

---

# Index

---

## A

- abort signals, 100
- Accept header, 122
- Accept-Encoding header, 122
- Accept-Language header, 122
- access
  - control, 29
  - violation handler, 340-342
- ActiveX controls, 284
- Adobe Acrobat PDF control, 294-296
  - fuzzer development, 287-289
    - heuristics, 298
    - loadable controls, enumerating, 289-293
    - monitoring, 299
    - properties, methods, parameters, and types, 293-297
    - test cases, 298
  - history, 25
  - overview, 285-287
  - vulnerabilities, 273-275
  - WinZip FileView, 298
- adbg, 370
- address bar spoofing vulnerabilities, 281
- addresses
  - reading, 476-478
  - writing to, 478-479
- Adobe
  - Acrobat
    - PDF control, 294-296
    - shell script workaround, 192
  - Macromedia Shockwave Flash file format. *See* SWF
- agents
  - fault detection, 233
  - network monitor, 402
  - process monitor, 403
  - Sulley sessions, 402-404
  - VMWare control, 403
- AIM (AOL Instant Messenger) protocol, 49-52
  - logon credentials sent, 52
  - server reply, 51
  - username, 51
- Aitel, Dave, 23
- algorithms
  - GAs, 431-435
    - CFG with connecting path, 433
    - CFG with exit nodes, 434
    - CFG with potential vulnerability, 433
    - fitness function, 432
    - reproduction function, 431
    - Sidewinder, 433-434
    - as stochastic global optimizers, 432
  - Needleman-Wunsch, 428
  - Smith-Waterman local sequence alignment, 428
  - Unweighted Pairwise Mean by Arithmetic Averages, 429
- alignment of sequences, 427
- amino acid sequence alignment, 427
- antiparser, 354-356
- AOL Instant Messenger. *See* AIM protocol
- APIs
  - antiparser, 354-356
  - CreateProcess, 321
  - debugging APIs, 179
  - Windows debugging, 320-323, 332-333
- apKeywords() data type, 355
- Apple Macbook vulnerability, 228
- application layer vulnerabilities, 230

- application logs, fault detection, 233
- applications
  - manually testing, 10
  - service provider (ASP), 114
  - setuid, 37
  - sweeping, 10
  - web application fuzzers, 41
    - buffer overflow vulnerability, 130
    - configuring, 118-119
    - exceptions, detecting, 135-136
    - heap overflow vulnerability, 129
    - inputs. *See* web application fuzzers, inputs
    - overview, 113-115
    - targets, 117-118
    - technologies, 115
    - vulnerabilities, 132-135
- archiving utility vulnerabilities, 170
- argv module, 104-106
- ASCII text files, 202
- ASP (application service provider), 114
- ASP .Net, 116
- asynchronous sockets (WebFuzz), 150-153
- attack heuristics, 353
- audits, saving, 204-205
- authentication (weak), 133
- Autodafe[as], 369-371
- automation
  - benefits, 73
    - human computation power, 73
    - reproducibility, 74
  - binary testing, 17-18
  - debugger monitoring, 481
    - advanced example, 486-489
    - architecture, 481
    - basic example, 482-484
    - PaiMei crash binning utility, 487-489
  - environment variable fuzzing
    - GDB method, 96-97
    - getenv function, 98
    - library preloading, 98-99
  - protocol dissection
    - bioinformatics, 427-431
    - genetic algorithms, 431-435
    - heuristic-based, 421-427
  - source code auditing tools, 7
  - length calculation, 352
  - protocol generation testing, 36
  - white box testing, 6-8
- availability
  - black box testing, 13
  - gray box testing, 18
  - white box testing, 9
- av\_handler() routine, 483
- AWStats Remote Command Execution Vulnerability
  - website, 118
- AxMan, 25, 275

**B**

- backdoor fuzzing limitations, 30
- basic blocks
  - defined, 440
  - start/stop points, 440
  - tracking, 444
- Beddoe, Marshall, 428
- BeginRead() method, 152
- BeginWrite() method, 151
- beSTORM, 138, 508
- binary auditing
  - automated, 17-18
  - manual, 14-16
- binary files (FileFuzz), 201
- binary protocols, 49-52
- binary visualization, 439-441
- BinAudit, 18
- BindAdapter() function, 259
- bioinformatics, 427-431
- bits, 93, 385
- bit\_field class, 377
- black box testing, 9
  - fuzzing, 12
  - manual, 10
  - pros/cons, 13-14
  - tool (beSTORM), 138, 508
- Blaster worm, 225
- blocks
  - basic
    - defined, 440
    - start/stop points, 440
    - tracking, 444
  - helpers, 395
    - checksums, 396
    - example, 397
    - repeaters, 396
    - sizers, 395-396
  - identifiers, 58
  - protocol
    - modeling, 242
    - representation, 361-362
    - sizes, 58
  - Sulley framework, 392
    - dependencies, 394-395
    - encoders, 394
    - grouping, 392-393
    - helpers, 395-397
- BLOSUM (Blocks Substitution Matrix), 429
- boundary value analysis (BVA), 21
- BoundsChecker, 493
- bp\_set() routine, 333
- break command, 97
- BreakingPoint, 509
- breakpoints
  - handler, 342
  - hardware, 324

- software, 324
  - WS2\_32.dll recv(), 336
  - Brightstor backup software vulnerabilities, 424
  - browser fuzzers, 41-42
    - ActiveX controls, 287-289
      - heuristics, 298
      - loadable controls, enumerating, 289-293
      - monitoring, 299
      - properties, methods, parameters, and types, 293-297
      - test cases, 298
    - approaches, 269-271
    - fault detection, 282
    - heap overflows, 277-279
    - history, 24
    - inputs, 271
      - ActiveX controls, 273-275
      - client-side scripting, 276
      - CSS, 275-276
      - Flash, 279
      - HTML headers, 271
      - HTML tags, 271-273
      - URLs, 280
      - XML tags, 273
    - methods, 269, 275-276, 279-280
    - Month of Browser Bugs, 268
    - overview, 268
    - targets, 269
    - vulnerabilities, 280-282
  - brute force fuzzing
    - file formats, 172-173
    - network protocols, 231
  - brute force testing, 36
  - btnRequest\_Click() method, 153
  - buffer overflows
    - case study (Web applications), 158-160
    - vulnerabilities, 130
      - web applications, 133
      - web browsers, 281
  - BugScam, 17
  - BVA (boundary value analysis), 21
  - byref() function, 318
- ## C
- call graphs, 439
  - callbacks
    - access violation handler, 340-342
    - breakpoint handler, 342
    - registering, 401-402
  - Cascading Style Sheet. *See* CSS
  - cc\_hits database, 455-457
  - CCR (Command Continuation Request) example, 78-79
  - cc\_tags database, 455
  - CFGs (control-flow graphs), 433, 440-441
    - disassembled dead listing, 441
    - GAs
      - connecting path, 433
      - exit nodes, 434
      - potential vulnerability, 433
  - CGI (Common Gateway Interface), 115
  - character translations, 85
  - character-delimited fields, 47
  - checksums
    - as block helpers, 396
    - protocols, 58
  - child processes, forking off and tracing, 185-187
  - choosing
    - fuzz values, 480-481
    - hook points, 331
    - memory mutation locations, 331
    - packet capture libraries, 256-257
    - protocol fields, 47
    - web application inputs, 119-121
      - cookies, 129
      - headers, 128
      - method, 123-125
      - post data, 130
      - protocol, 128
      - request-URI, 126-128
  - CISC (Complex Instruction Set Computer)
    - architecture, 438
  - class IDs (CLSIDs), 285
  - classes
    - apKeywords(), 355
    - bit\_field, 377
    - PyDbg, 332-334, 340-345
    - TcpClient, 148-149
  - clfuzz, 38
  - clients
    - Ethereal, 335
    - launching, 334
  - client-side vulnerabilities, 223, 276, 280-282
  - CLSIDs (class IDs), 285
  - code
    - coverage. *See* tracking
    - reusability
      - frameworks, 354
      - Peach, 366
  - Code Red worm, 225
  - Codenomicon, 41, 510-511
    - foundation, 23
    - HTTP test tools, 41, 138
  - CodeSpy website, 7
  - coding phase (SDLC), 501-502
  - COM (Component Object Model), 284
    - ActiveX controls
      - Adobe Acrobat PDF control, 294-296
      - fuzzer development. *See* ActiveX controls, fuzzer development
      - overview, 285-287
      - WinZip FileView, 298
    - history, 284
    - interfaces, 284

- objects, 284
  - Raider, 25, 42, 275
  - VARIANT data structure, 293
  - Command Continuation Request (CCR) example, 78-79
  - command-line arguments, 89
  - command-line fuzzers, 37-38
  - commands
    - break, 97
    - CREA, 248
    - commands, 97
    - execution vulnerabilities, 281
    - find, 93
    - injection vulnerabilities, 87
  - commands command, 97
  - commercial tools, 507
    - beSTORM, 138, 508
    - BreakingPoint, 509
    - Codonomicon, 41, 510-511
      - foundation, 23
      - HTTP test tools, 41, 138
    - Holodeck, 514-515
    - Mu-4000, 512
    - ProtoVer Professional, 512
  - Common Gateway Interface (CGI), 115
  - community involvement (frameworks), 43
  - compile time checkers, 6
  - Complex Instruction Set Computer (CISC)
    - architecture, 438
  - complex protocols, 40
  - complexity
    - frameworks, 44
    - gray box testing, 18
    - in-memory fuzzing, 43
    - white box testing, 9
  - Component Object Model. *See* COM
  - Computer Associates' Brightstor backup software
    - vulnerabilities, 424
  - Compuware DevPartner BoundsChecker, 493
  - configuring web applications, 118-119
  - CONNECT method, 125
  - Connection header, 123
  - connections
    - dropped, 135
    - testing, 472
  - ContinueDebugEvent() routine, 323
  - control-flow graphs. *See* CFGs
  - Convert, 367
  - Cookie header, 123
  - cookies, 129
  - coverage
    - black box testing, 14
    - gray box testing, 18
    - white box testing, 9
  - crash binning tool, 487-489
  - crash locations, viewing, 405-406
  - crashbin\_explorer.py tool, 405
  - CRC (Cyclic Redundancy Check) values, 353
  - CREA command, 248
  - CreateFile() method, 299
  - CreateProcess() function, 203, 10, 203, 299, 321
  - create\_string\_buffer() function, 318
  - cross-domain restriction bypass vulnerabilities, 281
  - cross-site scripting (XSS), 132, 163-166
  - CSRF (cross site request forgery), 134
  - CSS (Cascading Style Sheet)
    - CSSDIE fuzzer, 24, 42, 275-276
    - vulnerabilities, 275-276
  - CSSDIE, 24, 275, 42, 276
  - ctypes module, 318
  - Cyclic Redundancy Check (CRC) values, 353
- ## D
- data
    - capture
      - networks, 251
      - ProtoFuzz design, 258-259
      - PStalker, 454
    - exploration, 453
    - generating
      - CCR example, 78-79
      - character translations, 85
      - command injection, 87
      - directory traversal, 86
      - field delimiters, 83-84
      - format strings, 85
      - generators, 364
      - integer values, 79-81
      - pseudo-random data, 353
      - string repetitions, 82
      - SWF fuzzing test case, 384
    - link layer vulnerabilities, 228
    - packets, assembling, 250-251
    - parsing
      - networks, 251-252
      - ProtoFuzz design, 259-261
    - sources
      - Dfuz, 358
      - PStalker, 452-453
    - storage, 454-457
    - types
      - apKeywords(), 355
      - block helpers, 395-397
      - blocks, 392-395
      - character translations, 85
      - command injection, 87
      - delimiters, 391
      - directory traversal, 86
      - field delimiters, 83-84
      - format strings, 85
      - integers, 79-81, 390-391
      - legos, 398-399
      - passing, 318

