

Index

A

- Absinthe, 322
- absolute URLs, open redirection
 - vulnerabilities
 - blocking, 544–545
 - prefix, 545–546
- “accept known good” approach, input, 24
- access
 - ASP attackers, 658–660
 - ASP.NET API methods
 - database, 721
 - file, 720
 - ASPs and customer, 665–666
 - database
 - ASP.NET API methods, 721
 - Java API methods, 714–715
 - Perl language API methods, 737–738
 - PHP API methods, 729–730
 - defense mechanisms handling, 18–21
 - authentication, 18–19
 - control, 20–21
 - session management, 19–20
 - Java API methods
 - database, 714–715
 - file, 713
 - Perl language API methods
 - database, 737–738
 - file, 737
 - PHP API methods
 - database, 729–730
 - file, 727–729
 - shared hosting
 - attackers, 658–660
 - customer, 665–666
 - trust relationships in tiered architecture, 649
 - access controls
 - account testing, 267–270
 - API methods, 276–277
 - HTTP methods, 278
 - limited access, 273–276
 - multistage function, 271–273
 - static resources, 277
 - application mapping, 268–269
 - attackers, 266–278
 - types, 258–260
 - usernames and passwords, 275–276
 - back-end components, 357
 - broken, 7, 274
 - context-dependent, 258
 - declarative, 282–283
 - defective, 257
 - discretionary, 282
 - flaws, 284
 - hacker’s methodology
 - insecure access, 823
 - limited access, 822–823
 - multiple accounts, 822
 - requirements, 821
 - horizontal, 258
 - identifier-based functions, 261–262
 - insecure methods, 265–266
 - location-based, 266
 - multistage functions, 262–263
 - testing, 271–273
 - parameter-based, 265–266
 - per-user segregation, 274
 - platforms, 264–265
 - programmatic, 282
 - referrer-based, 266
 - role-based, 282
 - security, 278–283
 - best practices, 279–280
 - central component approach, 280
 - multilayered privilege model, 280–283
 - pitfalls, 278–279
 - static resources, 263–264
 - account testing, 277
 - unprotected functionality, API methods, 260–261
 - vertical, 258
 - vulnerabilities, 258–266, 276
 - application logic flaws, 411
 - Access-Control-Allow-Origin headers, 528–529
 - account activation URLs, 184
 - account suspension, 197–198
 - account testing, access controls, 267–270
 - API methods, 276–277
 - HTTP methods, 278
 - limited access, 273–276
 - multistage function, 271–273
 - static resources, 277
 - Achilles proxy, 751
 - Action Message Format (AMF), 135
 - Burp Suite, 137
 - active scanning, 764–765
 - ActiveX controls, 447
 - COMRaider, 558

