

Index

Symbols

& (AND) operator, 412
- - (pair of hyphens), 162
/dev/random, 403
| (OR) operator, 412

A

A1 certification, 31
ABM (Analyzer Benchmark), 41
abstract interpretation, local
 analysis, 89
abstract syntax, building program
 models, 74-75
access
 back-door code, debugging, 290
 files, race conditions, 440-446
 passwords, exposing in source
 code, 389-391
Action class, 337
ActionForm objects, 337, 340
actions
 logging, 288
 mapping, 337
adding security reviews to existing
 development processes,
 56-62
Address Space Layout Random-
 ization (ASLR), 259
Adobe Reader, external entity
 attacks, 359-360
adoption anxiety, adding security
 reviews to existing develop-
 ment processes, 58-62
 programmers, 59
 security team, 59
AES (Advanced Encryption
 Stanard), 408
Ajax programming, JavaScript
 hijacking. *See* JavaScript
 hijacking
algorithms
 AES, 408
 analysis algorithms. *See* analysis
 algorithms, 83
 cryptography, 407
 implementing, 409-412
 selecting, 407-409
 passwords, encryption, 392-395
 RSA, 408
 SHA, 408
 SHA1PRNG, 399
 work-queue algorithm, 92
alias analysis, 82

- `_alloca()` function, 271
- allocation, buffer-allocation
 - strategies, 179-180
 - dynamic, 182-185
 - static, 180-181
- ALWAYS_TERMINATE flag, 217
- analysis
 - algorithms
 - identifying home-grown, 412
 - selecting, 409
 - compilers, optimizing, 418
 - cookies, 301-302
 - cross-site scripting, input/output validation, 318
 - debugging code, 290
 - enforcing NULL after `free()`, 186
 - entropy, 405-407
 - error messages, auditing, 326
 - exception handling, 275
 - exceptions, catching, 278
 - Fortify Source Code Analyzer (SCA)
 - applications, 478-479, 520-521
 - Audit Workbench, 479-487, 521-529
 - auditing, 487-491, 529-531
 - C, 514-515
 - customizing rules, 491-499, 531-537
 - installing, 460-461, 504-505
 - Java, 471-472
 - results, 472-473, 475-478, 515-516, 518-520
 - functions, banning, 201-203
 - logging, 287
 - passwords
 - formatting strong, 396
 - hard-coding, 390-391
 - managing, 395-396
 - privacy violations, 387-388
 - privileged programs, 452
 - `open()` method, 455
 - PRNGs, 404
 - random numbers, seeding
 - SecureRandom, 400
 - resources, managing, 285
 - session identifiers, formatting, 331
 - sessions, logout links, 333
 - strings, libraries, 232-233
 - taint propagation, 217
 - unchecked return values, 268
 - Validators, 345
- analysis algorithms, 83
 - checking assertions, 84-85
 - global analysis, 91-93
 - local analysis, 89
 - abstract interpretation, 89
 - model checking, 90
 - predicate transformers, 89-90
 - naïve local analysis, 85-89
 - research tools, 94-95
- analysis—repost under static analysis, 268
- Analyzer Benchmark (ABM), 41
- `analyze_function()`, 93
- `analyze_program()`, 93
- analyzing source code versus compiled code, 42-45
- AND, 88
- AND (&) operator, 412
- Anderson, Ross, 389
- annotations, rules, 99-100
- answers to exercises
 - C, 537-539
 - Java, 499-501

- Apache
 - configuration files, validating, 122
 - fix for buffer overflow, 123
- API, security-enhanced API, 144
- API abuse, 16
- Apple, OS X (software updates that trust too much), 129
- applications
 - Fortify Source Code Analyzer (SCA)
 - C, 520-521
 - Java, 478-479
 - privacy
 - managing, 380-383
 - violations, 383-388
 - Web
 - Struts Web Application Framework, 336-346
 - validating input/output, 298-308, 310-318
- applying
 - Audit Workbench
 - C, 529-531
 - Java, 487-491
 - POST requests, 319-321
 - secrets, 416
 - statistical PRNGs, 398-407
 - Validators, 338-341
- arc injection, 179
- Arce, Ivan, 10
- ARCHER, 94
- architectural risk analysis, 13
- architecture, JCA, 410
- arguments, command-line, 426
- ASLR (Address Space Layout Randomization), 259
- assertions, checking with analysis algorithms, 84-85
- assumptions, detecting and preventing integer overflow, 243
- attack surfaces, identifying, 120
- attacks
 - buffer overflow, 175. *See also* buffer overflow
 - cross-site request forging, 327
 - cryogenic sleep, 443
 - external entity attacks, Adobe Reader, 359-360
 - format string, 228
 - horizontal privilege escalation, 421
 - privilege-escalation, 439-446, 449-452, 454-455
 - session fixation, 334
 - vertical privilege escalation, 421
- Audit Workbench, 106
- C
 - applying, 529-531
 - reviewing audits, 505-512
- Java
 - applying, 487-491
 - Fortify Source Code Analyzer (SCA), 479-487, 521-529
 - WebGoat Version 3.7, reviewing audits, 461-468
- auditing
 - C
 - reviewing, 505-512
 - source code manually, 513-514
 - cookies, hidden fields, 301-302
 - error messages, 326
 - Java, source code manually, 469, 471
 - security
 - C, 529-531
 - Java, 487-491

- audits, 67-69
- authentication, sessions, 333-336
- avoiding blacklisting, input validation, 137-139
- B**
- back-door access code, debugging, 290
- backup files, debugging, 293
- Baker, Jeffrey, 303
- banks, phishing schemes, 316-318
- banning dangerous functions, 201-203
- Basic Multilingual Plane (BMP), 220
- benchmarks for static analysis tools, 40-41
- Berkeley fingerd daemon, 190
- black-box texts, 10
- blacklisting, 133
 - input validation, 137-139
- blacklisting, preventing cross-site scripting, 312
- Blaster virus, 176
- blocks, managing exceptions, 273-274
- BMP (Basic Multilingual Plane), 220
- BOON, 94
- bound numeric input, 157-160
- bounded operations, functions, 195-201, 203, 205-213
- browsers. *See* interfaces, Web browsers
- buffer overflow
 - allocation strategies, 179-185
 - Apache fix, 123
 - interger overflows, 235
- Java, JNI (Java Native Interface), 254-255
- overview of, 176-178
- runtime protection, 251
 - dynamic buffer overflow protections, 258-263
 - dynamic protection benchmark results, 263
 - safe programming languages, 251-253
 - safer c dialects, 255-257
- sizes, tracking, 186-188
- strings, 189
 - character sets, 218-224
 - format errors, 224-228
 - functions, 189-201, 203, 205-213
 - libraries, 229-233
 - managing null terminators, 213-218
- bug finding, problem solving with static analysis, 32-33
- Building Secure Software*, 444
- C**
- C
- answers to exercises, 537-539
- Audit Workbench, 521-529
 - reviewing audits, 505-512
- buffer overflow, 176-178
 - allocation strategies, 179-185
 - tracking sizes, 186-188
- Fortify SCA
 - applications, 520-521
 - customizing rules, 531-537
 - results, 515-520