- smart data sets, 81
- strings, 82, 391
- Sulley framework, 388-390
- databases
  - cc\_hits, 455-457
  - cc\_tags, 455
- DataRescue Interactive Disassembler Pro (IDA Pro), 439
- DBI (Dynamic Binary Instrumentation), 68, 491-493
  - DynamoRIO, 491
  - error detection, 68
  - fault detection, 492
  - Pin, 492
  - tool development, 492
- DCOM (Distributed COM), 284
- DDE (Dynamic Data Exchange), 284
- dead listing, 441
- debug event loops, 322-323
- debug exception events, 323
- DebugActiveProcess() routine, 321
- debuggers, 16
  - adbg, 370
  - automated monitoring, 481
    - advanced example, 486-489
    - architecture, 481
    - basic example, 482-484
    - PaiMei crash binning utility, 487-489
  - DBI, 491-493
    - DynamoRIO, 491
    - fault detection, 492
    - Pin, 492
  - fault detection, 233
- GDB, 16
- OllyDbg, 16, 335
  - before/after demangle, 336
  - conceptual diagram, 339
  - parse routine, 338
  - restore point, 338
  - WS2\_32.dll recv() breakpoint, 336
- process information, 178
- web application fuzzing, 136
- Web browser errors, 282
- WinDbg, 16
- debugging API (Windows), 320-323
  - process information, 179
  - process instrumentation with PyDbg, 332-333
- DebugSetProcessKillOnExit() routine, 321
- DEBUG\_EVENT structure, 323
- decompilers, 16
- DELETE method, 124
- delimiters (Sulley framework), 391
- DeMott, Jared, 366
- denial-of-service. *See* DoS attacks
- dependencies (blocks), 394-395
- depth (process), 64-66
- design
  - FileFuzz
    - applications, launching, 211-213
    - exception detection, 213-217
    - files, creating, 210
    - source files, reading, 210-211
    - writing to files, 211
  - logic, 30
  - ProtoFuzz, 257, 262
    - data capture, 258-259
    - fuzz variables, 261
    - hexadecimal encoding/decoding, 262
    - network adapters, 257-258
    - parsing data, 259-261
  - SDLC, 500-501
  - WebFuzz
    - asynchronous sockets, 150-153
    - requests, generating, 153-155
    - responses, receiving, 155-156
    - TcpClient class, 148-149
- detecting
  - errors, 67-69
    - DBI platforms, 68
    - debug clients, 67
    - ping checks, 67
  - exceptions
    - detection engines, 183
    - FileFuzz, 203-204, 213-217
  - faults
    - crash locations, viewing, 405-406
    - DBI, 492
    - execution control, transferring, 475-476
    - first-chance versus last-chance exceptions, 489-491
    - format string vulnerability, 477
    - frameworks, 353
    - graphical representation, 406
    - in-memory fuzzing, 311-312
    - network protocol fuzzing, 232-233, 254
    - page faults, 474
    - primitive, 472-474
    - reading addresses, 476-478
    - software memory corruption vulnerability
      - categories, 475
    - stack layout example, 477
    - Web browsers, 282
      - writing to addresses, 478-479
  - file format fuzzing, 178-179
  - local fuzzing problems, 99-101
  - web application exceptions, 135-136
- developers, 503-504
- development
  - ActiveX fuzzer, 287-289
    - heuristics, 298
    - loadable controls, enumerating, 289-293
    - monitoring, 299
    - properties, methods, parameters, and types, 293-297
    - test cases, 298
  - DBI tools, 492
  - environment integration of fuzzing, 516

- file format fuzzing
    - core fuzzing engine, 184-185
    - exception detection engine, 183
    - exception reporting, 183-184
    - forking off/tracing child processes, 185-187
    - interesting UNIX signals, 187
    - not interesting UNIX signals, 188
  - FileFuzz, 209
    - applications, launching, 211-213
    - approach, 209
    - design, 210-217
    - exception detection, 213-217
    - files, creating, 210
    - language, choosing, 210
    - source files, reading, 210-211
    - writing to files, 211
  - iFuzz, 105-107
  - ProtoFuzz
    - design, 258-262
    - goals, 255
    - packet capture library, choosing, 256-257
    - programming language, 256
  - software development lifecycles. *See* SDLC
  - tracking tool, 442-443
    - basic blocks, tracking, 444
    - cross-referencing, 447
    - instruction execution, tracing, 444-445
    - recordings, filtering, 446
    - target profiling, 443-444
  - web application fuzzers
    - approach, 148
    - asynchronous sockets, 150-153
    - programming language, choosing, 148
    - requests, generating, 153-155
    - responses, receiving, 155-156
    - TcpClient class, 148-149
  - DevInspect tool, 504
  - Dfuz, 357-360
    - data sources, 358
    - functions, 357-358
    - lists, declaring, 358
    - protocols, emulating, 359
    - rule files, 359-360
    - variables, defining, 357
  - DHTML (Dynamic HTML), 24
  - directory traversal
    - case study, 156-157
      - Ipswitch Imail Web Calendaring, 157
      - Trend Micro Control Manager, 156
    - vulnerabilities, 86, 132
  - disassembled binaries, visualizing, 439-441
  - disassemblers, 15
  - disassembly-level heuristics, 426-427
  - discussion board vulnerabilities, 117
  - Distributed COM (DCOM), 284
  - documentation, 62
  - DOM (Document Object Model), 285
  - DOM-Hanoi, 42
  - DoS (denial-of-service) attacks
    - file formats, 175
    - web applications, 132
    - web browsers, 280
  - DownloadFile() method, 299
  - drivers, 255
  - Dynamic Binary Instrumentation. *See* DBI
  - Dynamic Data Exchange (DDE), 284
  - Dynamic HTML (DHTML), 24
  - DynamoRIO DBI system, 491
- ## E
- ECMAScript, 276
  - Eddington, Michael, 40
  - elements of protocols
    - block identifiers/sizes, 58
    - checksums, 58
    - name-value pairs, 57
  - encoders (blocks), 394
  - Enterprise Resource Planning (ERP), 117
  - enumerating
    - environment variables
      - GDB method, 96-97
      - getenv function, 98
      - library preloading, 98-99
    - loadable ActiveX controls, 289-293
  - environment variables, 38-39, 90-92
    - GDB method, 96-97
    - getenv function, 98
    - library preloading, 98-99
  - ERP (Enterprise Resource Planning), 117
  - error detection, 67-69
    - DBI platforms, 68
    - debug clients, 67
    - ping checks, 67
  - Ethereal client, server data capture, 335
  - event logs
    - debugging,, 322-323
    - logs
      - process information, 178
      - web application fuzzing, 136
      - Web browser errors, 282
  - Evron, Gadi, 25
  - Excel eBay vulnerability, 199
  - exceptional elements, 510
  - exceptions
    - debug exception events, 323
    - detecting, 183, 203-204, 213-217
    - first-chance versus last-chance, 489-491
    - monitoring, 28
    - reporting, 183-184
    - web applications, 135-136, 147
  - Execute() method, 299
  - executing
    - fuzz data, 28
    - instructions, 438. *See also* tracking
  - execution control, transferring, 475-476

- 
- exit nodes (GAs), 434
  - exploitability, 28
  - External Data Representation (XDR), 230
  - F**
  - fault detection
    - crash locations, viewing, 405-406
    - DBI, 492
    - execution control, transferring, 475-476
    - first-chance versus last-chance exceptions, 489-491
    - format string vulnerability, 477
    - frameworks, 353
    - graphical representation, 406
    - in-memory fuzzing, 311-312
    - network protocol fuzzing, 232-233
      - network protocol fuzzing, 254
    - page faults, 474
    - primitive, 472-474
    - reading addresses, 476-478
    - software memory corruption vulnerability
      - categories, 475
    - stack layout example, 477
    - web browsers, 282
    - writing to addresses, 478-479
  - fencing memory allocation, 492
  - field delimiters, 83-84
  - file format fuzzers, 39, 54-57
    - benefits, 221
    - case study, 217-220
    - detection, 178-179
    - development, 209
      - applications, launching, 211-213
      - approach, 209
      - core fuzzing engines, 184-185
      - design, 210-217
      - exception detection, 183-184, 213-217
      - files, creating, 210
      - language, choosing, 210
      - source files, reading, 210-211
      - writing to files, 211
  - FileFuzz
    - applications, launching, 203
    - ASCII text files, 202
    - audits, saving, 204-205
    - binary files, 201
    - exception detection, 203-204
    - features, 201
    - goals, 200
  - forking off/tracing child processes, 185-187
  - future improvements, 221
  - interesting UNIX signals, 187
  - methods, 171-172
    - brute force, 172-173
    - inputs, 174
    - intelligent brute force, 173-174
  - not interesting UNIX signals, 188
  - notSPIKEfile
    - features, 182
    - missing features, 182
    - programming language, 195
    - RealPix format string vulnerability, 193-195
    - shell script workarounds, 192
  - ODF, 54
  - Open XML, 54
  - SPIKEfile
    - features, 182
    - missing features, 182
    - programming language, 195
    - shell script workarounds, 192
  - targets, 170-171, 205, 208-209
    - Windows Explorer, 206-209
    - Windows Registry, 209
  - vulnerabilities
    - DoS, 175
    - examples, 170
    - format strings, 177
    - heap overflows, 177
    - integer handling, 175-177
    - logic errors, 177
    - race conditions, 178
    - simple stack, 177
  - Windows, 197-198
  - zombie processes, 189-191
  - File Transfer Protocol (FTP), 48, 362-363
  - FileFuzz, 39
    - applications, launching, 203
    - ASCII text files, 202
    - audits, saving, 204-205
    - benefits, 221
    - binary files, 201
    - case study, 217-220
    - development, 209
      - applications, launching, 211-213
      - approach, 209
      - design, 210-217
      - exception detection, 213-217
      - files, creating, 210
      - language, choosing, 210
      - source files, reading, 210-211
      - writing to files, 211
    - error detection, 67
    - exception detection, 203-204
    - features, 201
    - future improvements, 221
    - goals, 200
  - files. *See also* file format fuzzers
    - fuzzing, 24
    - permissions, 95
    - rule, 359-360
  - SWF, 372
    - bit\_field class, 378
    - component relationships, 383
-