- hacker's methodology, browser
 - extensions, 804
- HTML modification, 557
- "safe for scripting"
 - registration, 555–557
- vulnerabilities, 555–556
 - finding, 556–558
 - preventing, 558–559
- administrative functions, web
 - applications, 35–36
- administrators
 - DBA, 325–326
 - defense mechanisms handling
 - attackers, alerting, 33–34
- Ajax
 - HTML5, 487
 - stored XSS in uploaded files
 - via, 486–487
 - web functionality, 62–63, 384
- Alcon, Wade, 565
- alerts, 33–34
- Allaire JRun, 690–691
- allow_url_include, 729
- AMF. *See* Action Message Format
- ampersand character, batch
 - function, 360–361, 363
- Anley, Chris, 218, 322, 634
- anomalous event alerts, 33
- anti-CSRF tokens, 508–509, 516–517
 - XSS defeating, 509–510
- anti-XSS filters, 452
 - IE, 748
- AOL AIM Enterprise Gateway
 - application, 409
- Apache
 - chunked encoding overflow, 688
 - error messages, 628
 - mod_isapi, 688
 - mod_proxy, 688
 - reflected XSS, 442
 - Tomcat, 673
 - virtual hosting, 683
- API methods
 - access controls to, 260–261
 - account testing, 276–277
- ASP.NET
 - database, 721
 - dynamic code execution, 722
 - file access, 720
 - OS command execution, 722–723
 - sockets, 723
 - URL redirection, 723
 - user input, 718–719
- Java
 - database access, 714–715
 - dynamic code execution, 715
 - file access, 713
 - OS command execution, 715–716
 - potentially dangerous, 713–716
 - sockets, 716
 - URL redirection, 716
- Java user input, 712
- JavaScript DOM-based, 740
- Perl language
 - database access, 737–738
 - dynamic code execution, 738
 - file access, 737
 - OS command execution, 738
 - potentially dangerous, 736–739
 - sockets, 739
 - URL redirection, 738
- PHP
 - database access, 729–730
 - dynamic code execution, 730–731
 - file access, 727–729
 - OS command execution, 731
 - potentially dangerous, 727–732
 - sockets, 732
 - URL redirection, 731–732
 - server-side redirection, 392
 - SQL injection, 291
 - versatility, 358
- Apple iDisk Server, path
 - traversal vulnerabilities, 690
- application. *See* web application
- application architecture. *See* tiered architectures
- application logic flaws
 - access controls vulnerabilities, 411
 - attack surface, 405
 - audit trail, 429
 - authentication, 415–416
 - avoiding, 428–429
 - beating business limit, 416–417, 429
 - breaking bank, 414–416
 - bulk discount cheating, 418, 429
 - debugger messages, 424–426
 - developers, 429–430
 - encryption oracle, 407–408
 - "remember me" function, 407
 - escaping, 419–420
 - financial services, 412–416
 - forced browsing, 411
- hacker's methodology
 - attack surface, 842
 - incomplete input, 843
 - multistage functions, 842–843
 - transaction logic, 844
 - trust relationships, 844
- hacker's methodology, authentication, 811–813
- invalidating input validation, 420–422
- lessons, 428–429
- login function, 426–427
 - race conditions, 427
- nature of, 406
- password change function, 409–410
- proceeding to checkout, 410–411
- real-world, 406–407
- rolling your own insurance, 412–413
- search function, 429
 - abuse, 422–424
- security, 428
- session management, 429
- shell metacharacters, 419
- source code, 428
- SQL injection, 420–422
- application logs, 262
- application mapping, 73
 - access controls, 268–269
 - analyzing, 97–113
 - key areas, 97–98
 - attack surface, 111
 - example, 112–113
- Burp Suite, 268
- comparisons, 268–269
- enumerating content and functionality, 74–97
- hacker's methodology, 795–798
 - debug parameters, 798
 - default content, 797
 - enumerating identifiers, 797–798
 - hidden content, 796–797
 - public information resources, 796
 - tokens to sessions, 818
 - visible content, 795–796
- hidden content
 - brute-force techniques
 - discovering, 81–85
 - discovering, 80–93
 - inference from published content discovering, 85–89

- public information
 - discovering, 89–91
- web server leveraged for
 - discovering, 91–93
- hidden parameters, 96–97
- input entry points
 - HTTP headers, 100–101
 - out-of-band channels, 101
 - request parameters, 99
 - URL file paths, 98–99
- methodology, 114
- naming schemes, 85–86
 - brute-force exercise, 88
 - identifying, 87
- path traversal vulnerabilities, 371
- server-side
 - functionality identification, 106–110
 - technology identification, 101–106
- web application pages *versus*
 - functional paths, 93–96
- application servers. *See* web servers
- application service providers (ASPs), 656–657. *See also* ASP.NET; cloud computing
- attackers, 658–665
 - access, 658–660
 - deliberate backdoor scripts, 660–661
 - between web applications, 660–663
- financial services, 658
- organization, 658
- securing, 665–667
 - component segregation, 667
 - customer access, 665–666
 - customer functionality segregation, 666
- shared, 657–658
- threats, 657
- VPN, 659
- arbitrary input. *See* user input
- architecture. *See* tiered architectures
- Armstrong, Dave, 505
- The Art of Software Security Assessment* (Dowd & McDonald & Schuh), 634
- ASCII code, 67
- US-ASCII, 464
- Asirra puzzles, Microsoft, 612
- ASP.NET, 54, 103
- API methods
 - database, 721
- dynamic code execution, 722
- file access, 720
- OS command execution, 722–723
- sockets, 723
- URL redirection, 723
- user input, 718–719
- error messages, 628
- OS command injection via, 360–361
- redirection, 392
- security configuration, 723–724
- session interaction, 719–720
- stack traces, 617
- ViewState
 - attackers, 127
 - Base64 encoding, 125–126
 - Burp Suite, 126
 - client-side data transmission, 124–127
 - purpose, 125
 - security, 155
- ASPs. *See* application service providers
- .aspx file extension, 107
- Astely, Rick, 541
- attack payloads, XSS, 443–447
 - autocomplete, 446
 - escalating client-side, 447
 - escalation to other pages, 473–474
 - inducing actions, 445–446
 - Trojan injection, 444–445
 - trust relationship exploitation, 446–447
 - virtual defacement, 443–444
- attack surface
 - application logic flaws, 405
 - application mapping, 111
 - example, 112–113
 - hacker's methodology, application logic flaws, 842
 - hacker's methodology mapping, 800
- attackers. *See also specific attacks*
 - access controls, 266–278
 - types, 258–260
 - usernames and passwords, 275–276
- ASP.NET ViewState, 127
- ASPs, 658–665
 - access, 658–660
 - deliberate backdoor scripts, 660–661
 - between web applications, 660–663
- browser extensions casino component, 134
- CAPTCHA, 198–199
 - customized automation, 610–611
- client-side attacks, 13
- cloud computing, 14, 663–665
 - cloned systems, 664
 - tokens, 665
- cookie injection methods, 536–537
- credentials, 171
- defense mechanisms handling, 30–35
 - administrator alerting, 33–34
 - audit log maintenance, 31–32
 - errors, 30–31
 - reacting to, 34–35
- disabled elements, 132–133
- encoding and, 66–67
- forgotten password, 14
- format string vulnerabilities, 644
- HTTP header injection, 534–535
- intentions, 13
- login function, 164–165
- MS-SQL databases, 326–327
- multilayered privilege model, 283
- multistage login function, 188
- MySQL, 328
- network hosts, 561–562
- non-HTTP services, 562–563
- NULL bytes, 23–24
- opaque data, 124
- Oracle databases, 327
- other users, 431–432
- path traversal vulnerabilities
 - circumventing obstacles, 374–377
 - successful, 374
 - target locations, 370–371
- remote, 427
- session management, 20
- session token scripts, 217
- shared hosting, 658–665
 - access, 658–660
 - deliberate backdoor scripts, 660–661
 - between web applications, 660–663
- stored XSS steps, 438–439
- tiered architectures, 648–654
 - categories, 648–649
- tokens
 - encrypting, 232–233