- running, 514-515
 - Fortify Source Code Analysis,
 - installing, 504-505
 - gets(), 155
 - memory, unlocking values, 415
 - random numbers, 401-407
 - regular expressions, 136-137
 - resource leaks, preventing,
 - 278-282
 - return values, checking, 266-267
 - runtime protection, buffer overflows, 255-257
 - secrets, deleting, 416
 - security, auditing, 529-531
 - source code, auditing manually, 513-514
 - strings, 189
 - character sets, 218-224
 - format errors, 224-228
 - functions, 189-201, 203, 205-213
 - libraries, 229-233
 - managing null terminators, 213-218
 - temporary files, security, 446-451
- C++
- buffer overflow, 176-178
 - allocation strategies, 179-185
 - tracking sizes, 186-188
 - gets(), 156
 - parameterized SQL, 164-167
 - random numbers, 401-407
 - regular expressions, 136-137
 - resource leaks, preventing,
 - 278-282
 - secrets, deleting, 416
 - strings, 190. *See also* strings
 - cache, poisoning, 314
 - Carmargo, Luiz, 199
 - Cashdollar, 200
 - catching, exceptions, 274-275
 - CCured, 255
 - centralized logging, 286-289
 - time-stamp log entries, 287
 - CFG (context-free grammar), 73
 - characters, 218, 220
 - buffer overflow vulnerabilities, 222-224
 - cross-site scripting, 312
 - checked exceptions, 271. *See also* exceptions
 - maintaning, 276-278
 - checking returned values
 - in C, 266-267
 - in Java, 269-270
 - Children's Online Protection Act (COPPA), 382
 - chroot() function, 433-434
 - CL, compiler warnings (integer overflow), 244
 - ClassCastException, 274
 - classes
 - Action, 337
 - Exception, 276
 - File_handle, 281
 - java.io.File, 446
 - strings, 229-233
 - classifying vulnerabilities, 14-15
 - cleanse rules, 102
 - clear-text passwords, storing, 391-396
 - clear-text passwords. *See also* passwords, 391

- code
 - coupon, 399
 - debugging, 286, 289-292
 - backup files, 293
 - Easter eggs, 293
 - logging, 286
 - centralized, 286-289
 - time-stamp log entries, 287
 - principle of least privilege, 423-432
 - random numbers, 397
 - C/C++, 401-407
 - Java, 398-400
 - code points, 218
 - code quality, Seven Pernicious Kingdoms, 18
 - Code Red virus, 176
 - code review, performing, 48
 - review cycle, 48-53
 - steering clear of exploitability debates, 54-55
 - code values, 219
 - codes
 - exceptions
 - catching, 274-275
 - maintaining checked, 276-278
 - managing, 271
 - top levels, 272-273
 - try/finally syntax, 273-274
 - return, handling errors with, 266-270
 - collaborative auditing, 52
 - Comair Airlines, integer overflow, 238-239
 - command injection
 - preventing metacharacter vulnerabilities, 168-169
 - vulnerabilities, 450
 - command-line arguments, 426
 - command-line parameters, input validation, 124-125
 - Common Vulnerabilities and Exposures (CVE), 176
 - comparing projects by severity, 65
 - compile-time instrumentation, runtime protection (buffer overflows), 259-261
 - compiled code, analyzing (versus analyzing source code), 42-45
 - compiler warnings, detecting and preventing integer overflow, 244-245
 - compilers, optimizing, 417-418
 - confidentiality, loggers (creating to identify), 386
 - conditions
 - privileged programs, handling unexpected, 436-438
 - race, file access, 440-446
 - confidence, 107
 - confidentiality, 380. *See also* privacy
 - configuration
 - files, storing clear-text passwords, 391-396
 - Validators, 338
 - configuration files, input validation, 122-123
 - configuration information, 38
 - configuration rules, print, 104
 - connections, databases (hard-coding passwords), 389
 - constraint solvers, 96
 - context sensitivity, 83
 - context-free grammar (CFG), 73
 - context-specific defects, 14

- control flow, building program
 - models, 77-80
- control flow graphs, 77
- cookies
 - auditing, 301-302
 - session identifiers, 328-331
 - sessions, 327-328
- COPPA (Children's Online Protection Act), 382
- copying secrets, preventing, 418-419
- counterexample, 29
- coupon codes, 399, 402
- random numbers, 403
- Coverity, 33
- CQual, 94
- cross-site request forging attacks, 327
- cross-site scripting vulnerabilities, 303-308, 310-318
- cryogenic sleep attacks, 443
- CryptGenRandom() method, 401
- Crypto++, 411
- CryptoAPI, 402, 411
- cryptographic PRNGs, 397. *See also* PRNGs
- cryptography
 - algorithms, 407
 - implementing, 409-412
 - selecting, 407, 409
 - random numbers, 397
 - C/C++, 401-407
 - Java, 398-400
- custom rules, 40
- customization, rules (Fortify SCA), 491-499, 531-537
- CVE (Common Vulnerabilities and Exposures), 176
- Cyclone, 256

D

- daemons, Berkeley fingerd, 190
- data, input validation (rejecting bad data), 143-144
- dataflow, building program
 - models, 80-81
- database queries, input validation, 125-127
- databases, passwords (hard-coding), 389
- debit cards, 384
- debugging, 286, 289-292
 - backup files, 293
 - Easter eggs, 293
- decodeFile() function, 281
- decryption, passwords, 392-395
- decryptPassword() method, 396
- defect density, 63-64
- defensive programming, 4-6
- deleting secrets, 416-418
- density, 63-64
 - vulnerability density, measuring, 65
- descriptors, standard file, 452-454
- detecting integer overflow
 - bad assumptions, 243
 - compiler warnings, 244-245
 - integer conversion rules, 245
 - restricting numeric user input, 243
 - sanity checks, 244
 - unsigned types, 242-243
 - verifying conditions for operators that can overflow, 246-249
- development processes, adding security reviews to, 56, 62
- adoption anxiety, 58-62