- data generation, 384
- data structures, 374-377
- dependent\_bit\_field class, 379
- environment, 385
- header, 372-373
- MATRIX struct, 379-380
- methods, 385
- RECT/RGB structs, 378
- string primitives, 383
- SWF files, modeling, 374
- tags, 374
- tags, defining, 380-382
- FilesAttributes tag (Flash), 374
- filtering tracking recordings, 446
- find command, 93
- first-chance exceptions, 489-491
- fixed length fields (protocols), 47
- Flash, 115, 279
- flatten() routine, 378
- Flawfinder website, 7
- forking off child processes, 185-187
- format string vulnerabilities, 85, 106
  - example, 477
  - exploiting, 85
  - file formats, 177
  - RealPlayer RealPix format string, 193-195
- frameworks, 43-44
  - antiparser, 354-356
  - attack heuristics, 353
  - Autodafe[as], 369-371
  - automatic length calculation, 352
  - code reusability, 354
  - community involvement, 43
  - complexity, 44
  - CRC values, 353
  - data parsing, 353
  - development time, 44
  - Dfuz, 357-360
    - data sources, 358
    - functions, 357-358
    - lists, declaring, 358
    - protocols, emulating, 359
    - rule files, 359-360
    - variables, defining, 357
  - fault detection, 353
  - features, 352-353
  - fuzzing metrics, 353
  - GPF, 366-369
  - limitations, 43
  - overview, 352-354
  - PaiMei, 449
    - advanced target monitoring, 486-489
    - basic target monitoring, 482-484
    - components, 450
    - crash binning utility, 487-489
    - PIDA interface, 450
    - PStalker module. *See* PStalker
    - SWF fuzzing, 385
  - Peach, 364-366
    - code reusability, 366
    - disadvantages, 366
    - generators, 364
    - groups, 365
    - publishers, 364
    - transformers, 364
  - PI, 428
  - programming languages, 352
  - protocol modeling, 352
  - pseudo-random data, generating, 353
  - reusability, 43
  - SPIKE, 361-364
    - block-based protocol representation, 361-362
    - disadvantage, 363
    - FTP fuzzer, 362-363
  - Sulley, 386-387
    - blocks, 392-397
    - data types, 388-390
    - delimiters, 391
    - directory structure, 387-388
    - download website, 386
    - environment, setting up, 413
    - features, 386
    - fuzzers, launching, 414-416
    - integers, 390-391
    - legos, 398-399
    - postmortem phase, 405-409
    - requests, building, 410-411
    - RPC endpoint walkthrough. *See* Sulley framework, RPC endpoint walkthrough
    - session, 399-404, 411-413
    - strings, 391
  - FTP (File Transfer Protocol), 48, 362-363
  - functions. *See also* routines
    - BindAdapter(), 259
    - byref(), 318
    - CreateProcess(), 10, 203
    - create\_string\_buffer(), 318
    - Dfuz, 357-358
    - GetCurrentProcessId(), 318
    - getenv, 98
    - printf(), 248
    - ReadProcessMemory(), 318
    - ReceivePacket(), 259
    - s\_block\_end(), 242, 392
    - s\_block\_start(), 242, 392
    - taboo(), 475
    - WriteProcessMemory(), 319
  - func\_resolve() routine, 333
  - future of fuzzing
    - development environment integration, 516
    - hybrid analysis of vulnerabilities, 515
  - fuzz\_client.exe, 334-346

- fuzz\_server.exe, 334
  - conceptual diagram, 339
  - launching, 345
  - memory allocation, 344-345
  - OllyDbg, 336-338
  - restore point, 335
  - snapshot, 335, 343
  - WS2\_32.dll recv() breakpoint, 336
- fuzz\_trend\_server\_protect\_5168.py, 414-416
- fuzz values, choosing, 480-481
- fuzzer stepping, 473
- fuzzing
  - as black box testing, 12
  - defined, 21
  - history, 22-27
    - ActiveX, 25
    - Codenomicon, 23
    - files, 24
    - Miller, Professor Barton, 22
    - PROTOS test suites, 23
    - sharefuzz, 24
    - SPIKE, 23
  - limitations
    - access control, 29
    - backdoors, 30
    - design logic, 30
    - memory corruption, 31
    - multistage vulnerabilities, 32
  - phases, 27-29
- Fuzzing mailing list, 25
- fuzzing metrics, 353
- G**
- GAs (genetic algorithms), 431-435
  - CFG
    - connecting path, 433
    - exit nodes, 434
    - potential vulnerability, 433
  - fitness function, 432
  - reproduction function, 431
  - Sidewinder tool, 433-434
  - as stochastic global optimizers, 432
- GDB (GNU debugger), 16, 96-97
- GDI+ buffer overflow vulnerability, 198, 217-220
- General Purpose Fuzzer (GPF), 366-369
- generating data
  - CCR example, 78-79
  - data types
    - character translations, 85
    - command injection, 87
    - directory traversal, 86
    - field delimiters, 83-84
    - format strings, 85
    - integer values, 79-81
    - string repetitions, 82
  - generators, 364
    - pseudo-random data, 353
    - SWF fuzzing test case, 384
- generation-based fuzzers, 22, 231
- generators, 364
- generic line-based TCP fuzzer, 240-243
- generic script-based fuzzers, 244
- genetic algorithms. *See* GAs
- GET method, 123
- GetCurrentProcessId() function, 318
- getenv function, 98
- GetFuncDesc() routine, 296
- GetNames() routine, 295
- getopt module, 104, 107
- getPayload() routine, 356
- GetThreadContext() API, 327
- GetTypeInfoCount() routine, 294
- GetURL() method, 299
- Gizmo Project case study
  - data sources, 462
  - execution, monitoring, 464-465
  - features list, 457
  - results, 465
  - SIP
    - packets, testing, 458-461
    - processing code, 462
  - strategy, 458-461
  - tags, choosing, 462
  - tracking capture options, 462-463
- GNU debugger (GDB), 16, 96-97
- Gosling, James, 116
- GPF (General Purpose Fuzzer), 366-369
- graphs
  - call graphs, 439
  - CFGs, 440-441
    - disassembled dead listing, 441
    - GAs, 433-434
  - example SMTP session, 399-401
  - Sulley crash bin graphical representation, 406
- gray box testing, 14
  - binary auditing
    - automated, 17-18
    - manual, 14-16
  - pros/cons, 18
- Greene, Adam, 38
- groups
  - blocks, 392-393
  - Peach framework, 365
- H**
- Hamachi, 24, 42
- handler\_bp() routine, 333
- hardware breakpoints, 324
- HEAD method, 124
- headers
  - Accept, 122
  - Accept-Encoding, 122

- Accept-Language, 122
  - Connection, 123
  - Cookie, 123
  - Host, 122
  - HTML, 271
  - HTTP protocol, 122
  - SWF, 372-373
  - User-Agent, 122
  - web applications input, 128
  - heap overflow vulnerabilities, 129
    - browsers, 277-279
    - file formats, 177
  - Heller, Thomas, 318
  - heuristics, 79
    - ActiveX, 298
    - attack, 353
    - character translations, 85
    - command injection, 87
    - directory traversal, 86
    - field delimiters, 83-84
    - format strings, 85
    - integer values, 79-81
    - protocol dissection, 421
      - disassembly-level, 426-427
      - hierarchical breakdown, 424
      - improved analysis, 425-426
      - proxy fuzzing, 422-423
    - string repetition, 82
  - Hewlett-Packard Mercury LoadRunner vulnerability, 263
  - hexadecimal encoding/decoding, 262
  - hierarchy
    - protocols, 424
    - Sulley directory structure, 387-388
  - history
    - COM, 284
    - fuzzing, 22-27
      - ActiveX, 25
      - Codenomicon, 23
      - files, 24
      - Miller, Professor Barton, 22
      - PROTOS test suites, 23
      - sharefuzz, 24
      - SPIKE, 23
    - Samba, 420
  - Hoglund, Greg, 312
  - Holodeck, 514-515
  - hook points, 331
  - hooking processes, 324-327
    - context modifications, 327
    - EIP, adjusting, 326
    - INT3 opcode, writing, 324
    - original bytes at breakpoint address, saving, 324
    - software breakpoint, catching, 325
  - Host header, 122
  - HTML (Hypertext Markup Language)
    - header vulnerabilities, 271
    - status codes, 146
    - tag vulnerabilities, 271-273
  - HTTP (Hypertext Transfer Protocol)
    - field delimiters, 83
    - methods, 133
    - protocol, 122
    - requests, identifying, 144
    - response splitting, 134
    - status codes, 135
  - human computation power, 73
  - hybrid analysis of vulnerabilities, 515
  - Hypertext Preprocessor (PHP), 115
- ## I
- IAcroXDocShim interface, 295
  - IBM
    - AIX 5.3 local vulnerabilities, 110
    - Rational Purify, 492
  - ICMP, dissecting, 429-430
  - IDA (Interactive Disassembler), 15
  - IDA Pro (DataRescue Interactive Disassembler), 439
  - IDE (integrated development environment), 503
  - IETF (Internet Engineering Task Force), 54
  - iFuzz, 38
    - case study, 110-111
    - development approach, 105-107
    - features, 103-105
    - fork, execute, and wait approach, 107
    - fork, ptrace/execute, and wait/ptrace approach, 108-109
    - getopt hooker, 105
    - language, 109
    - modules, 104
      - argv, 104-106
      - getopt, 104, 107
      - preloadable getenv fuzzer, 105
      - single-option/multi-option, 104, 107
    - postdevelopment observations, 111-112
  - IIDs (interface IDs), 285
  - improperly supported HTTP methods, 133
  - in-memory fuzzers, 42-43
    - advantages, 42, 302
    - complexity, 43
    - control flow diagram example, 306
    - example
      - access violation handler, 340-342
      - breakpoint handler, 342
      - client, launching, 334
      - conceptual diagram, 339
      - data mutation, 343
      - fuzz\_client, launching, 346
      - fuzz\_server memory allocation, 344-345
      - fuzz\_server snapshot, 343
      - fuzz\_server, launching, 345
    - false positives, 43
    - fault detection, 311-312