- meaningful, 212
 - URL translation, 396–397
 - username, 168
 - web application security, 6
 - web browsers, 559–568
 - websites created by, 448–449
 - `XMLHttpRequest`, 529
 - XSS, 251
 - attribute delimiters, HTML
 - by passing filters, 461–462
 - attribute names, HTML
 - by passing filters, 461
 - attribute values, HTML
 - by passing filters, 462
 - audit logs
 - defense mechanisms handling
 - attackers, maintaining, 31–32
 - key events, 32
 - poorly protected, 32
 - value, 31
 - audit trail, 429
 - authentication. *See also*
 - access controls; session management
 - anomalies, 201
 - application logic flaws, 415–416
 - broken, 7
 - brute-force login function, 162–165
 - CAPTCHA, 198–199
 - credentials
 - incomplete validation, 180–181
 - insecure distribution, 184
 - insecure storage, 190–191
 - secret handling of, 192–193
 - strength, 192
 - transmission vulnerability, 169–171
 - validation, 193–195
 - CSRF, 507–508
 - as defense, 159
 - defense mechanisms handling
 - access with, 18–19
 - design flaws, 161–184
 - drop-down menus, 193
 - eavesdroppers, 169
 - hacker's methodology
 - application logic flaws, 811–813
 - credentials, autogenerated, 809–810
 - credentials, unsafe
 - distribution, 810–811
 - credentials, unsafe
 - transmission, 810
 - impersonation, 808–809
 - insecure web storage, 811
 - password guessing, 807
 - password quality, 806
 - password recovery, 807–808
 - “remember me” functions, 808
 - understanding, 805
 - username enumeration, 806–807
 - username uniqueness, 809
 - vulnerability exploitation for
 - unauthorized access, 813
 - HTML forms, 160–161
 - HTTP, 50–51
 - sessions avoided with, 208–209
 - impersonation, 178–180
 - hacker's methodology, 808–809
 - implementation flaws in, 185–191
 - information leakage
 - prevention, 195–196
 - logging, 201
 - login function
 - account suspension, 197–198
 - fail-open, 185–186, 194
 - multistage, 186–190, 194–195
 - verbose failure messages, 166–169
 - monitoring, 201
 - notifying, 201
 - passwords
 - change functionality, 171–172, 193
 - change functionality misuse, 199
 - forgotten functionality, 173–175
 - predictable initial, 183
 - weak, 161–162
 - problems with, 19
 - “remember me” functions, 175–176, 193
 - hacker's methodology, 808
 - security, 191–201
 - brute-force attack prevention, 196–199
 - subtleties, 195
 - smartcards, 206
 - standalone vulnerability
 - scanners, 778–779
 - technologies, 160–161
 - tokens, 160
 - usernames
 - enumeration, 166–169, 806–807
 - nonunique, 181–182
 - predictable, 182–183, 197
 - uniqueness, 809
 - XSS, 473–474
 - autocomplete
 - local privacy attacks, 552
 - XSS attack payloads, 446
 - automation. *See* customized automation
- ## B
- backdoor password, 178–179
 - source code, 708
 - backdoor scripts, deliberate, 660–661
 - back-end components. *See also*
 - file inclusion; operating system commands; path traversal vulnerabilities
 - access controls, 357
 - data transmission, 357
 - e-mail header injection, 398–399
 - HPI, 390
 - causes, 393–394
 - HPP, 394–395
 - server-side HTTP redirection, 390–392
 - exploiting, 391–392
 - SMTP injection, 397–402
 - flaws, 400–401
 - preventing, 402
 - SOAP injection, 386–388
 - banking application, 387–388
 - error messages, 388
 - finding and exploiting, 389
 - preventing, 27, 390
 - URL translation attacks, 396–397
 - back-end request injection, 841
 - backslash character, escaping with, 419
 - backtick character, encapsulating function of, 363
 - banking application
 - multistage function, 263
 - per-page tokens, 252–253
 - SOAP injection, 387–388
 - banner grabbing, 101
 - Base64 encoding, 69
 - ASP.NET ViewState, 125–126
 - basic authentication, 50–51
 - batch queries, MS-SQL
 - databases, 317
 - beating business limit,
 - application logic flaws, 416–417, 429