- Direct Web Remoting (DWR),
 - 369
 - disabling signals, 437-438
 - doAlloc(), 159
 - document queries, XML, 362-366
 - doGet() method, 272
 - doPost() method, 272
 - double-free errors, 185
 - duplication, secrets (preventing),
 - 418-419
 - DWR (Direct Web Remoting),
 - 369
 - dynamic buffer allocation,
 - 182-185
 - dynamic buffer overflow protections, runtime protection (buffer overflows), 258-263
 - dynamic protection benchmark results, runtime protection (buffer overflows), 263
- E**
- E*Trade, 303
 - e-commerce
 - random numbers, 402-403
 - SecureRandom, 399
 - e-commerce Web sites, phishing schemes, 316-318
 - Easter eggs, 293
 - Eau Claire tool, 94
 - effective group IDs, 428
 - effective user IDs, 427
 - EJBs (Enterprise Java Beans),
 - 385
 - elevating privileges, disabling signals, 437-438
 - eliminating unwanted results,
 - 108-109
 - encapsulation, Seven Pernicious Kingdoms, 18
 - encoding
 - characters, 218-220
 - buffer overflow vulnerabilities, 222-224
 - preventing cross-site scripting, 312
 - UTF, 220
 - encryption
 - algorithms, implementing, 409-412
 - passwords, 392-395
 - enforcing
 - session idle timeouts, 331-333
 - trust boundaries, 131-132
 - enforcing NULL after free(), 186
 - EnterCriticalSection() function,
 - 272
 - Enterprise Java Beans (EJBs),
 - 385
 - entropy, 405-407
 - entropy, 397
 - entry points, finding, 370
 - entry-point rules, 102
 - environments
 - privileged programs, 426
 - Seven Pernicious Kingdoms, 18
 - Epstein, Jeremy, 388
 - equivalence checking, 28
 - erasing passwords, 416
 - erasing secrets, 416-418
 - error handling, Seven Pernicious Kingdoms, 18
 - errors
 - double-free, 185
 - HTTP, handling, 322, 325-326
 - messages, auditing, 326

- privileged programs, checking, 436
 - recovering from, 437
 - with return codes, handling, 266-270
 - sign extension errors, integer overflow, 239-241
 - signed-to-unsigned conversions, integer overflow, 241-242
 - strings, format, 224-228
 - truncation, 210-213
 - integer overflow, 239-241
 - Web Services, opportunities for old errors, 370
 - wrap-around errors, integer overflow, 236-238
 - escalating privilege-escalation attacks, 439-446, 449-452, 454-455
 - establishing trust boundaries. *See* trust boundaries
 - events, handling unexpected events, 436-438
 - exception, `IndexOutOfBoundsException`, 274
 - Exception class, 276
 - exceptions
 - catching, 274-275
 - checked, maintaining, 276-278
 - `ClassCastException`, 274
 - `InterruptedException`, 277
 - `java.lang.Exception`, 271, 274
 - `java.lang.Throwable`, 325
 - managing, 271
 - top levels, 272-273
 - try/finally syntax, 273-274
 - `NullPointerException`, 274
 - `RareException`, 276
 - `RuntimeException`, 276
 - `SQLException`, 388
 - `UnknownHostException`, 272
 - `execSQL()` method, 386-388
 - `executeQuery()` method, 388
 - exercises
 - Audit Workbench
 - reviewing WebGoat Version 3.7 audits, 461-468
 - C
 - answers to, 537-539
 - auditing source manually, 513-514
 - Fortify Source Code Analyzer (SCA), 514-515
 - installing Fortify Source Code Analysis, 504-505
 - reviewing audits, 505-512
 - Fortify Source Code Analyzer (SCA)
 - applications, 478-479, 520-521
 - Audit Workbench, 479-487, 521-529
 - auditing, 487-491, 529-531
 - customizing rules, 491-499, 531-537
 - results, 472-473, 475-478, 515-520
 - Java
 - answers to, 499-501
 - auditing source manually, 469-471
 - Fortify Source Code Analyzer (SCA), 471-472
 - installing Fortify Source Code Analysis, 460-461
- Exploiting Software*, 176

exploits. *See also* attacks
 arc injection, 179
 buffer overflow, 176-178
 allocation strategies, 179-185
 tracking sizes, 186-188
 format string, 228
 return-into-libc, 179
exposing passwords in source
 code, 389-391
exploitability debates, steering
 clear of, 54-55
ext, 88
external references, XML,
 358-362

F

fake Web sites, 316. *See also*
 phishing
false alarms, 23
false negatives, 23
false paths, 88
false positives, 23, 105
Federal Information Security
 Management Act (FISMA),
 382
fgets() function, 266
file systems, privileged programs,
 427
filenames, unique, 447-448
files
 access, race conditions, 440-446
 backup, debugging, 293
 configuration, storing clear-text
 passwords, 391-396
 standard file descriptors,
 452-454
 temporary, security, 446-451
 unique, 448-449

filesystems
 functions, TOCTOU vulnerabil-
 ties, 442
 privileges, restricting on,
 433-435
File_handle class, 281
finally blocks, managing excep-
 tions, 273-274
FindBugs, 33, 100
finding entry points, Web Services,
 370
FISMA (Federal Information Secu-
 rity Management Act), 382
fixation, limiting session, 334
flag functions, security-enhanced
 API, 152
flags
 ALWAYS_TERMINATE, 217
 hard-coded passwords, 391
 privayc violations, 386
flow-insensitive analysis, 89
form bean mapping, 337
formal verification, 31
formatting
 session identifiers, 329-331
 strings, errors, 224-228
 unique filenames, 447-448
forms, validator, 337
Fortify rule, 98
Fortify Software, 34
Fortify Source Code Analyzer
 (SCA)
 C
 applications, 520-521
 Audit Workbench, 521-529
 auditing, 529-531
 customizing rules, 531-537
 installing, 504-505
 results, 515-520
 running, 514-515

Java

- applications, 478-479
- Audit Workbench, 479-487
- auditing, 487-491
- customizing rules, 491-499
- installing, 460-461
- results, 472-478
- running, 471-472

Foundations of AJAX, 304

free() function, 185

- enforcing NULL after, 186

Fujaba, 27

Fujitsu, 384

function summaries, 92

functions

- banning dangerous, 201-203
- chroot(), 433-434
- decodeFile(), 281
- EnterCriticalSection(), 272
- fgets(), 266
- filesystems, TOCTOU vulnerabilities, 442
- free(), 185
 - enforcing NULL after, 186
- GetTempFileName(), 447
- Helper, sizing buffers, 188
- malloc(), 185
- mkstemp(), 449
- read(), 269
- strings
 - bounded operations, 195-201, 203, 205-213
 - gets(), 189
 - reimplementation, 194
 - scanf(), 190-191
 - sprintf(), 193
 - strcpy(), 192
- strlcat(), 196
- strncpy(), 196
- strncat(), 200, 208