- hook points, choosing, 331
- language, choosing, 317-319
- memory mutation locations, choosing, 331
- methods
  - mutation loop insertion, 308-309, 316
  - snapshot restoration mutation, 309-310, 316
- overview, 302-307
- process depth, 310-311
- process hooking, 324-327
  - context modifications, 327
  - EIP, adjusting, 326
  - INT3 opcode, writing, 324
  - original bytes at breakpoint address, saving, 324
  - software breakpoint, catching, 325
- process snapshots/restores, 327-331
  - memory block contents, saving, 329-330
  - thread context, saving, 328-329
- reproduction, 43
- required feature sets, 316-317
- restore point, 335
- speed, testing, 310-311
- shortcuts, 42
- snapshot point, 335
- speed, 42
- targets, 307-308
- WS2\_32.dll recv() breakpoint, 336
  - implementing with PyDbg, 340-345
  - obfuscation method, 335
  - OllyDbg, 336-338
  - server data capture, 335
  - server, launching, 334
- inputs
  - file format fuzzing, 174
  - identifying, 27
  - web applications
    - choosing, 119-121
    - cookies, 129
    - headers, 128
    - identifying, 130-131
    - method, 123-125
    - post data, 130
    - protocol, 128
    - request-URI, 126-128
  - web browser fuzzing
    - ActiveX controls, 273-275
    - client-side scripting, 276
    - CSS, 275-276
    - Flash, 279
    - HTML headers, 271
    - HTML tags, 271-273
    - URLs, 280
    - XML tags, 273
- Inspector, 18, 312
- instructions, tracing, 444-445. *See also* tracking
- INT3 opcode, writing, 324
- integers
  - handling vulnerabilities, 175-177
  - Sulley framework, 390-391
  - values, 79-81
- integrated development environment (IDE), 503
- intelligence
  - black box testing, 14
  - brute force fuzzing
    - file formats, 173-174
    - network protocols, 231
  - fault detection
    - first-chance versus last-chance exceptions, 489-491
    - page faults, 474
    - software memory corruption vulnerability categories, 475-479
  - load monitoring, 182
- Interactive Disassembler (IDA), 15
- interface IDs (IIDs), 285
- interfaces
  - COM, 284
  - IAcroXDocShim, 295
  - IObjectSafety, 292
  - PIDA, 450
  - Sulley Web monitoring, 404
- Internet protocols, 46
- Internet Engineering Task Force (IETF), 54
- IObjectSafety interface, 292
- Ipswitch
  - I-Mail
    - vulnerability, 420
    - Web Calendaring directory traversal, 157
  - Whatsup Professional SQL injection attack, 118, 160-162
- ITS4 download website, 7
- J**
- Java, 116
- JavaScript, 116
- Jlint website, 7
- K**
- kill bitting, 292
- L**
- languages (programming)
  - choosing, 77
  - ECMAScript, 276
  - FileFuzz, 210
  - frameworks, 352
  - iFuzz, 109
  - in-memory fuzzing, 317-319
  - ProtoFuzz, 256
  - Python. *See* Python
  - SPIKEfile/notSPIKEfile tools, 195
  - WebFuzz, 148
- last-chance exceptions, 489-491

- Layer 2 vulnerabilities, 228
  - Layer 3 vulnerabilities, 229
  - Layer 4 vulnerabilities, 229
  - Layer 5 vulnerabilities, 229
  - Layer 6 vulnerabilities, 230
  - Layer 7 vulnerabilities, 230
  - legos, 398-399
  - libdasm library, 75
  - libdisasm library, 75
  - Libnet library, 76
  - LibPCAP library, 76
  - libraries
    - data types, including
      - character translations, 85
      - command injection, 87
      - directory traversal, 86
      - field delimiters, 83-84
      - format strings, 85
      - integer values, 79-81
      - string repetitions, 82
    - libdasm, 75
    - libdisasm, 75
    - Libnet, 76
    - LibPCAP, 76
    - Metro Packet Library, 76
    - packet capture, 256-257
    - PaiMei, 299
    - preloading, 98-99
    - PyDbg, 445-446
    - SIPPhoneAPI, 462
    - vulnerable web applications, 134
    - WinPcap, 256
  - limitations
    - frameworks, 43
    - fuzzing
      - access control, 29
      - backdoors, 30
      - design logic, 30
      - memory corruption, 31
      - multistage vulnerabilities, 32
      - resource constraints, 69
    - PStalker, 454
  - LoadTypeLib() routine, 294
  - local fuzzers, 37
    - abort signals, 100
    - command-line, 37-38, 89
    - environment variable, 38-39, 90-92
      - GDB method, 96-97
      - getenv function, 98
      - library preloading, 98-99
    - file format, 39
    - iFuzz
      - case study, 110-111
      - development approach, 105-107
      - features, 103-105
      - fork, execute, and wait approach, 107
      - fork, ptrace/execute, and wait/ptrace approach, 108-109
      - getopt hooker, 105
      - language, 109
      - modules, 104-106
      - postdevelopment observations, 111-112
      - preloadable getenv fuzzer, 105
      - methods, 95
      - principles, 92
      - problems, detecting, 99-101
      - ptrace method, 100
      - su application example, 92
      - targets, 93-95
  - log files
    - analysis vulnerabilities, 118
    - web application fuzzing, 136
  - logic error vulnerabilities, 177
  - LogiScan, 17
  - loops, debug event, 322-323
- ## M
- Macbook vulnerability, 228
  - Macromedia Flash, 115, 279
  - Macromedia Shockwave Flash. *See* SWF
  - malformed arguments, 37
  - mangleme, 24, 42
  - manual testing
    - applications, 10
    - protocol mutation, 35
    - RCE, 14-16
  - Matasano's Protocol Debugger, 423
  - MATRIX struct, 379-380
  - media server vulnerability, 226
  - memory
    - allocating, 344-345, 492
    - block contents, saving, 329-330
    - corruption, 31
    - in-memory fuzzing
      - benefits, 302
      - control flow diagram example, 306
      - example. *See* in-memory fuzzers, example
      - fault detection, 311-312
      - hook points, choosing, 331
      - language, choosing, 317-319
      - memory mutation locations, choosing, 331
      - mutation loop insertion, 308-309, 316
      - overview, 302, 306-307
      - process depth, 310-311
      - process hooking, 324-327
      - process snapshots/restores, 327-331
      - required feature sets, 316-317
      - snapshot restoration mutation, 309-310, 316
      - speed, testing, 310-311
      - targets, 307-308
    - mutable blocks, creating, 318
    - mutation locations, choosing, 331

- process memory reading/writing, 318-319
  - protection attributes, 303
  - software memory corruption vulnerabilities, 475
    - execution control, transferring, 475-476
    - reading addresses, 476-478
    - writing to addresses, 478-479
  - Windows memory model, 302-306
  - metadata test case, 179
  - method input (web applications), 123-125
  - methods. *See also* functions; routines
    - ActiveX, 293-297
    - automatic protocol generation testing, 36
    - BeginRead(), 152
    - BeginWrite(), 151
    - brute force testing, 36
    - btnRequest\_Click(), 153
    - CONNECT, 125
    - CreateFile(), 299
    - CreateProcess(), 299
    - DELETE, 124
    - DownloadFile(), 299
    - Execute(), 299
    - file format fuzzing, 171-174
      - brute force, 172-173
      - inputs, 174
      - intelligent brute force, 173-174
    - GET, 123
    - GetURL(), 299
    - HEAD, 124
    - HTTP, 133
    - in-memory fuzzing
      - mutation loop insertion, 308-309, 316
      - process depth, 310-311
      - snapshot restoration mutation, 309-310, 316
    - local fuzzing, 95
    - manual protocol mutation testing, 35
    - network protocol fuzzing, 230
      - brute force, 231
      - generation-based, 231
      - intelligent brute force, 231
      - modified client mutation, 232
      - mutation-based, 231
    - OnReadComplete(), 152
    - OPTIONS, 125
    - POST, 124
    - pregenerated test cases, 34
    - ptrace, 100
    - PUT, 124
    - random, 34
    - SWF fuzzing, 385
    - TRACE, 124
    - web browser fuzzing, 269
      - approaches, 269-271
      - inputs, 275-280
  - Metro Packet Library, 76
  - Microsoft
    - COM. *See* COM
    - fuzzing, 13
    - NDIS protocol driver, 255
    - Open XML format, 54
    - Remote Procedure Call (MSRPC), 40
    - Samba, 420
    - security, 498-499
    - source code leak, 5
    - vulnerabilities
      - Excel eBay vulnerability, 199
      - GDI+ buffer overflow, 198, 217-220
      - Outlook Express NNTP vulnerability discovery, 447-449
      - Outlook Web Access Cross-Site Scripting, 117
      - PNG, 199
      - WMF, 199
    - Windows
      - Live/Office Live, 114
      - memory model, 302-306
      - worms, 224-226
  - Miller, Professor Barton, 22
  - MLI (mutation loop insertion), 308-309, 316
  - MMalloc() routine, 81
  - MoBB (Month of Browser Bugs), 268
  - modified client mutation fuzzing, 232
  - modules
    - ctypes, 318
    - iFuzz, 104-107
  - monitoring
    - ActiveX fuzzer, 299
    - automated debugger, 481
      - advanced example, 486-489
      - architecture, 481
      - basic example, 482-484
      - PaiMei crash binning utility, 487-489
    - exceptions, 28
    - instruction execution. *See* tracking
    - network vulnerabilities, 118
  - Month of Browser Bugs (MoBB), 268
  - Moore, H.D., 24-25, 42
  - MSRPC (Microsoft Remote Procedure Call), 40
  - Mu Security, 25
  - Mu-4000, 512
  - multi-option module (iFuzz), 104, 107
  - Multiple Vendor Cacti Remote File Inclusion Vulnerability website, 118
  - multistage vulnerabilities, 32
  - Murphy, Matt, 24
  - mutable memory blocks, creating, 318
  - mutation loop insertion (MLI), 309, 308-309, 316
  - mutation-based fuzzers, 22, 231
- ## N
- name-value pairs, 57
  - naming schemes, 439
  - NDIS (Network Driver Interface Specification) protocol driver, 255