- BeEF, 565–566
 - bit flipper, Burp Intruder, 593
 - encrypting tokens, 228–231
 - black-box code review, 702–703
 - blacklist-based filters, 23–24
 - XSS, 451–452
 - blind SQL injection, 626
 - blocked characters, filters, 311–312
 - blog applications, input, 22
 - Boolean conditions, UNION operator, 329
 - Boolean flag, 107
 - boundary validation, input, 25–28, 313
 - breaking bank, application logic flaws, 414–416
 - browser extensions. *See also*
 - Flash; Java; Silverlight
 casino component, 133–134
 - attackers, 134
 Chrome, 750
 - client-side control of user input with, 133–153
 data transmission interception, 135–139
 - obstacles, 138–139
 - serialized data, 136–138
 debugger attached to, 151–152
 - decompiling, 139–150
 - bytecode, 139–141
 - bytecode obfuscation, 144–146
 - Java applets example, 146–150
 - JavaScript manipulating
 - original bytecode, 144
 - source code, 142–144
 Firefox, 750
 - hacker's methodology, 802–804
 - ActiveX controls, 804
 - debugger, 803–804
 - decompiling, 802–803
 - native client components, 153
 - same-origin policy, 525–527
 - Flash, 525–526
 - Java, 527
 - Silverlight, 526–527
 - targeting approaches, 135
 - technologies, 65
 - browsers. *See* web browsers
 - browsing history
 - JavaScript stealing, 560
 - local privacy attacks, 552
 - brute-force techniques
 - application mapping naming schemes exercise, 88
 - authentication security preventing, 196–199
 - hidden content, 81–85
 - login function, 162–165
 - passwords in wiki, 424
 - buffer overflow
 - detecting, 639–640
 - hacker's methodology, 837–838
 - heap overflows, 635–636
 - off-by-one vulnerabilities, 636–638
 - software, 687
 - source code, 709
 - stack overflows, 634–635
 - uncontrolled, 639
 - URL length, 639
 - bulk discount cheating,
 - application logic flaws, 418, 429
 - Burp Intruder, 82–84, 86
 - bit flipper, 593
 - encrypting tokens, 228–231
 - “character frobber,” 593
 - customized automation, 590–602
 - data harvesting, 598–600
 - enumerating identifiers, 594–597
 - fuzzing, 600–602
 - payloads
 - choosing, 592–594
 - positioning, 591–592
 - predictable tokens, 213–214
 - response analysis, 594
 - sniper attack, 592
 - Unicode encoding, 375
 - user agent strings, 100
 - Burp Proxy, 754–755
 - Burp Repeater, 473, 681, 766
 - Burp Scanner, 764–765
 - Burp Sequencer, 767
 - auto analyze setting, 223
 - token randomness testing, 219–221
 - Burp Spider, 74–76, 80
 - Burp Suite
 - AMF, 137
 - application mapping, 268
 - ASP.NET ViewState, 126
 - CA certificate, 758–759
 - “change request method” command, 474–475
 - Comparer, 167
 - Content Discovery, 88–89
 - DSer, 136–137
 - “request in browser,” 272–273
 - session-handling mechanisms, 603–609
 - cookie jar, 603–604
 - request macros, 604–606
 - session-handling rules, 606–609
 - session-handling tracer, 609
 - business limit, application logic flaws, 416–417, 429
 - business logic exploitation, 259
 - bytecode
 - decompiling browser
 - extensions, 139–141
 - JavaScript manipulation, 144
 - obfuscation, 144–146
 - downloading, 140
 - Flash, 141
 - Java, 141
 - Silverlight, 141
 - source code recompiling
 - within browser, 142–143
 - outside browser, 143
 - URL, 140
- ## C
- CA certificate, Burp Suite, 758–759
 - callbacks, function, 520
 - canonicalization
 - input, 28–29
 - web server software, 689–694
 - CAPTCHA
 - attackers, 198–199
 - customized automation, 610–611
 - authentication, 198–199
 - bugs, 610–611
 - customized automation, 610–612
 - attackers, 610–611
 - automatically solving, 611–612
 - humans solving, 612
 - drones, 612
 - Cascading Style Sheets (CSS)
 - dynamically evaluated styles, 459
 - font-family property, 518–519
 - injection, cross-domain data capture, 517–519
 - web functionality, 60–61
 - casino component, browser
 - extensions, 133–134
 - attackers, 134
 - CBC. *See* cipher block chaining
 - CGI query, 735–736
 - chaining
 - CBC
 - encrypting tokens, 227–233
 - PKC # 5 padding, 227–233