strncpy(), 204

temporary files, security, 446, 448-449

Thread.sleep(), 277

tmpfile(), 449

_alloca(), 271

fuzzing, 11

G

Gaim instant-messaging client, 205

GCC, compiler warnings (integer overflow), 245

generateSeed() method, 400

generating random numbers

C/C++, 401-407

Java, 398-400

generic defects, 14

Get requests, 319-321

getConnection() method, 390-391

gets() function, 155, 189

GetTempFileName() function, 447

GLBA (Gramm-Leach-Bliley Act), 382

global analysis, 83, 91-93

goals, establishing for code review, 49-50

Gramm-Leach-Bliley Act (GLBA), 382

grouping results, 106-108

H

halting problem, 35

handling

errors

HTTP, 322, 325-326

with return code, 266-270

unexpected events, 436-438

- hard-coding passwords, 389
- hardened system libraries, runtime protection (buffer overflows), 262
- headers
 - HTTP, 301-302
 - referer, 327
 - User-Agent, input validation, 299-302
- Health Insurance Portability and Accountability Act (HIPAA), 382
- helper functions, sizing buffers, 188
- hidden fields, Show Source option, 302
- hijacking Web pages, 314
- HIPAA (Health Insurance Portability and Accountability Act), 382
- holding secrets, minimizing time, 414-415
- horizontal privilege escalation attacks, 421
- Howard, Michael, 13, 52
- HTTP
 - headers, 301-302
 - requests, validating, 298-299, 301-308, 310-318
 - responses, splitting, 314-315
 - security, 319
 - applying POST, 319-321
 - handling errors, 322, 325-326
 - maintaining session state, 328-336
 - ordering requests, 322-323
 - request provenance, 327-328
 - traffic sniffers, 302
- Huseby, Sverre H., 359-360
- I**
 - identifiers
 - random request, 328
 - session, 328-331
 - identifying attack surfaces, 120
 - identities, phishing, 316-318
 - IDEs (integrated development environments), 27
 - idle timeouts, enforcing session, 331-333
 - image-display software, vulnerabilities, 8
 - impersonating servers, 129
 - implementation, algorithms, 409-412
 - inbound passwords, 388
 - IndexOutOfBoundsException, 274
 - indirect selection, 133
 - input validation, 133-134
 - injection, command vulnerabilities, 450-452
 - input
 - Struts Web Application Framework, 336
 - analyzing Validators, 345
 - applying Validators, 338-341
 - configuring Validators, 338
 - maintaining validation logic, 343-346
 - validating parameters, 342
 - transforming, 140-141
 - validating, 298-299, 301-308, 310-318
 - input length, checking, 153-156
 - input validation, 119
 - blacklisting, 137-139
 - bound numeric input, 157-160

- check input length, 153-156
 - establishing trust boundaries, 130-131
 - good input validation as the default, 144-152
 - mistaking usability for security, 142
 - rejecting bad data, 143-144
 - strong input validation, 133-134
 - validating all input, 120
 - validating input from all sources, 121-122
 - command-line parameters, 124-125
 - configuration files, 122-123
 - database queries, 125-127
 - network services, 127-128
 - Web Services, 366-368
 - whitelisting, 135-136
 - input validation and representation, 16
 - installation, Fortify Source Code Analysis
 - C, 504-505
 - Java, 460-461
 - integer conversion rules, detecting and preventing integer overflow, 245
 - integer overflow, 236-237
 - Comair Airlines, 238-239
 - conversion between signed and unsigned data types, 241-242
 - detecting and preventing
 - bad assumptions, 243
 - compiler warnings, 244-245
 - integer conversion rules, 245
 - restricting numeric user input, 243
 - sanity checks, 244
 - unsigned types, 242-243
 - verifying conditions for operators that can overflow, 246-249
 - Java, 157-158
 - sign extension errors, 239-241
 - truncation errors, 239-241
 - user input, 250
 - wrap-around errors, 236-238
 - integer overflows, 235
 - integers, 236
 - integral user input, interger overflow vulnerabilities, 250
 - integrated development environments (IDEs), 27
 - interfaces, SingleThreadModel, 385
 - International Organization for Standardization (ISO), 220
 - Internet Explorer High Encryption Pack, 411
 - interprocedural analysis, 83
 - InterruptedException, 277
 - intraprocedural analysis, 83
 - IntSafe, 246-248
 - ISO (International Organization for Standardization), 220
- ## J
- Java
 - answers to exercises, 499-501
 - Audit Workbench, 479, 481, 483-487
 - bounds checking, 156

Java, continued

- buffer overflows, JNI (Java Native Interface), 254-255
 - cryptography, 393
 - Fortify SCA
 - applications, 478-479
 - customizing rules, 491-499
 - results, 472-478
 - running, 471-472
 - Fortify Source Code Analysis, installing, 460-461
 - gets(), 156
 - hard-coding passwords, 389
 - integer overflow, 157-158
 - passwords, storing clear-text passwords, 392
 - random numbers, 398-400
 - resource leaks, preventing, 283-285
 - return values, checking, 269-270
 - security, auditing, 487-491
 - source code, auditing manually, 469-471
 - Web applications
 - HTTP, 319-328
 - input/output validation, 298-299, 301-308, 310-318
 - maintaining session state, 328-336
 - Java Cryptography Architecture (JCA), 410
 - Java Cryptography Extension (JCE), 409-410
 - Java Modeling Language (JML), 99
 - Java Native Interface (JNI), 254-255
 - java.io.File class, 446
 - java.lang.Exception, 271, 274
 - java.lang.Throwable, 325
 - java.util.logging package, 286
 - java.util.Random, 274
 - JavaScript hijacking, Web Services, 370-375
 - preventing direct execution of responses, 375-376
 - JavaScript Object Notation (JSON), 371
 - JavaServer Pages (JSP), 43
 - JCA (Java Cryptography Architecture), 410
 - JCE (Java Cryptography Extension), 409-410
 - JML (Java Modeling Language), 99
 - JNI (Java Native Interface), 254-255
 - JSON (JavaScript Object Notation), 371
- K**
- keys, secret, 393-395
 - Klockwork, 33
- L**
- LAPSE (Lightweight Analysis for Program Security in Eclipse), 94
 - leaks, resource (preventing), 278-285
 - least privilege, principle of, 423-432
 - lexical analysis, 72-73
 - libraries
 - hardened system libraries, runtime protection (buffer overflows), 262
 - strings, 229-233
 - Libsafe, 262

- Libverify, 262
- lifetimes, maximum session, 331-333
- limitations of static analysis, 23
- limiting session fixation, 334
- links, logout, 333
- Linux, privileges, 423, 427-432
- Lipner, Steve, 13, 48
- Livshits, Benjamin, 40
- local analysis, 83, 89
 - abstract interpretation, 89
 - model checking, 90
 - predicate transformers, 89-90
- log forging, preventing meta-character vulnerabilities, 169-172
- loggers, 388
 - creating, 386
- logging, 286
 - centralized, 286-289
 - time-stamp log entries, 287
 - queries, 384-385
- logic, validation, 343-346
- login, privileges, 428
- logout links, 333
- lowering, 75