Needleman-Wunsch algorithm, 428  
 Netcat, 49  
 NetMail Networked Messaging Application Protocol. *See* NMAP  
 Network Driver Interface Specification (NDIS) protocol driver, 255  
 Network News Transfer Protocol (NNTP), 447  
 networks  
   adapters, 257-258  
   client-side vulnerabilities, 223  
   layer vulnerabilities, 229  
   monitoring, 118, 402  
   protocol fuzzers, 40  
     complex protocols, 40  
     defined, 224  
     fault detection, 232-233, 254  
     methods, 230-232  
     protocol drivers, 255  
     ProtoFuzz. *See* ProtoFuzz  
     requirements, 250-253  
     simple protocols, 40  
   server-side vulnerabilities, 223  
   socket communication component, 224  
 targets, 226-230  
   application layer, 230  
   categories, 226  
   data link layer, 228  
   network layer, 229  
   presentation layer, 230  
   session layer, 229  
   transport layer, 229  
   vulnerabilities, 226  
 UNIX systems, 236  
   SPIKE NMAP fuzzer script, 244-248  
   targets, 236-237  
 Windows. *See* ProtoFuzz  
 NMAP (NetMail Networked Messaging Application Protocol), 236  
   overview, 237-240  
   SPIKE NMAP fuzzer script, 245-248  
 NNTP (Network News Transfer Protocol), 447  
 nonalphanumeric characters, 83-84  
 nonexecutable (NX) page permissions, 476  
 notSPIKEfile, 39  
   core fuzzing engine, 184-185  
   exception handling, 183-184  
   features, 182  
   forking off/tracing child processes, 185-187  
   interesting UNIX signals, 187  
   missing features, 182  
   not interesting UNIX signals, 188  
   programming language, 195  
   RealPix format string vulnerability, 193-195  
   shell script workarounds, 192  
   zombie processes, 189-191  
 Novell NetMail IMAPD Command Continuation Request Heap Overflow security advisory, 81

NW (Needleman-Wunsch) algorithm, 428  
 NX (nonexecutable) page permissions, 476

**O**

OASIS (Organization for the Advancement of Structured Information Standards), 54  
 ODF (OpenDocument format), 54  
 Office Live, 114  
 OLE (Object Linking and Embedding), 284  
 OllyDbg, 16, 335  
   before/after demangle, 336  
   conceptual diagram, 339  
   parse routine, 338  
   restore point, 338  
   WS2\_32.dll recv() breakpoint, 336  
 OnReadComplete() method, 152  
 Open Document format (ODF), 54  
 open protocols, 54  
 open source disassemble libraries, 75  
 Open System for Communication in Realtime (OSCAR), 49  
 open XML format, 54  
 OpenSSH Remote Challenge vulnerability, 226  
 operating system logs, fault detection, 233  
 OPTIONS method, 125  
 Organization for the Advancement of Structured Information Standards (OASIS), 54  
 OSCAR (Open System for Communication in Realtime), 49  
 Outlook Express NNTP vulnerability discovery, 447-449  
 Overflow fuzz variable, 142  
 overflows (stack), 246  
 OWASP (WebScarab), 41

**P**

packets  
   assembling, 250-251  
   capture libraries, choosing, 256-257  
 page faults, 474  
 PAGE\_EXECUTE attribute, 303  
 PAGE\_EXECUTE\_READ attribute, 303  
 PAGE\_EXECUTE\_READWRITE attribute, 303  
 PAGE\_NOACCESS attribute, 304  
 PAGE\_READONLY attribute, 304  
 PAGE\_READWRITE attribute, 304  
 PaiMei framework  
   ActiveX fuzzer, 299  
   crash binning utility, 487-489  
   SWF fuzzing, 385  
   target monitoring  
     advanced example, 486-489  
     basic example, 482-484  
 PAIMEfilefuzz, 39  
 PAIMEIpstalker. *See* PStalker  
 PAM (Percent Accepted Mutation), 429  
 parameters (ActiveX), 293-297  
 parse() routine, 306

- parsing
    - data
      - networks, 251-252
      - ProtoFuzz design, 259-261
    - framework features, 353
  - passing data types, 318
  - Pattern Fuzz (PF), 368
  - PDB (Protocol Debugger), 423
  - PDML2AD, 371
  - Peach framework, 40, 364-366
    - code reusability, 366
    - disadvantages, 366
    - generators, 364
    - groups, 365
    - publishers, 364
    - transformers, 364
  - Percent Accepted Mutation (PAM), 429
  - performance
    - degradation, 147
    - monitors, Web browser errors, 282
  - PF (Pattern Fuzz), 368
  - phases of fuzzing, 29
    - exceptions, monitoring, 28
    - exploitability, 28
    - fuzz data, executing/generating, 28
    - inputs, identifying, 27
    - targets, identifying, 27
  - phishing vulnerabilities, 281
  - PHP (Hypertext Preprocessor), 115
  - phpBB Group phpBB Arbitrary File Disclosure
    - Vulnerability website, 117
  - PI (Protocol Informatics), 428
  - PIDA interface, 450
  - Pierce, Cody, 39
  - Pin DBI system, 492
  - pinging, error detection, 67
  - plain text protocols, 48-49
  - PNG (Portable Network Graphics) vulnerability, 199
  - post data, web applications input, 130
  - POST method, 124
  - postmortem phase (Sulley), 405-409
    - crash bin graphical exploration, 406
    - crash locations, viewing, 405-406
  - precompile security solution, 480
  - pregenerated test cases, 34
  - preloading libraries, 98-99
  - presentation layer vulnerabilities, 230
  - primitive fault detection, 472-474
  - principles of local fuzzing, 92
  - printf() function, 248
  - Process Stalker. *See* PStalker
  - processes
    - child, forking off and tracing, 185-187
    - depth, 64-66
    - hooking, 324-327
      - context modifications, 327
      - EIP, adjusting, 326
      - INT3 opcode, writing, 324
      - original bytes at breakpoint address, saving, 324
      - points, choosing, 331
      - software breakpoint, catching, 325
  - memory
    - reading, 318
    - writing, 319
  - monitor agent, 403
  - snapshots/restores
    - handling, 327-331
    - memory block contents, saving, 329-330
    - thread context, saving, 328-329
  - state, 64-66
  - zombie, 189-191
- process\_restore() routine, 345
- process\_snapshot() routine, 343
- profiling targets, 443-444
- ProgIDs (program IDs), 285
- programming languages
  - choosing, 77
  - ECMAScript, 276
  - FileFuzz, 210
  - frameworks, 352
  - iFuzz, 109
  - in-memory fuzzing, 317-319
  - ProtoFuzz, 256
  - Python. *See* Python
  - SPIKEfile/notSPIKEfile tools, 195
  - WebFuzz, 148
- properties (ActiveX), 293-297
- proprietary protocols, 54, 421
- Protocol Debugger (PDB), 423
- Protocol Informatics (PI), 428
- protocol-specific fuzz scripts, 244
- protocol-specific fuzzers (SPIKE), 243
- protocols
  - AIM, 49-52
    - logon credentials sent, 52
    - server reply, 51
    - username, 51
  - automated dissection
    - bioinformatics, 427-431
    - genetic algorithms, 431-435
    - heuristic-based, 421-427
  - binary, 49-52
  - block-based representation, 361-362
  - complex, 40
  - defined, 46
  - documentation, 421
  - drivers, 255
  - elements
    - block identifiers, 58
    - block sizes, 58
    - checksums, 58
    - name-value pairs, 57

- emulating with Dfuz, 359
  - fields, 46-48
  - FTP, 48
  - fuzzing, 419-421
  - hierarchical breakdown, 244
  - HTTP, 122
  - ICMP, 429-430
  - Internet, 46
  - modeling, 352
  - network fuzzers, 40, 53-54
    - application layer vulnerabilities, 230
    - complex protocols, 40
    - data capture, 251
    - data link layer vulnerabilities, 228
    - defined, 224
    - fault detection, 232-233, 254
    - fuzz variables, 253
    - methods, 230-231
    - modified client mutation, 232
    - network layer vulnerabilities, 229
    - packet assembling, 250-251
    - parsing data, 251-252
    - presentation layer vulnerabilities, 230
    - protocol drivers, 255
    - ProtoFuzz. *See* ProtoFuzz
    - sending data, 253
    - session layer vulnerabilities, 229
    - simple protocols, 40
    - socket communication component, 224
    - targets, 226-230
    - transport layer vulnerabilities, 229
    - UNIX systems, 236-237, 244-247
    - Windows systems. *See* ProtoFuzz
  - NMAP, 236
    - overview, 237-240
    - SPIKE NMAP fuzzer script, 244-247
  - NNTP, 447
  - open, 54
  - overview, 45
  - plain text, 48-49
  - proprietary, 54, 421
  - simple, 40
  - SIP
    - processing code, 462
    - testing, 458-461
  - testing
    - automatic protocol generation testing, 36
    - manual protocol mutation testing, 35
  - third-party, 421
  - TLV style syntax, 352
  - web applications input, 128
- ProtoFuzz
- case study, 262-264
  - data capture, 251
  - design, 257, 262
    - data capture, 258-259
    - fuzz variables, 261
    - hexadecimal encoding/decoding, 262
    - network adapters, 257-258
    - parsing data, 259-261
  - disadvantages, 264-265
  - fuzz variables, 253
  - goals, 255
  - NDIS protocol driver, 255
  - packets
    - assembling, 250-251
    - capture library, choosing, 256-257
    - parsing data, 251-252
    - programming language, 256
    - sending data, 253
  - PROTOS, 23, 458-460, 510
  - ProtoVer Professional, 512
  - proxy fuzzing, 422-423
  - ProxyFuzzer, 422-424
  - pseudo-random data generation, 353
  - PStalker
    - data
      - capture, 454
      - exploration, 453
      - sources, 452-453
      - storage, 454-457
    - Gizmo Project case study
      - data sources, 462
      - execution, monitoring, 464-465
      - features list, 457
      - results, 465
      - SIP packets, testing, 458-461
      - SIP processing code, 462
      - strategy, 458-461
      - tags, choosing, 462
      - tracking capture options, 462-463
    - layout overview, 451-452
    - limitations, 454
  - ptrace method, 76, 100
  - PTRACE\_TRACEMENT request, 187
  - PureFuzz, 367
  - PUT method, 124
  - PyDbg class, 332-334
    - in-memory fuzzer implementation, 340-345
    - single stepper tracking tool implementation, 445-446
- Python
- ctypes module, 318
  - extensions, 77
  - interfacing with COM, 287
  - PaiMei framework, 449
    - advanced target monitoring, 486-489
    - basic target monitoring, 482-484
    - components, 450
    - crash binning utility, 487-489
    - PIDA interface, 450
    - PStalker module. *See* PStalker
    - SWF fuzzing, 385
  - Protocol Informatics (PI), 428

