
  - hijacking, 258–263
  - stack, SYN flood attempt to exhaust
    - states, 252
- tcp\_v4\_send\_reset() function, 267
- teardrop, 256
- telnet, 207, 222
  - to open TCP/IP connection to
    - webserver, 208
- temporary variable, from print
  - command, 31



- text segment, of memory, 69
- then keyword, 8–9
- th\_flags field, of tcp\_hdr structure, 234
- time() function, 97
- time\_example.c program, 97
- time\_example2.c program, 98–99
- time\_ptr variable, 97
- time/space trade-off attack, 424
- timestamp() function, 352
- tiny\_shell.s program, 298–299
- tinyweb.c program
  - converting to system daemon, 321
  - as daemon, 324–328
  - exploit for, 275
  - vulnerability in, 273
- tinywebd.c program, 325–328, 355
  - exploit tool, 329–333
  - log file, 334
- tinyweb\_exploit.c program, 275
- tinyweb\_exploit2.c program, 278
- tm time struct, 97
- translator, for machine language, 7
- Transmission Control Protocol (TCP), 198, 222
  - connection for remote shell access, 308–309
  - flags, 222
  - opening connection, 314
  - packet header, 233–234
  - sniffing, with raw sockets, 226
  - structure, 231
- transport layer (OSI), 196, 197
  - for web browser, 217, 221–224
- Triple-DES, 399
- two's complement, 42, 49
  - to remove null bytes, 291
- typecasting, 51–58
  - from tm struct pointer to integer pointer, 98
- typecasting.c program, 51
- typedef, 245
- typeless pointers, 56
- types. *See* data types

## U

- UDP (User Datagram Protocol), 198–199, 222, 224
  - echo packets, amplification attacks with, 257

- uid\_demo.c program, 90
- ulimit command, 289
- uname command, 134
- unary operator
  - address-of operator, 45
  - dereference operator, 47, 50
- unconditional jumps, in assembly
  - language, 36
- unconditional security, 394
- unencrypted data transmission, 226
- Unicode character set, 117
- Unix systems
  - manual pages, 283
  - signals for interprocess communication, 322–324
  - time on, 97
- unsigned keyword, 42
- unsigned numerical values, 41
  - integer for pointer address, 57
- unswitched network, 224
- until loop, 10
- update\_info.c file, 363–364
- usage() function, 82
- User Datagram Protocol (UDP), 198–199, 222, 224
  - echo packets, amplification attacks with, 257
- user IDs, 88–96
  - displaying notes written by, 93
  - setting effective, 299
- users, file permissions for, 87
- user-supplied input, length check or restriction on, 120
- /usr/include/asm-i386/unistd.h file, 284–285
- /usr/include/asm/socket.h file, 205
- /usr/include/bits/socket.h file, 200, 201
- /usr/include/if\_ether.h file, 230
- /usr/include/linux/if\_ether.h file, 230
- /usr/include/netinet/ip.h file, 230, 231–232
- /usr/include/netinet/tcp.h file, 230, 233–234
- /usr/include/stdio.h file, 19
- /usr/include/sys/sockets.h file, 199
- /usr/include/time.h file, 97
- /usr/include/unistd.h file, 284
- /usr/src/mitm-ssh, 407

## V

- values
  - assigning to variable, 12
  - returned by function, 16
- variables, 11–12
  - arithmetic operators for, 12–14
  - C compiler and data type, 58
  - comparison operators for, 14–15
  - scope, 62–69
  - structs, 96–100
  - temporary, from print command, 31
  - typecasting, 51–58
- void keyword, 56
  - for declaring function, 17
- void pointer (C), 56, 57
- vuln.c program, 377
- vulnerabilities
  - format strings, 170–171
  - in software, 451–452
  - stack-based, 122–133
  - in tinyweb.c program, 273
  - zero-day VML, 119

## W

- warnings, about pointer data type, 54
- web browser, OSI layers for, 217–224
- web client, 207–213
- web requests, processing after intrusion, 336
- webserver
  - telnet for TCP/IP connection to, 208
  - tinyweb server, 213–217
- webserver\_id.c file, 212–213
- WEP (Wired Equivalent Privacy), 433, 434–435
  - attacks, 436–449

- where command, 61
- while/until loops, 9–10
- Wired Equivalent Privacy (WEP), 433, 434–435
  - attacks, 436–449
- wireless 802.11b encryption, 433–436
- word, 28–29
- worms, 119
- Wozniak, Steve, 3
- WPA wireless protocol, 448
- write() function, 83
  - file descriptor for, 82
  - manual page for, 283
  - pointer for, 92
- write permission, 87
  - for text segment, 69

## X

- %x format parameter, 171, 173
  - field-width option, 179
- x/3xw command, 61
- x86 processor, 20, 23–25
  - assembly instructions for, 285
- xchg (exchange) instruction, 312
- X-mas scans, 264–265
- xor instruction, 293, 294
- xtool\_tinywebd\_reuse.sh script, 358
- xtool\_tinywebd.sh script, 333
- xtool\_tinywebd\_silent.sh script, 353–354
- xtool\_tinywebd\_spoof.sh script, 349–350
- xtool\_tinywebd\_stealth.sh script, 335

## Z

- zeroing registers, 294
  - EAX (Accumulator) register, 368
  - with polymorphic shellcode, 366