- M**
- malloc(), 159, 185
- management
 - exceptions, 271
 - catching, 274-275
 - maintaining checked, 276-278
 - top levels, 272-273
 - try/finally syntax, 273-274
 - null terminators, 213-218
 - passwords, 395-396
 - formatting strong, 396
 - privacy, 380-383
 - violations, 383-385, 387-388
 - privileged programs, 427
 - applying principle of least privilege, 427-432
 - handling unexpected events, 436-438
 - privilege-escalation attacks, 439-446, 449-452, 454-455
 - restricting on filesystems, 433-435
 - requests, ordering, 322-323
 - resources, preventing leaks, 278-285
 - session state, 328-336
 - validation logic, 343-346
- manual bounded operation
 - checks, 199
- manual null-terminate, bounded operations, 207
- maps
 - ActionForm objects, 340
 - actions, 337
 - form bean mapping, 337
- maximum session lifetimes, 331-333
- McGraw, Gary, 13
- measuring vulnerability density, 65
- memory
 - buffer overflow, 176-178
 - allocation strategies, 179-185
 - tracking sizes, 186-188
 - nonexecutable memory
 - segments, 258-259
 - secrets, 412-413
 - applying, 416
 - deleting, 416-418
 - minimizing time holding, 414-415
 - preventing duplication, 418-419

- memory safety, 251
 - memset() method, 417-418
 - messages, auditing errors, 326
 - metacharacters, preventing
 - vulnerabilities, 160-161
 - command injection, 168-169
 - log forging, 169-172
 - parameterized requests, 161-166
 - path manipulation, 167-168
 - methods
 - CryptGenRandom(), 401
 - decryptPassword(), 396
 - doGet(), 272
 - doPost(), 272
 - execSQL(), 386, 388
 - executeQuery(), 388
 - generateSeed(), 400
 - getConnection(), 390
 - getConnection(), 391
 - memset(), 417-418
 - open(), 455
 - Random.nextInt(), 398
 - rand_s(), 402
 - realloc(), 419
 - RtlGenRandom(), 401
 - RtlGenRandom(), 402
 - Utils.processHost(), 272
 - methods,doGet(), 272
 - methods,doPost(), 272
 - metrics
 - process metrics, 67-69
 - static analysis metrics, 62
 - breaking down results by category, 66
 - comparing projects by severity, 65
 - measuring vulnerability density, 65
 - monitoring trends, 66
 - process metrics, 67-69
 - Microsoft Passport, debugging, 291-292
 - Microsoft CryptoAPI, 411
 - mkstemp() function, 449
 - model checking, local analysis, 90
 - model-checking rules, print, 104
 - Model-View-Controller (MVC)
 - pattern, 336
 - modifying memory, buffer overflow, 176-178
 - MOPS (Model Checking Programs for Security properties), 95
 - Morris Worm, 176, 190
 - MVC (Model-View-Controller)
 - pattern, 336
 - MySpace, cross-site scripting, 309-312
- N**
- naïve local analysis, analysis algorithms, 85-89
 - names, unique filenames, 447-448
 - Nettle, 411
 - network services, input validation, 127-128
 - Nimda virus, 176
 - nonexecutable memory segments, runtime protection (buffer overflows), 258-259
 - NOT, 88
 - notation, 88
 - NULL, free() function (enforcing after), 186
 - null terminators, managing, 213-218

NullPointerException, 274
numbers, random, 274, 397
 C/C++, 401-407
 Java, 398-400
numeric user input, detecting and
 preventing integer overflow,
 243

O

objects, ActionForm, 337, 340
open redirects, 316-318
Open Web Application Security
 Project (OWASP), 19, 461
open() method, 455
operators
 AND (&), 412
 OR (|), 412
 sizeof, 186
optimization, compilers, 417-418
options, Show Source, 302
OR, 88
OR (|) operator, 412
Orange Book, 31
ordering requests, 322-323
Ounce Labs, 34
outbound passwords, 388
 clear-text passwords, storing,
 391-396
 source code, exposing in,
 389-391
output, validating, 298-299,
 301-308, 310-318
over exposure, Web Services, 369
OWASP (Open Web Application
 Security Project), 19, 461
*OWASP Guide to Building Secure
 Web Applications*, 312

P

packages, java.util.logging, 286
parameterized requests,
 preventing metacharacter
 vulnerabilities, 161-166
parameterized SQL in C++,
 164-167
parameters, validating, 342
parse trees, 73
parsers, standards-compliant
 XML parsers, 350-352
parsing, building program models,
 73-74
partial specification, 29
pass-through rules, 102
Passport, debugging, 291-292
passwords
 decryption, 392-395
 encryption, 392-395
 erasing, 416
 formatting, 396
 inbound, 388
 managing, 395-396
 outbound, 388
 exposing in source code,
 389-391
 storing clear-text passwords,
 391-396
patches, updating passwords, 389
path manipulation, preventing
 metacharacter vulnerabili-
 ties, 167-168
patterns, MVC, 336
Payment Card Industry (PCI)
 Data Security Standard,
 382-384, 392
PCI (Payment Card Industry)
 Data Security Standard,
 382-384, 392

- penetration tests, 10
- performing code review, 48
 - review cycle, 48-53
 - steering clear of exploitability debates, 54-55
- personal identification numbers (PINs), 384
- Petroski, Henry, 3
- phishing, 304, 316-318
 - Phishing and Countermeasures*, 316
- PID (process ID), 403
- PINs (personal identification numbers), 384
- Pixy, 95
- pointer aliasing, building program models, 82-83
- pointers, volatile, 417
- POST requests, applying, 319-321
- postcondition, 105
- PQL (Program Query Language), 101
 - Practical Cryptography*, 407
- precondition, 104
- predicate transformers, local analysis, 89-90
- preventing
 - direct execution of responses, JavaScript hijacking, 375-376
 - integer overflow
 - bad assumptions, 243
 - compiler warnings, 244-245
 - integer conversion rules, 245
 - restricting numeric user input, 243
 - sanity checks, 244
 - unsigned types, 242-243
 - verifying conditions for operators that can overflow, 246-249
- metacharacter vulnerabilities, 160-161
 - command injection, 168-169
 - log forging, 169-172
 - parameterized requests, 161-166
 - path manipulation, 167-168
 - secret duplication, 418-419
- prevention
 - buffer overflow
 - allocation strategies, 179-185
 - overview of, 176-178
 - tracking sizes, 186-188
 - cross-site scripting, 312-314
 - resource leaks, 278-285
- principle of least privilege, 423-432
- print, rules, 103
 - configuration rules, 104
 - model-checking rules, 104
 - structural rules, 104
 - taint-propagation rules, 104-105
- privacy, 380
 - managing, 380-383
 - violations, 383-388
- Privilege Separated OpenSSH project, 433
- privilege-escalation attacks, 439-440, 442-444, 446, 449-452, 454-455
- privileged programs, 421-423
 - command-line arguments, 426
 - environments, 426
 - file systems, 427