- PyDbg class, 332, 334, 332
  - in-memory fuzzer implementation, 340-345
  - single stepper tracking tool implementation, 445-446
- PythonWin COM browser, 288
- PythonWin type library browser, 288
- Q**
- QA Researchers, 504
- R**
- race condition vulnerabilities, 178
- Raff, Aviv, 24, 42
- random fuzzing, 367
- random primitives, 389-390
- random testing, 34
- randomize() routine, 378
- Rational Purify, 492
- RATS (Rough Auditing Tool for Security), 7-8
- RCE (reverse code engineering), binary auditing
  - automated, 17-18
  - manual, 14-16
- reading
  - addresses, 476-478
  - process memory, 318
- ReadProcessMemory() function, 318
- RealPlayer
  - RealPix format string vulnerability, 193-195
  - shell script workaround, 192
- RealServer ../ DESCRIBE vulnerability, 226
- ReceivePacket() function, 259
- recordings (tracking), filtering, 446
- record\_crash() routine, 483, 487
- RECT struct, 378
- Reduced Instruction Set Computer (RISC)
  - architecture, 438
- registering callbacks, 401-402
- remote access services vulnerability, 226
- remote code injection, 134
- remote fuzzers, 39
  - network protocol, 40
  - web application, 41
  - web browser, 41-42
- repeaters as block helpers, 396
- reporting exceptions, 183-184
- reproducibility, 62
  - black box testing, 13
  - value of automation, 74
- reproduction, in-memory fuzzing, 43
- request-URI input (web applications), 126-128
- requests
  - HTTP, 144
  - PTRACE\_TRACEME, 187
  - Sulley, 410-411
  - timeouts, 147
  - Web application fuzzing, 140-141
  - WebFuzz, 153-155
- Requests for Comment (RFCs), 54
- researches
  - QA, 504
  - security, 504-505
- resource constraints, 69
- responses
  - error messages, 146-147
  - user input, 147
  - Web application fuzzing, 143
  - WebFuzz, 155-156
- restores
  - points
    - fuzz\_server.exe, 335
    - OllyDbg, 338
  - processes
    - handling, 327-331
    - memory block contents, saving, 329-330
    - thread context, saving, 328-329
- return codes, 179
- reusability, 43, 62-64
- reverse code engineering (RCE), 14
- reverse engineering frameworks (PaiMei), 449-450.
  - See also* PStalker
- RFCs (Requests for Comment), 54
- RGB struct, 378
- RISC (Reduced Instruction Set Computer)
  - architecture, 438
- Rough Auditing Tool for Security (RATS), 8
- routines. *See also* functions; methods
  - ap.getPayload(), 356
  - av\_handler(), 483
  - bp\_set(), 333
  - ContinueDebugEvent(), 323
  - DebugActiveProcess(), 321
  - DebugSetProcessKillOnExit(), 321
  - flatten(), 378
  - func\_resolve(), 333
  - GetFuncDesc(), 296
  - GetNames(), 295
  - GetThreadContext(), 327
  - GetTypeInfoCount(), 294
  - handler\_bp(), 333
  - LoadTypeLib(), 294
  - MMalloc(), 81
  - parse(), 306
  - process\_restore(), 345
  - process\_snapshot(), 343
  - randomize(), 378
  - record\_crash(), 483, 487
  - s\_checksum(), 396
  - self.push(), 398
  - setMaxSize(), 356
  - setMode(), 356
  - SetThreadContext(), 327
  - set\_callback(), 333
  - smart(), 378

- s\_repeat(), 396
- sscanf(), 427
- s\_sizer(), 395
- strcpy(), 5
- syslog(), 478
- Thread32First(), 328
- to\_binary(), 378
- to\_decimal, 378
- unmarshal(), 306
- VirtualQueryEx(), 329
- write\_process\_memory(), 344
- xmlComposeString(), 407
- Royce, Winston. *See* waterfall model
- RPC DCOM Buffer Overflow vulnerability, 226
- RPC-based services vulnerabilities, 226
- rule files (Dfuz), 359-360
- S**
- Samba, 420
- SAP Web Application Server sap-exiturl Header HTTP Response Splitting website, 117
- saving
  - audits, 204-205
  - memory block contents, 329-330
  - original bytes at breakpoint address, 324
  - test case metadata, 179
  - thread context, 328-329
- s\_block\_end() function, 242, 392
- s\_block\_start() function, 242, 392
- s\_checksum() routine, 396
- scripts
  - client-side scripting vulnerabilities, 276
  - protocol-specific fuzz, 244
  - SPIKE NMAP fuzzer, 244-248
  - XSS scripting case study, 163-166
- SDL (Security Development Lifecycle), 498
- SDLC (software development lifecycle), 69
  - Microsoft SDL, 498
  - security, 69
  - waterfall model, 499
    - analysis, 500
    - coding, 501-502
    - design, 500-501
    - maintenance, 502-503
    - testing, 502
- secure shell (SSH) servers, 64
- security
  - development lifecycle, 498
  - Microsoft, 499
  - researchers, 504-505
  - software development lifecycle. *See* SDLC
  - zone vulnerabilities, 281
- SecurityReview, 18
- SEH (Structured Exception Handler), 484
- self.push() routine, 398
- sequences, aligning, 427
- servers
  - data capture example, 335
  - launching, 334
  - media vulnerability, 226
  - Microsoft worms vulnerabilities, 224-226
- sessions
  - layer vulnerabilities, 229
  - Sulley framework, 399
    - callbacks, registering, 401-402
    - creating, 411-413
    - instantiating, 401
    - linking requests into graphs, 399-401
    - targets and agents, 402-404
    - Web monitoring interface, 404
- setgid bits, 93
- setMaxSize() routine, 356
- setMode() routine, 356
- SetThreadContext() API, 327
- setuid applications, 37
- setuid bits, 93
- set\_callback() routine, 333
- SGML (Standardized General Markup Language), 273
- Sharefuzz, 24, 38
- Shockwave Flash. *See* SWF
- Sidewinder (GAs), 433-434
- SIGABRT signal, 188
- SIGALRM signal, 188
- SIGBUS signal, 188
- SIGCHLD signal, 188
- SIGFPE signal, 188
- SIGILL signal, 188
- SIGKILL signal, 188
- signals
  - handlers, 31
    - UNIX, 187-188
- SIGSEGV signal, 31, 188
- SIGSYS signal, 188
- SIGTERM signal, 188
- simple protocols, 40
- simple stack vulnerabilities, 177
- Simple Web Server buffer overflow, 159
- single-option modules (iFuzz), 104, 107
- SIP
  - processing code, 462
  - testing, 458-461
- SIPPhoneAPI library, 462
- sizers as block helpers, 395-396
- Slammer worm, 225
- smart data sets, 81
- smart() routine, 378
- Smith-Waterman (SW) local sequence alignment algorithm, 428
- snapshots
  - fuzz\_server.exe, 343
  - points, 335

- processes
    - handling, 328-331
    - memory block contents, saving, 329-330
    - thread context, saving, 328-329
  - restoration mutation (SRM), 309-310, 316
  - socket communication, 224
  - software
    - breakpoints, 324
    - development lifecycles. *See* SDLC
    - memory corruption vulnerabilities, 475
      - execution control, transferring, 475-476
      - reading addresses, 476-478
      - writing to addresses, 478-479
  - source code
    - browsers, 7
    - white box testing analysis, 4-9
  - SPI Dynamics, 41
  - SPI Dynamics Free Bank application vulnerability, 163-164
  - SPI Fuzzer, 41, 138
  - SPIKE, 23, 40, 361-364
    - block-based protocol
      - modeling, 242
      - representation, 361-362
    - disadvantage, 363
    - FTP fuzzer, 362-363
    - fuzz engine, 240
    - generic line-based TCP fuzzer, 240-243
    - generic script-based fuzzer, 244
    - protocol-specific fuzzers, 243-244
    - Proxy, 138
    - UNIX fuzzing, 236
      - SPIKE NMAP fuzzer script, 244-248
      - targets, 236-237
  - SPIKEfile
    - core fuzzing engine, 184-185
    - exceptions
      - detection engine, 183
      - reporting, 183-184
    - features, 182
    - forking off/tracing child processes, 185-187
    - interesting UNIX signals, 187
    - missing features, 182
    - not interesting UNIX signals, 188
    - programming language, 195
    - shell script workarounds, 192
    - zombie processes, 189-191
  - Splint website, 7
  - SQL injections
    - case study (Web applications), 160-162
    - vulnerabilities, 132
  - s\_repeat() routine, 396
  - SRM (snapshot restoration mutation), 309-310, 316
  - sscanf() routine, 427
  - SSH (secure shell) servers, 64
  - s\_size() routine, 395
  - stacks
    - fault detection example, 477
    - overflow vulnerabilities
      - Hewlett-Packard Mercury LoadRunner, 263
      - NMAP protocol, 246
    - unwinding, 485
  - Standardized General Markup Language (SGML), 273
  - start points (basic blocks), 440
  - state (process), 64-66
  - static primitives, 389-390
  - stepping, 473
  - stop points (basic blocks), 440
  - storage (data), 455-457
  - strcpy() routine, 5
  - strings
    - format, 85
      - file format vulnerabilities, 177
      - RealPlayer RealPix format string vulnerability, 193-195
      - vulnerabilities, 85, 477
    - primitives, 383
    - repetitions, 82
    - Sulley framework, 391
  - Structured Exception Handler (SEH), 484
  - su application example, 92
  - Sulley framework, 386-387
    - block helpers, 395
      - checksums, 396
      - example, 397
      - repeaters, 396
      - sizers, 395-396
    - blocks, 392
      - dependencies, 394-395
      - encoders, 394
      - grouping, 392-393
  - data types, 388-390
  - delimiters, 391
  - directory structure, 387-388
  - download website, 386
  - environment, setting up, 413
  - features, 386
  - fuzzers, launching, 414-416
  - integers, 390-391
  - legos, 398-399
  - postmortem phase, 405-409
  - requests, building, 410-411
  - RPC endpoint walkthrough, 409-410
    - environment, setting up, 413
    - launching, 414-416
    - requests, building, 410-411
    - sessions, creating, 411-413
- sessions, 399
  - callbacks, registering, 401-402
  - creating, 411-413
  - instantiating, 401
  - linking requests into graphs, 399-401