- XSS, 450–451
 - “change request method”
 - command, 474–475
 - “character frobber,” Burp
 - Intruder, 593
 - checked exceptions, 30
 - checkout, application logic flaws,
 - 410–411
 - CheckQuantity applet, 141
 - Chrome, 750
 - chrooted file system
 - path traversal vulnerabilities,
 - 380–381
 - UNIX, 381
 - cipher block chaining (CBC)
 - encrypting tokens, 227–233
 - PKC # 5 padding, 686–687
 - ciphertext, 224–226
 - .class files, 141
 - ClearedFunds element,
 - 387–388
 - cleartext, passwords, 190–191
 - clickjacking, 511. *See also* user
 - interface redress attacks
 - client components, native, 153
 - client-side
 - attacks, 13
 - data transmission, 118–127
 - ASP.NET ViewState,
 - 124–127
 - for developers, 118
 - hacker’s methodology, 801
 - hidden HTML forms, 118–120
 - HTTP cookies, 121
 - opaque data, 123–124
 - Referrer header, 122
 - security, 154–156
 - URL parameters, 121–122
 - hacker’s methodology, data
 - transmission, 801
 - HPP, 548–550
 - information disclosure leaks,
 - 629
 - injection, 531–550
 - SQL, 547–548
- JavaScript, validation with,
 - 130–131, 156
- security, 431–432
- session token hijacking,
 - 243–244
- SQL injection, 547–548
- SSL certification, 138
- user input controlled by, 117
 - browser extensions, 133–153
 - hacker’s methodology,
 - 801–802
 - HTML forms, 127–133
- validation myths, 155–156
- web functionality, 57–65
 - Ajax, 62–63, 384
 - browser extension
 - technologies, 65
 - CSS, 60–61
 - DOM, 62
 - forms, 58–60
 - HTML, 58
 - HTML5, 64–65
 - hyperlinks, 58
 - JavaScript, 61
 - JSON, 63
 - same-origin policy, 64
 - VBScript, 61
- XSS attack payloads escalating,
 - 447
- cloned systems, 664
- cloud computing
 - attackers, 14, 663–665
 - cloned systems, 664
 - tokens, 665
 - defense mechanism, 664
 - feature-first approach, 664–665
 - loss of control in, 663–664
 - management tool migration
 - to, 664
 - web applications, 5
 - web storage, 665
- CMS. *See* content management
 - system
- code browsing tools, 743
- code injection, 288
- code review. *See* source code,
 - review
- commands. *See* operating system
 - commands
- comments
 - MySQL, 303–304, 312
 - source code, 710–711
 - SQL, 312
- Comparer, Burp Suite, 167
- compiled applications. *See* native
 - client components
- concealed sequences, 213–215
- concurrent logins, 250
- conditional errors, SQL injection,
 - 320–322
- conjunctive queries filters, 350
 - LDAP injection, 352–353
- CONNECT method, 682, 755
- content
 - enumerating and functionality,
 - 74–97
 - hidden
 - brute-force techniques
 - discovering, 81–85
 - discovering, 80–93
 - hacker’s methodology,
 - application mapping,
 - 796–797
 - inference from published
 - content discovering,
 - 85–89
 - Nikto discovering, 93
 - public information
 - discovering, 89–91
 - user-directed spidering
 - discovering, 81–83
 - web server leveraged for
 - discovering, 91–93
 - Wikto discovering, 92–93
 - web server and default, 92,
 - 671–677
 - debug functionality, 671–672
 - hacker’s methodology, 847
 - JMX, 674–676
 - powerful functions, 673–674
 - sample functionality, 672–673
- Content Discovery, Burp Suite,
 - 88–89
- content management system
 - (CMS), 77
 - web servers, 92
- Content-Length header, 42
 - POST request, 581
- Content-Type header, 136, 138,
 - 476, 478, 525–526
- context-dependent, access
 - controls, 258
- Cookie header, 41, 47
- cookie injection
 - attacker methods, 536–537
 - session fixation, 537–540
- cookie jar, Burp Suite, 603–604
- cookies
 - arbitrary, 537
 - attributes, 47
 - domain restrictions, 245–247
 - hacker’s methodology, 820–821
 - HTTP, 19, 47
 - client-side data transmission,
 - 121
 - session management tokens,
 - 207–208, 234–236
 - HTTP header injection, 533
 - login function, 163
 - path restrictions, 247–248
 - persistent, 550
 - reflected XSS, 437–438
 - RememberMe, 407–408
 - “remember me” functions,
 - 175–176
 - ScreenName, 407–408