- managing, 427
 - applying principle of least privilege, 427-432
 - handling unexpected events, 436-438
 - restricting on filesystems, 433-435
- principle of least privilege, 423-432
- privilege-escalation attacks, 439-440, 442-444, 446, 449-452, 454-455
- PRNGs (pseudo-random number generators), 397
 - C/C++, generating, 401-407
 - Java, generating, 398-400
- process ID (PID), 403
- process metrics, 67-69
- profiles, privileges, 424
- program models, building, 72
 - abstract syntax, 74-75
 - lexical analysis, 72-73
 - parsing, 73-74
 - pointer aliasing, 82-83
 - semantic analysis, 76
 - taint propagation, 82
 - tracking control flow, 77-80
 - tracking data flow, 80-81
- Program Query Language (PQL), 101
- program slicing, 88
- program understanding, problem solving with static analysis, 27
- program verification, problem solving with static analysis, 28-31
- programmers, adoption anxiety (adding security reviews to existing development processes), 59
- Programming Jakarta Struts, 2nd Edition (*italics*), 336
- programming languages, runtime protection (buffer overflows), 251-253
- programs
 - privacy
 - managing, 380-383
 - violations, 383-388
 - privileged, 421-423
 - applying principle of least privilege, 427-432
 - command-line arguments, 426
 - environments, 426
 - file systems, 427
 - handling unexpected events, 436-438
 - managing, 427
 - principle of least privilege, 423-432
 - privilege-escalation attacks, 439-440, 442-444, 446, 449-452, 454-455
 - restricting on filesystems, 433-435
- projects, comparing by severity, 65
- propagation, taint, 387-388
- property checking, problem solving with static analysis, 28-31
- ProPolice, 259

protecting private data, 382. *See also* privacy
protection, Web browsers,
303-308, 310-318
provenance, requests, 327-328
pseudo-random number genera-
tors. *See* PRNGs
publicly vetted algorithms, imple-
menting, 409-412

Q

quality, 9
testing, 9-11
queries
database queries, input valida-
tion, 125-127
document queries, XML,
362-366
logging, 384-385
privacy violations, 388

R

race conditions, file access,
440-446
RAII (Resource Acquisition Is
Initialization), 281
random numbers, 274, 397
C/C++, 401-407
Java, 398-400
random request identifiers, 328
Random.nextInt() method, 398
rand_s() method, 402
RareException, 276
read() function, 269
readlink(), 146-149
real group IDs, 428
real user IDs, 427

realloc() method, 419
reasons for not fixing bad code,
55
recovering from errors, 437
recovering passwords, 389
references, external references
(XML), 358-362
referrer headers, 327
reflected cross-site scripting, 306.
See also cross-site scripting
regular expressions in C and C++,
136-137
regulation, 380
reimplementation, functions, 194
rejecting bad data, input valida-
tion, 143-144
Remote Procedure Call (RPC),
349
reporting results, 105-106
eliminating unwanted results,
108-109
grouping and sorting, 106-108
significance of results, 109-113
requests
GET, 319-321
HTTP, validating, 298-299,
301-308, 310-318
ordering, 322-323
POST, applying, 319-321
provenance, 327-328
random identifiers, 328
requirements
privacy, 380
managing, 380-383
violations, 383-388
privileges, 424
research tools, analysis algo-
rithms, 94-95

- resizing buffers, 187
- Resource Acquisition Is Initialization (RAII), 281
- resources, preventing leaks, 278-285
- responses, HTTP (splitting), 314-315
- restrictions, privileges on filesystems, 433-435
- results
 - breaking down by categories, 66
 - Fortify Source Code Analyzer (SCA)
 - Audit Workbench, 479-487, 521-529
 - C, 515-520
 - Java, 472-478
 - reporting, 105-106
 - eliminating unwanted results, 108-109
 - grouping and sorting, 106-108
 - significance of results, 109-113
- return codes, handling errors, 266-270
- return-into-libc, 179
- review cycle, performing code review, 48-53
 - establishing goals, 49-50
 - making fixes, 53
 - running static analysis tools, 50-51
- reviewing audits, C, 505-512
- Rice's theorem, 35
- Rice, Henry, 35
- Rivest, Shamir, and Adleman. *See* RSA algorithm
- RPC (Remote Procedure Call), 349
- RSA (Rivest, Shamir, and Adleman) algorithm, 408
- RtlGenRandom() method, 401-402
- rule formats, 97, 100-101
 - annotations, 99-100
 - specialized rule files, 97-98
- rule sets, 40
- rules, 96-97
 - Fortify, 98
 - Fortify SCA, customizing, 491-497, 499, 531-537
 - Functions, banning, 201-203
 - print, 103
 - configuration rules, 104
 - model-checking rules, 104
 - structural rules, 104
 - taint-propagation rules, 104-105
 - rule formats, 97, 100-101
 - annotations, 99-100
 - specialized rule files, 97-98
 - taint propagation, 101-103
- running Fortify SCA
 - C, 514-515
 - Java, 471-472
- runtime protection, buffer overflows, 251
 - dynamic buffer overflow protections, 258-263
 - dynamic protection benchmark results, 263
 - safe programming languages, 251-253
 - safer C dialects, 255-257
- RuntimeException, 276

S

- Safe Harbor Privacy Framework, 382
- SafeInt, 246
- safety, bounded operations, 205-206
- SAL (Standard Annotation Language), 100
- SAMATE group, 41
- sanity checks, detecting and preventing integer overflow, 244
- Sasser virus, 176
- SATURN, 95
- saved group IDs, 428
- saved user IDs, 427
- SCA (Fortify Source Code Analyzer), 98
- C
 - applications, 520-521
 - Audit Workbench, 521-529
 - auditing, 529-531
 - customizing rules, 531-537
 - results, 515-520
 - running, 514-515
- Java
 - applications, 478-479
 - Audit Workbench, 479-487
 - auditing, 487-491
 - customizing rules, 491-497, 499
 - results, 472-478
 - running, 471-472
- scanf() function, 190-191
- scripting cross-site, 303-308, 310-318
- secret keys, 393-395
- secrets
 - decryption, 392-395
 - encryption, 392-395
 - memory, 412-413
 - applying, 416
 - deleting, 416-418
 - minimizing time holding, 414-415
 - preventing duplication, 418-419
 - random numbers, 397
 - C/C++, 401-407
 - Java, 398-400
- Secure Hash Algorithm (SHA), 408
- SecureRandom, 399
- security
 - auditing
 - C, 529-531
 - Java, 487-491
 - HTTP, 319
 - applying POST requests, 319-321
 - handling errors, 322-326
 - maintaining session state, 328-336
 - ordering requests, 322-323
 - request provenance, 327-328
 - input validation, 142
 - passwords
 - exposing in source code, 389-391
 - outbound, 388
 - storing clear-text passwords, 391-396
 - random numbers, 397
 - C/C++, 401-407
 - Java, 398-400
 - temporary files, 446-451