- targets and agents, 402-404
  - Web monitoring interface, 404
- strings, 391
- SuperGPF, 368
- Sutton, Michael, 39
- SW (Smith-Waterman) local sequence alignment algorithm, 428
- sweeping applications, 10
- SWF (Shockwave Flash), 372
  - bit\_field class, 378
  - component relationships, 383
  - data
    - generation, 384
    - structures, 374-377
  - dependent\_bit\_field class, 379
  - environment, 385
  - header, 372-373
  - MATRIX struct, 379-380
  - methods, 385
  - RECT/RGB structs, 378
  - string primitives, 383
  - SWF files, modeling, 374
  - tags, 374, 380-382
- syslog() routine, 478
- T**
- taboo() function, 475
- tag vulnerabilities
  - HTML, 271-273
  - XML, 273
- targets
  - audiences for fuzzing, 503-504
  - debugger-assisted monitoring, 481
    - advanced example, 486-489
    - architecture, 481
    - basic example, 482-484
    - PaiMei crash binning utility, 487-489
  - file format fuzzing, 170-171
  - identifying, 27
  - in-memory fuzzing, 307-308
  - local fuzzing, 93-95
  - network protocol fuzzing, 226-230
    - application layer, 230
    - categories, 226
    - data link layer, 228
    - network layer, 229
    - presentation layer, 230
    - session layer, 229
    - transport layer, 229
  - profiling, 443-444
  - setuid applications, 37
  - Sulley sessions, 402-404
  - UNIX, 236-237
  - web application fuzzing
    - environment, configuring, 118-119
    - examples, 117-118
    - inputs. *See* web application fuzzing, inputs
    - web browsers, 269
    - Windows file formats
      - identifying, 205-209
      - Windows Explorer, 206-209
      - Windows Registry, 209
- TCP/IP vulnerabilities, 229
- TcpClient class, 148-149
- TEBs (thread environment blocks), 485
- test cases
  - connectivity checks, 472
  - metadata, saving, 179
- testing
  - black box, 9
    - fuzzing, 12
    - manual, 10
    - pros/cons, 13-14
  - gray box, 14
    - binary auditing, 14-18
    - pros/cons, 18
  - SDL, 502
  - white box
    - pros/cons, 9
    - source code analysis, 4-5
    - tools, 6-8
- third-party protocols, 421
- threads
  - context, saving, 328-329
  - environment blocks (TEBs), 485
- Thread32First() API, 328
- Tikiwiki tiki-user\_preferences Command Injection
  - Vulnerability website, 117
- TLV (Type, Length, Value) style syntax, 352
- tools. *See also* debuggers
  - Autodafe[as], 369-371
  - AxMan, 275
  - beSTORM, 138, 508
  - BinAudit, 18
  - BoundsChecker, 493
  - BreakingPoint, 509
  - BugScam, 17
  - clfuzz, 38
  - Codonomicon, 510-511
    - foundation, 23
    - HTTP test tools, 41, 138
  - COM Raider, 42
  - compile time checkers, 6
  - COMRaider, 25, 42, 275
  - Convert, 367
  - crash binning, 487-489
  - crashbin\_explorer.py, 405
  - CSSDIE, 276, 275, 42
  - DevInspect, 504
  - Dfuz, 357-360
    - data sources, 358
    - functions, 357-358
    - lists, declaring, 358
    - protocols, emulating, 359

- rule files, 359-360
- variables, defining, 357
- DOM-Hanoi, 42
- FileFuzz, 39
  - applications, launching, 203
  - ASCII text files, 202
  - audits, saving, 204-205
  - benefits, 221
  - binary files, 201
  - case study, 217-220
  - development. *See* FileFuzz, development
  - error detection, 67
  - exception detection, 203-204
  - features, 201
  - future improvements, 221
  - goals, 200
- fuzz\_trend\_server\_protect\_5268.py, 414-416
- GPF, 366-369
- Hamachi, 42
- Holodeck, 514-515
- iFuzz, 38
  - case study, 110-111
  - development approach, 105-107
  - features, 103-105
  - fork, execute, and wait approach, 107
  - fork, ptrace/execute, and wait/ptrace approach, 108-109
  - getopt hooker, 105
  - language, 109
  - modules, 104-107
  - postdevelopment observations, 111-112
- Inspector, 18, 312
- LogiScan, 17
- mangleme, 42
- Mu-4000, 512
- Netcat, 49
- notSPIKEfile, 39
  - core fuzzing engine, 184-185
  - exception handling, 183-184
  - features, 182
  - forking off/tracing child processes, 185-187
  - interesting UNIX signals, 187
  - missing features, 182
  - not interesting UNIX signals, 188
  - programming language, 195
  - RealPix format string vulnerability, 193-195
  - shell script workarounds, 192
  - zombie processes, 189-191
- PAIMEfilefuzz, 39
- Pattern Fuzz, 368
- PDML2AD, 371
- Peach, 40, 364-366
  - code reusability, 366
  - disadvantages, 366
  - generators, 364
  - groups, 365
- publishers, 364
- transformers, 364
- Process Stalker. *See* PStalker
- ProtoFuzz
  - case study, 262-264
  - data capture, 251, 258-259
  - disadvantages, 264-265
  - fuzz variables, 253, 261
  - goals, 255
  - hexadecimal encoding/decoding, 262
  - NDIS protocol driver, 255
  - network adapters, 257-258
  - packet capture library, choosing, 256-257
  - packets, assembling, 250-251
  - parsing data, 259-261
  - programming language, 256
  - sending data, 253
- PROTOS, 458-460, 510
- ProtoVer Professional, 512
- ProxyFuzzer, 422-424
- PStalker
  - data capture, 454
  - data exploration, 453
  - data sources, 452-453
  - data storage, 455-457
  - Gizmo Project case study. *See* Gizmo Project case study
  - layout overview, 451-452
  - limitations, 454
- ptrace(), 76
- PureFuzz, 367
- Python extensions, 77
- Rational Purify, 492
- SecurityReview, 18
- Sharefuzz, 38
- Sidewinder, 433-434
- source code, 7
- SPI Fuzzer, 41, 138
- SPIKE, 361, 362, 40, 364
  - block-based protocol modeling, 242
  - block-based protocol representation, 361-362
  - disadvantage, 363
  - FTP fuzzer, 362-363
  - fuzz engine, 240
  - generic line-based TCP fuzzer, 240-243
  - generic script-based fuzzer, 244
  - protocol-specific fuzzers, 243
  - Proxy, 138
  - UNIX fuzzing, 236-237, 244-248
- SPIKE Proxy, 138
- SPIKEfile
  - core fuzzing engine, 184-185
  - exception detection engine, 183
  - exception reporting, 183-184
  - features, 182
  - forking off/tracing child processes, 185-187

- interesting UNIX signals, 187
  - missing features, 182
  - not interesting UNIX signals, 188
  - programming language, 195
  - shell script workarounds, 192
  - zombie processes, 189-191
  - Sulley framework, 387, 386
    - block helpers, 395-397
    - blocks, 392-395
  - data types, 388-390
  - delimiters, 391
  - directory structure, 387-388
  - download website, 386
  - environment, setting up, 413
  - features, 386
  - fuzzers, launching, 414-416
  - integers, 390-391
  - legos, 398-399
  - postmortem phase, 405-409
  - requests, building, 410-411
  - RPC endpoint walkthrough, 409-416
  - sessions. *See* Sulley framework, sessions
  - strings, 391
  - SuperGPF, 368
  - tracking
    - benefits, 466-467
    - binary visualization, 439-441
    - future improvements, 467-469
      - as metrics, 66
    - Outlook Express NNTP vulnerability, 447-449
      - overview, 437-439
    - PaiMei framework, 449-450
    - PStalker. *See* PStalker
    - PyDbg single stepper tool, 445-446
    - tool development, 442-447
  - TXT2AD, 371
  - Valgrind, 493
  - WebFuzz
    - approach, 148
    - asynchronous sockets, 150-153
    - benefits, 166
    - buffer overflow case study, 158-160
    - directory traversal case study, 156-157
    - error messages, 146-147
    - future improvements, 166
    - fuzz variables, 142
    - generating requests, 153-155
    - handled/unhandled exceptions, 147
    - HTML status codes, 146
    - HTTP requests, identifying, 144
    - performance degradation, 147
    - programming language, choosing, 148
    - request timeouts, 147
    - requests, 140-141
    - responses, 143, 155-156
    - SQL injection case study, 160-162
    - TcpClient class, 148-149
      - user input, 147
      - vulnerabilities, identifying, 145-147
      - XSS scripting case study, 163-166
    - WebScarab, 41, 131, 138
    - white box testing, 6-8
    - Wireshark, 75
  - to\_binary() routine, 378
  - to\_decimal() routine, 378
  - TRACE method, 124
  - tracing
    - child processes, 185-187
    - instruction execution, 444-445
  - tracking
    - benefits, 466-467
    - binary visualization, 439
      - call graphs, 439
      - CFGs, 440-441
    - future improvements, 467-469
      - as metrics, 66
    - Outlook Express NNTP vulnerability, 447-449
    - overview, 437-439
    - PaiMei framework, 449-450
    - PStalker tool
      - data capture, 454
      - data exploration, 453
      - data sources, 452-453
      - data storage, 454-457
      - Gizmo Project case study. *See* Gizmo Project case study
      - layout overview, 451-452
      - limitations, 454
    - PyDbg single stepper tool, 445-446
    - tool development, 442
      - basic blocks, tracking, 444
      - cross-referencing, 447
      - instruction execution, tracing, 444-445
      - recordings, filtering, 446
      - target profiling, 443-444
  - traffic generation, 509
  - transformers, 364
  - transport layer vulnerabilities, 229
  - Trend Micro Control Manager directory traversal, 156
  - Tridgell, Andrew, 420
  - Trustworthy Computing Security Development Lifecycle document (Microsoft), 13
  - TXT2AD, 371
  - Type, Length, Value (TLV) style syntax, 352
- ## U
- UNIX
    - file format fuzzing. *See* SPIKEfile; notSPIKEfile
    - file permissions, 95
    - interesting/not interesting signals, 187-188
    - targets, 236-237
    - unmarshal() routine, 306