- session management, liberal
 - scope, 244–248
- XSS exploiting via, 475
- COPY method, 679
- count () function, 348
- credentials
 - attackers, 171
 - authentication vulnerability, 169–171
 - e-mail containing, 184
 - hacker's methodology, authentication
 - autogenerated, 809–810
 - unsafe distribution, 810–811
 - unsafe transmission, 810
 - incomplete validation, 180–181
 - insecure distribution, 184
 - insecure storage, 190–191
 - secret handling of, 192–193
 - strength, 192
 - validation, 193–195
 - web server and default, 670–671
 - hacker's methodology, 846
- cross-domain data capture, 515–516
- CSS injection, 517–519
- Firefox, 521
- HTML injection, 516–517
- JavaScript hijacking, 519–520
 - E4X, 523–524
 - function callbacks, 520
 - JSON, 521
 - preventing, 524
 - variable assignment, 522
- proxy services, 529–531
- cross-domain requests
 - JSON, 477
 - XMLHttpRequest, 528–529
 - XSS sending XML, 477–478
- /crossdomain.xml, 525–526
- cross-site request forgery (CSRF), 8, 244, 504–511
 - anti-CSRF tokens, 508–509, 516–517
 - XSS defeating, 510–511
 - authentication, 507–508
 - flaws
 - exploiting, 506–507
 - preventing, 508–510
 - real-world, 505
 - hacker's methodology, 820
 - session management, 251
- cross-site scripting (XSS), 8
 - attack payloads, 443–447
 - autocomplete, 446
 - escalating client-side, 447
 - escalation to other pages, 473–474
 - inducing actions, 445–446
 - Trojan injection, 444–445
 - trust relationship
 - exploitation, 446–447
 - virtual defacement, 443–444
 - attackers, 251
 - authentication, 473–474
 - chaining, 450–451
 - CSRF defeating anti-CSRF
 - tokens with, 510–511
 - database error messages, 620
 - defense, 28
 - delivery mechanisms, 447–451
 - in-band, 449–450
 - out-of-band, 450
 - DOM-based, 440–442
 - delivering, 448–449
 - finding and exploiting, 487–491
 - input validation, 497
 - output validation, 497–498
 - preventing, 496–498
 - reflected XSS converted into, 472–473
 - steps, 441
 - escaping, 420
 - exploits
 - cookies, 475
 - delivering, 473–481
 - JavaScript executed within
 - XML responses, 478–479
 - nonstandard request and response content, 476–479
 - Referrer header, 475–476
 - XML requests sent cross-domain, 477–478
 - filters
 - anti-, 452, 748
 - blacklist-based, 451–452
 - IE, 479–481
 - web browsers, 479–481
 - HTML tag pairs, 422
 - IE filter, 479–481
 - JavaScript, 436–438
 - non-HTTP services, 562–563
 - NULL bytes, 460
 - POST request changed to GET request, 474–475
 - prevalence, 432
 - preventing, 492–498
 - real-world, 442–443
 - reflected, 434–438
 - Apache, 442
 - cookies, 437–438
 - defensive filters, 455–456
 - delivering, 448–449
 - DOM XSS converted from, 472–473
 - exploiting, 435–438, 474
 - finding and exploiting, 452–481
 - hacker's methodology, 829–830
 - HTML limitations, 495–496
 - IE, 435
 - input insertion, 495
 - input validation, 492–493
 - length limits, 471–473
 - output validation, 493–495
 - preventing, 492–496
 - “remember me” function, 437
 - sanitizing filters, 468–471
 - signature-based filters, 455–456
 - steps, 436–437
 - stored XSS compared to, 439–440
 - user input testing, 453
 - user input testing to
 - introduce script, 454–455
- security evolution, 433
- session token vulnerabilities, 243–244
- source code, 704–705
- stored, 438–440
 - attacker steps, 438–439
 - delivering, 449–450
 - e-mail testing, 483–484
 - finding and exploiting, 481–487
 - HTML limitations, 495–496
 - input insertion, 495
 - input validation, 492–493
 - MySpace, 442–443, 446
 - output validation, 493–495
 - preventing, 492–496
 - reflected XSS compared to, 439–440
 - search function, 439
 - uploaded files testing, 484–487
- vulnerabilities
 - identifying, 451–452
 - low-risk, 451
 - varieties, 433–442
 - XSS Shell, 566
- cryptographic algorithms, 687
- CSRF. *See* cross-site request forgery