- Security Engineering, 389
- security features, 6-9
 - Seven Pernicious Kingdoms, 17
- security reviews
 - adding to existing development processes, 56, 62
 - adoption anxiety, 58-62
 - examples of, 56-57
 - problem solving with static analysis, 33-35
- security teams, adoption anxiety (adding security reviews to existing development processes), 59
- security-enhanced API, 144
 - flag functions, 152
- selection, algorithms, 407-409
- semantic analysis, building
 - program models, 76
- semantic checks, 120
- Sendmail 8.10.1, privileges, 436
- servers, impersonating, 129
- Service-Oriented Architecture (SOA), 349
- services, network services (input validation), 127-128
- Servlet
 - privacy violations, 385-386
 - SingleThreadModel interface, 385
- session fixation attacks, 334
- sessions
 - authentication, 333-336
 - cookies, 327-328
 - maximum lifetimes, 331-333
 - random numbers, 397
 - C/C++, 401-407
 - Java, 398-400
 - state, maintaining, 328-336
- setuid root, 424-426
- Seven Pernicious Kingdoms, 15-16
 - API abuse, 16
 - code quality, 18
 - encapsulation, 18
 - environment, 18
 - error handling, 18
 - input validation and representation, 16
 - security features, 17
 - time and state, 17
 - vulnerabilities, 19
- severity, 107
 - rules, 105
- SHA (Secure Hash Algorithm), 408
- SHA1PRNG algorithm, 399
- The Shellcoder's Handbook*, 176
- Show Source option, 302
- Siegel, Aaron, 77
- sign extension errors, integer overflow, 239-241
- signals, disabling, 437-438
- signed data types, integer overflow, 241-242
- Simplify, 96
- SingleThreadModel interface, 385
- sink rules, 101
- sizeof operator, 186
- sizes, buffer (tracking), 186-188
- Slammer virus, 176
- SMTP daemon qwik-smtpd,
 - reviewing audit of, 505-512
- sniffers, traffic, 302
- SOA (Service-Oriented Architecture), 349
- software security, 4

- software-development methodologies, steps of, 11-13
- solving problems with static analysis, 24
 - bug finding, 32-33
 - program understanding, 27
 - program verification, 28-31
 - property checking, 28-31
 - security review, 33-35
 - style checking, 26-27
 - type checking, 24-25
- sorting results, 106-108
- source code, 460. *See also* code analyzing, versus compiled code, 42-45
- C, auditing manually, 513-514
- Fortify Source Code Analyzer (SCA)
 - applications, 478-479, 520-521
 - Audit Workbench, 479-487, 521-529
 - auditing, 487-491, 529-531
 - C, 514-515
 - customizing rules, 491-499, 531-537
 - installing, 460-461, 504-505
 - Java, 471-472
 - results, 472-478, 515-520
 - Java, auditing manually, 469-471
 - passwords, exposing in, 389-391
- Source Code Analyzer (SCA)
 - C
 - applications, 520-521
 - Audit Workbench, 521-529
 - auditing, 529-531
 - customizing rules, 531-537
 - results, 515-520
 - running, 514-515
 - Java
 - applications, 478-479
 - Audit Workbench, 479-487
 - auditing, 487-491
 - customizing rules, 491-497, 499
 - results, 472-478
 - running, 471-472
 - source rules, 101
 - specialized rule files, 97-98
- Splint, 95
- splitting, HTTP response, 314-315
- sprintf() function, 193
- SQL, parameterized SQL in C++, 164-167
- SQLException, 388
- SSP (Stack Smashing Protection), 259
- StackGuard, 260
- stacks, buffer overflow, 178
- StackShield, 261
- Standard Annotation Language (SAL), 100
- standard file descriptors, 452-454
- starting sessions upon authentication, 333-336
- state, maintaining sessions, 328-336
- static analysis, 3
 - benchmarks, 40-41
 - capabilities of, 22-23
 - limitations of, 23
 - methodologies, 11-13
 - solving problems with, 24
 - bug finding, 32-33
 - program understanding, 27

- program verification, 28-31
- property checking, 28-31
- security review, 33-35
- style checking, 26-27
- type checking, 24-25
- success criteria, 36-37
 - ease of use, 41-42
 - finding the right stuff, 40
 - programs, understanding, 37-38
 - trade-offs, 38-39
- static analysis metrics, 63
 - breaking down results by categories, 66
 - comparing projects by severity, 65
 - measuring vulnerability density, 65
 - monitoring trends, 66
 - process metrics, 67-69
- static buffer allocation, 180-181
- statistical PRNGs, applying, 398-407. *See also* PRNGs
- std::string class, 229
- storage, clear-text passwords, 391-396
- stored cross-site scripting, 308.
See also cross-site scripting
- strategies, buffer-allocation, 179-180
 - dynamic, 182-185
 - static, 180-181
- strcpy() function, 192
- strcpy(), 34
- string passwords, formatting, 396
- strings, 189
 - buffer overflow, 175
 - functions
 - bounded operations, 195-203, 205-213
 - character sets, 218-224
 - format errors, 224-228
 - gets(), 189
 - managing null terminators, 213-218
 - reimplementation, 194
 - scanf(), 190-191
 - sprintf(), 193
 - strcpy(), 192
 - libraries, 229-233
 - strlcat() function, 196
 - strncpy() function, 196
 - strncat() function, 200, 208
 - strncpy() function, 204
 - structural analysis, 76
 - structural rules
 - checking, 77
 - print, 104
 - Struts in Action*, 336
 - Struts Web Application Framework, 336
 - logic, maintainig validation, 343-346
 - parameters, validating, 342
 - Validator
 - applying, 338-341
 - configuring, 338
 - static analysis, 345
 - style checking, problem solving with static analysis, 26-27
- success criteria, 36-37
 - ease of use, 41-42
 - finding the right stuff, 40
 - programs, understanding, 37-38
 - trade-offs, 38-39
- surrogate pairs, 219
- symbolic simulation, 87
- syntax checks, 120
- system identifiers, 358