- unwinding stacks, 485
  - UPGMA (Unweighted Pairwise Mean by Arithmetic Averages) algorithm, 429
  - URL vulnerabilities, 280
  - User-Agent header, 122
  - utilities. *See* tools
- V**
- Valgrind, 493
  - values (fuzz), 480-481
  - variable length fields (protocols), 48
  - variables
    - defining, 357
    - environment, 38-39, 90-92
      - GDB method, 96-97
      - getenv function, 98
      - library preloading, 98-99
    - network protocols, 253
    - ProtoFuzz design, 261
    - Web application fuzzing, 142
  - VARIANT data structure, 293
  - Vector Markup Language (VML), 273
  - viewing
    - crash locations, 405-406
    - fault detections, 406
  - VirtualQueryEx() routine, 329
  - VML (Vector Markup Language), 273
  - VMs (virtual machines), 119
  - VMWare control agent, 403
  - vulnerabilities
    - ActiveX controls, 273-275
    - address bar spoofing, 281
    - Apple Macbook, 228
    - application layer, 230
    - Brightstor backup software, 424
    - buffer overflows, 130, 281
    - client-side, 223, 280-282
    - client-side scripting, 276
    - commands
      - execution, 281
      - injection, 87
    - cross-domain restriction bypass, 281
    - CSS, 275-276
    - data link layer, 228
    - data session link layer, 229
    - directory traversal, 86
    - discussion boards, 117
    - DoS, 280
    - ERP, 117
    - Excel eBay, 199
    - file formats
      - DoS, 175
      - examples, 170
      - format strings, 177
      - heap overflows, 177
      - integer handling, 175-177
      - logic errors, 177
      - race conditions, 178
      - simple stack, 177
    - Flash, 279
    - format strings, 85, 106, 477
    - GAs, 433
    - GDI+ buffer overflow, 198, 217-220
    - heap overflow, 129
    - HTML
      - headers, 271
      - tags, 271-273
    - hybrid analysis approach, 515
    - Ipswitch I-Mail, 420
    - libraries, 134
    - log analysis, 118
    - media servers, 226
    - Microsoft source code leak, 5
    - network
      - layer, 229
      - monitoring, 118
      - target categories, 226
    - NMAP stack overflow, 246
    - Outlook Express NNTP, 447-449
    - phishing, 281
    - PNG, 199
    - precompile security solution, 480
    - presentation layer, 230
    - RealPlayer RealPix format string, 193-195
    - remote access services, 226
    - RPC-based services, 226
    - security zones, 281
    - server-side, 223-226
    - software memory corruption, 475
      - execution control, transferring, 475-476
      - reading addresses, 476-478
      - writing to addresses, 478-479
    - SPI Dynamics Free Bank application, 163-164
    - stack overflow
      - Hewlett-Packard Mercury LoadRunner, 263
      - NMAP protocol, 246
    - TCP/IP, 229
    - transport layer, 229
    - URLs, 280
    - web applications, 132-135
    - Web Mail, 117
    - weblogs, 117
    - Wikis, 117
    - Windows file formatting, 197-198
    - winnuke attack, 229
    - WinZip FileView, 298
    - WMF, 199
    - XML tags, 273

**W**

- waterfall model, 499
  - analysis, 500
  - coding, 501-502
  - design, 500-501
  - maintenance, 502-503
  - testing, 502
- weak access control, 132
- weak authentication, 133
- weak session management, 133
- web application fuzzing, 41, 138
  - benefits, 166
  - beSTORM, 138, 508
  - buffer overflows, 130, 158-160
  - Codonomicon, 138
  - configuring, 118-119
  - development
    - approach, 148
    - asynchronous sockets, 150-153
    - programming language, choosing, 148
    - requests, generating, 153-155
    - responses, receiving, 155-156
    - TcpClient class, 148-149
  - directory traversal case study, 156-157
    - Ipswitch Imail Web Calendaring, 157
    - Trend Micro Control Manager, 156
  - error messages, 146-147
  - exceptions, detecting, 135-136
  - future improvements, 166
  - fuzz variables, 142
  - handled/unhandled exceptions, 147
  - heap overflow vulnerability, 129
  - HTML status codes, 146
  - HTTP requests, identifying, 144
  - inputs
    - choosing, 119-121
    - cookies, 129
    - headers, 128
    - identifying, 130-131
    - method, 123-125
    - post data, 130
    - protocol, 128
    - request-URI, 126-128
  - overview, 113-115
  - performance degradation, 147
  - request timeouts, 147
  - requests, 140-141
  - responses, 143
  - SPI Fuzzer, 138
  - SPIKE Proxy, 138
  - SQL injection case study, 160-162
  - targets, 117-118
  - technologies, 115
  - user input, 147
  - vulnerabilities, 132-135, 145-147
    - WebScarab, 138
    - XSS scripting case study, 163-166
- web browser fuzzing, 41-42
  - ActiveX, 287-289
    - heuristics, 298
    - loadable controls, enumerating, 289-293
    - monitoring, 299
    - properties, methods, parameters, and types, 294-297
    - test cases, 298
  - approaches, 269-271
  - fault detection, 282
  - heap overflows, 277-279
  - history, 24
  - inputs, 271
    - ActiveX controls, 273-275
    - client-side scripting, 276
    - CSS, 275-276
    - Flash, 279
    - HTML headers, 271
    - HTML tags, 271-273
    - URLs, 280
    - XML tags, 273
  - methods, 269
    - approaches, 269-271
    - inputs, 275-280
  - Month of Browser Bugs, 268
  - overview, 268
  - targets, 269
  - vulnerabilities, 280-282
- Web Mail vulnerabilities, 117
- Web monitoring interface (Sulley), 404
- web server error messages, 135
- web sites
  - AWStats Remote Command Execution
    - Vulnerability, 118
  - CodeSpy, 7
  - Flawfinder, 7
  - ITS4, 7
  - Jlint, 7
  - Splint, 7
  - Sulley download, 386
  - vulnerabilities
    - Ipswitch WhatsUp Professional 2005 (SP1) SQL
      - Injection, 118
    - Microsoft Outlook Web Access Cross Site
      - Scripting, 117
    - Multiple Vendor Cacti Remote File Inclusion, 118
    - OpenSSH Remote Challenge, 226
    - phpBB Group phpBB Arbitrary File Disclosure, 117
    - RATS download, 7
    - RealServer ../ DESCRIBE, 226
    - RPC DCOM Buffer Overflow, 226
    - SAP Web Application Server sap-exiturl Header
      - HTTP Response Splitting, 117
    - Tikiwiki tiki-user\_preferences Command
      - Injection, 117

- WinZip MIME Parsing Buffer Overflow advisory, 170
  - WordPress Cookie cache\_lastpostdate Variable Arbitrary PHP Code Execution, 117
  - Wireshark, 335
  - Wotsit, 421
  - WebFuzz
    - benefits, 166
    - buffer overflow case study, 158-160
    - development
      - approach, 148
      - asynchronous sockets, 150-153
      - programming language, choosing, 148
      - requests, generating, 153-155
      - responses, receiving, 155-156
      - TcpClient class, 148-149
    - directory traversal case study, 156-157
    - error messages, 146-147
    - future improvements, 166
    - fuzz variables, 142
    - handled/unhandled exceptions, 147
    - HTML status codes, 146
    - HTTP requests, identifying, 144
    - performance degradation, 147
    - request timeouts, 147
    - requests, 140-141
    - responses, 143
    - SQL injection case study, 160-162
    - user input, 147
    - vulnerabilities, identifying, 145-147
    - XSS scripting case study, 163-166
  - weblog vulnerabilities, 117
  - WebScarab, 41, 131, 138
  - white box testing
    - pros/cons, 9
    - source code analysis, 4-5
    - tools, 6-8
  - Wikis vulnerabilities, 117
  - WinDbg, 16
  - Windows
    - debugging API, 320-323, 332-333
    - Explorer file format targets, 206, 209
    - file format fuzzing, 205-209
    - file format vulnerabilities, 197-198. *See also* FileFuzz Live, 114
    - memory model, 302-303, 306
    - Meta File (WMF) vulnerability, 199
    - Registry, file format targets, 209
  - winnuke attack, 229
  - WinPcap library, 256
  - WINRAR, 174
  - WinZip vulnerabilities
    - FileView ActiveX Control Unsafe Method Exposure, 298
    - MIME Parsing Buffer Overflow, 170
  - Wireshark, 75
    - sniffer, 237
    - web site, 335
  - WMF (Windows Meta File) vulnerability, 199
  - WordPress Cookie cache\_lastpostdate Variable Arbitrary PHP Code Execution web site, 117
  - worms (Microsoft), 224-226
  - Wotsit web site, 421
  - WriteProcessMemory() function, 319
  - write\_process\_memory() routine, 344
  - writing
    - addresses, 478-479
    - INT3 opcode, 324
    - process memory, 319
  - WS2\_32.dll recv() breakpoint, 336
- X**
- XDR (External Data Representation), 230
  - XML tag vulnerabilities, 273
  - xmlComposeString() routine, 407
  - XSS (cross-site scripting), 132, 163-166
- Y - Z**
- Zalewski, Michal, 24
  - Zimmer, David, 25, 42
  - Zoller, Thierry, 24
  - zombie processes, 189-191