CSS. *See* Cascading Style Sheets
 Curl, 788
 custom development, web
 applications, 10
 custom encoding, path traversal
 vulnerabilities, 377–378
 customized automation
 barriers to, 602–612
 Burp Intruder, 590–602
 data harvesting attack,
 598–600
 enumerating identifiers
 attack, 594–597
 fuzzing attack, 600–602
 CAPTCHA puzzles, 610–612
 attackers, 610–611
 automatically solving,
 611–612
 humans solving, 612
 data harvesting, 572
 basic approach, 584–586
 Burp Intruder, 598–600
 causes, 583–584
 JAttack, 585–586
 uses, 584
 efficiency, 571
 enumerating identifiers,
 572–583
 basic approach, 574
 Burp Intruder, 594–597
 detecting hits, 574–576
 examples, 573
 HTTP status code, 574
 JAttack, 577–583
 Location header, 575
 response body, 575
 response length, 574–575
 scripting, 576–577
 Set-Cookie header, 575
 time delays, 575–576
 fuzzing, 572–573
 Burp Intruder, 600–602
 JAttack, 588–590
 objective, 586–587
 strings, 587
 session-handling mechanisms,
 602–609
 standalone vulnerability
 scanners, 780–781
 uses, 572–573
 Cygwin environment, 577

D

DAC. *See* discretionary access
 control
 data capture. *See* cross-domain
 data capture

data harvesting, 572
 basic approach, 584–586
 Burp Intruder, 598–600
 causes, 583–584
 JAttack, 585–586
 uses, 584
 data stores. *See also* Extensible
 Markup Language;
 Lightweight Directory
 Access Protocol; Structured
 Query Language
 accessing, 288–289
 NoSQL, 342–343
 privilege level, 287
 web applications relying on,
 287
 data transmission. *See also* user
 input
 back-end components, 357
 browser extensions
 intercepting, 135–139
 obstacles, 138–139
 serialized data, 136–138
 client-side, 118–127
 ASP.NET ViewState,
 124–127
 for developers, 118
 hacker's methodology, 801
 hidden HTML forms, 118–120
 HTTP cookies, 121
 opaque data, 123–124
 Referrer header, 122
 security, 154–156
 URL parameters, 121–122
 lazy load approach, 626
 opaque, 123–124
 attackers, 124
 database administrator (DBA),
 325–326
The Database Hacker's Handbook,
 326
 databases
 access
 ASP.NET API methods, 721
 Java API methods, 714–715
 Perl language API methods,
 737–738
 code components
 dangerous, 742
 SQL injection, 741–742
 error messages, 619–622
 encryption oracle, 620–622
 information disclosure,
 619–620
 XSS in, 620
 escalation attacks,
 319, 325–328
 fingerprinting, 303–304

information_schema,
 309–310
 MS-SQL
 attackers, 326–327
 automated exploitation, 330
 batch queries, 317
 default lockdown, 326–327
 error messages, 334–338
 out-of-band channels, 317
 syntax, 332–334
 WAITFOR command, 322–323
 Oracle
 attackers, 327
 11g, 318
 error messages, 334–338
 out-of-band channels,
 317–318
 syntax, 332–334
 time delays, 323–324
 UNION operator, 307–308
 searchable and sortable,
 321–322
 stored procedures, 339
 Davtest, 680
 DBA. *See* database administrator
 debuggers
 browser extensions attaching,
 151–152
 error messages, 425–426,
 618–619
 common, 619
 hacker's methodology,
 application mapping, 798
 hacker's methodology, browser
 extensions, 803–804
 Java, 151–152
 messages
 application logic flaws,
 424–426
 verbose, 425
 Silverlight, 152
 web server, 671–672
 declarative access controls,
 282–283
 decompiling
 browser extensions, 139–150
 bytecode, 139–141
 bytecode obfuscation,
 144–146
 Java applets example, 146–150
 JavaScript manipulating
 original bytecode, 144
 source code, 142–144
 hacker's methodology, browser
 extensions, 802–803
 Jad, Java, 148–150
 decryption algorithms, 650
 default content