T

- taint flags, 103
 - taint propagation, 217, 387-388
 - building program models, 82
 - rules, 101-105
 - TCSEC (Trusted Computer System Evaluation Criteria), 31
 - temporal safety properties, 29
 - temporary files, security, 446-451
 - termination, null (managing), 213-218
 - test, penetration tests, 10
 - testing, 9-11
 - fuzzing, 11
 - tests, black-box tests, 10
 - The Unicode Standard, 218
 - Thread.sleep() function, 277
 - throwing exceptions, checked, 276-278
 - time and state, Seven Pernicious Kingdoms, 17
 - time-of-check, time-of-use (TOCTOU) race conditions, 440-446
 - time-stamp log entries, 287
 - timeouts, sessions (enforcing idle), 331-333
 - tmpfile() function, 449
 - TOCTOU (time-of-check, time-of-use) race conditions, 440-446
 - Tomcat Servlet Container, 274
 - top levels, managing exceptions, 272-273
 - tracking
 - buffer sizes, 186-188
 - control flow, building program models, 77-80
 - data flow, building program models, 80-81
 - privacy violations, 387-388
 - traffic, sniffers, 302
 - transforming input, 140-141
 - transitions, privilege profiles, 424
 - trends, monitoring, 66
 - truncation
 - errors, 210-213
 - integer overflow, 239-241
 - static buffer allocation strategies, 181
 - trust
 - Apple OS X, software updates that trust too much, 129
 - privileged programs, 426-427
 - trust boundaries
 - enforcing, 131-132
 - input validation, 130-131
 - TRUSTe, 387
 - Trusted Computer System Evaluation Criteria (TCSEC), 31
 - try blocks, managing exceptions, 273-274
 - Turing, Alan, 35
 - type checking, problem solving with static analysis, 24-25
 - type safety, 251
 - types, buffer overflow, 176-178
 - allocation strategies, 179-185
 - tracking sizes, 186-188
-
- U**
- UAC (User Account Control), 423
 - UCS (Universal Character Set), 220
 - unchecked exceptions, 271. *See also* exceptions

- undecidability, 35
 - unexpected events, handling, 436-438
 - Unicode Transformation Format (UTF), 220
 - unique filenames, 447-448
 - unique files, 448-449
 - Universal Character Set (UCS), 220
 - UNIX, privileges, 423, 427-432
 - UnknownHostException, 272
 - unlocking values in memory, 415
 - unsigned data types
 - detecting and preventing integer overflow, 242-243
 - integer overflow, 241-242
 - unterminated strings, 217
 - updating passwords, 389
 - usability, input validation, 142
 - User Account Control (UAC), 423
 - user input, integer overflow, 250
 - User-Agent headers, input validation, 299-302
 - UTF (Unicode Transformation Format), 220
 - Utils.processHost() method, 272
- V**
- validation
 - bounded string operations, 197-198
 - input validation
 - blacklisting, 137-139
 - bound numeric input, 157-160
 - check input length, 153-156
 - establishing trust boundaries, 130-131
 - good input validation as the default, 144-152
 - mistaking usability for security, 142
 - rejecting bad data, 143-144
 - strong input validation, 133-134
 - logic, 343-346
 - parameters, 342
 - XML, 352-357
 - validator forms, 337
 - Validators
 - applying, 338-341
 - configuring, 338
 - static analysis, 345
 - values
 - code, 219
 - memory, unlocking, 415
 - return
 - checking in C, 266-267
 - checking in Java, 269-270
 - verifying conditions for operators that can overflow, 246-249
 - vertical privilege escalation attacks, 421
 - viewing
 - filesystems, 433
 - hard-coded passwords, 390-391
 - Show Source option, 302
 - violations, privacy, 383-388
 - virtual execution environments, runtime protection (buffer overflows), 262
 - viruses, buffer overflow, 176. *See also* buffer overflow
 - volatile pointers, 417
 - Von Neumann, John, 397
 - Vstr library, 230

- vulnerabilities
 - buffer overflow. *See* buffer overflow
 - classifying, 14-15
 - context-specific defects, 14
 - cross-site scripting, 303-308, 310-318
 - Easter eggs, 293
 - generic defects, 14
 - in image-display software, 8
 - preventing metacharacter
 - vulnerabilities, 160-161
 - command injection, 168-169
 - log forging, 169-172
 - parameterized requests, 161-166
 - path manipulation, 167-168
 - Seven Pernicious Kingdoms
 - API abuse, 16
 - code quality, 18
 - encapsulation, 18
 - environment, 18
 - error handling, 18
 - input validation and representation, 16
 - security features, 17
 - time and state, 17
 - vulnerabilities, 19
 - strings, 189
 - character sets, 218-224
 - format errors, 224-228
 - functions, 189-201, 203, 205-213
 - libraries, 229-233
 - managing null terminators, 213-218
 - temporary files, 446, 448-449
 - vulnerability density, measuring, 65
 - vulnerability dwell, 67
 - vulnerabilities. *See also* attacks
 - command injection, 450-452
 - TOCTOU, 440-446
- W**
- Wagner, David, 176, 397
 - Wall Street Journal, 384
 - weak session identifiers, 331
 - weakest precondition (WP), 89
 - Web applications
 - Java
 - HTTP, 319-328
 - input/output validation, 298-299, 301-308, 310-318
 - maintaining session state, 328-336
 - Struts Web Application Framework, 336
 - analyzing Validators, 345
 - applying Validators, 338-341
 - configuring Validators, 338
 - maintaining validation logic, 343-346
 - validating parameters, 342
 - Web browsers
 - input validation, 299-302
 - protecting, 303-308, 310-318
 - Sow Source option, 302
 - Web pages, hijacking, 314
 - Web Services, 349, 366
 - DWR, 369
 - entry points, finding, 370
 - input validation, 366-368
 - JavaScript hijacking, 371-375

- preventing direct execution of responses, 375-376
 - opportunities for old errors, 370
 - over exposure, 369
 - WSDL, 368-369
 - Web Services Description Language (WSDL), 368-369
 - Web sites, phishing, 316-318
 - WebGoat Version 3.7, reviewing audits, 461-468
 - WebMethods, 388
 - WebSphere Application Server (version 6.1), 392
 - whitelisting, input validation, 135-136
 - whole-program analysis, 92
 - Wilander, John, 263
 - Windows Vista, UAC, 423
 - work-queue algorithm, 92
 - worms
 - cross-site scripting, 309-312
 - Morris Worm, 176, 190
 - WP (weakest precondition), 89
 - wrap-around errors, integer overflow, 236-238
 - WSDL (Web Services Description Language), 368-369
- X**
- xg++, 95
 - XML (Extensible Markup Language), 349-350
 - document queries, 362-366
 - external references, 358-362
 - standards-compliant XML parsers, 350-352
 - validation, 352-357
 - XML injection, 354
 - XML Schema, 355
 - XPath injection, 362
 - XySSL, 411
- Z**
- Zotob virus, 176