- hacker's methodology,
 - application mapping, 797
 - web server, 671–677
 - hacker's methodology, 847
 - default credentials, web server, 670–671
 - hacker's methodology, 846
 - default lockdown, MS-SQL
 - databases, 326–327
 - defense in depth
 - SQL injection, 342
 - tiered architectures, 656
 - web server software, 696–697
 - defense mechanisms. *See also* security
 - access
 - authentication, 18–19
 - control, 20–21
 - session management, 19–20
 - attackers, 30–35
 - administrator alerting, 33–34
 - audit log maintenance, 31–32
 - errors, 30–31
 - reacting to, 34–35
 - elements, 17–18
 - input, 21–29
 - approaches to, 23–25
 - user access, 18–21
 - defensive filters, reflected XSS, 455–456
 - DELETE method, 679
 - DELETE statements, 297–298
 - deliberate backdoor scripts, 660–661
 - developers
 - application logic flaws, 429–430
 - client-side data transmission, 118
 - HTML encoding mistakes, 494–495
 - web applications security, 3
 - digest authentication, 50–51
 - directory listings, web servers, 677–679
 - Allaire JRun, 690–691
 - directory names, 105
 - disabled elements
 - attackers, 132–133
 - HTML forms, 131–133
 - discount cheating, application
 - logic flaws, 418, 429
 - discretionary access control (DAC), 282
 - disjunctive queries filters, 350
 - LDAP injection, 351
 - .dll files, 141
 - DNS rebinding, 563–564
 - DOCTYPE element, 384–385
 - document object model (DOM), 61
 - hacker's methodology, 849–850
 - JavaScript, 440
 - JavaScript API methods, 740
 - web functionality, 62
 - XSS, 440–442
 - delivering, 448–449
 - finding and exploiting, 487–491
 - input validation, 497
 - output validation, 497–498
 - preventing, 496–498
 - reflected XSS converted to, 472–473
 - steps, 441
 - DocumentRoot directive, 683
 - DOM. *See* document object model
 - domain restriction cookies, 245–247
 - DOMTracer, 488
 - dot character, script code
 - bypassing filters alternatives to, 466
 - “dot-dot-slash” sequence, 369. *See also* path traversal
 - vulnerabilities
 - Dowd, Mark, 634
 - downloading
 - bytecode, 140
 - encrypting tokens, 231–232
 - drop-down menus,
 - authentication, 193
 - DSer, Burp Suite, 136–137
 - Dump Servlet, Jetty, 672
 - dynamic code execution
 - ASP.NET API methods, 722
 - Java API methods, 715
 - OS command injection, 362
 - vulnerabilities, 366–367
 - Perl language API methods, 738
 - PHP API methods, 730–731
 - dynamically constructed strings, 466
- E**
- E4X. *See* ECMAScript for XML
 - Eagle, Chris, 634
 - eavesdroppers
 - authentication, 169
 - session tokens, 234
 - eBay, 505
 - ECB ciphers. *See* electronic cookbook ciphers
 - Echo Mirage, 139
 - ECMAScript for XML (E4X), 463
 - JavaScript hijacking, 523–524
 - edit parameter, 107
 - Edwards, Dean, 471
 - EJB. *See* Enterprise Java Bean
 - electronic cookbook ciphers (ECB ciphers), 224–226
 - e-mail
 - account activation URLs, 184
 - credentials sent in, 184
 - forged, 448
 - header injection, 398–399
 - stored XSS testing, 483–484
 - as username, 167, 196
 - encoding
 - Apache chunked overflow, 688
 - attackers and, 66–67
 - Base64, 69
 - ASP.NET ViewState, 125–126
 - custom, path traversal
 - vulnerabilities, 377–378
 - hex, 69–70
 - HTML, 68–69
 - developer mistakes, 494–495
 - script code bypassing filters, 468
 - Unicode, 67–68
 - Burp Intruder, 375
 - URL, 67
 - SQL injection, 300–301
 - truncating, 378
 - web server software, 689–694
 - encrypting
 - .NET, 686
 - “remember me” function, 177
 - tokens, 223–233
 - attackers, 232–233
 - Burp Intruder bit flipper, 228–231
 - CBC, 227–233
 - downloading, 231–232
 - ECB ciphers, 224–226
 - “reveal” encryption oracle, 232
 - encryption oracle
 - application logic flaws, 407–408
 - “remember me” function, 407
 - database error messages, 620–622
 - “reveal,” encrypting tokens, 232
 - Enterprise Java Bean (EJB), 53
 - enterprise resource planning software (ERP), 4
 - enumerating identifiers, 572–583
 - basic approach, 574
 - Burp Intruder, 594–597

- detecting hits, 574–576
- examples, 573
- hacker's methodology,
 - application mapping, 797–798
- HTTP status code, 574
- JAttack, 577–583
- Location header, 575
- response body, 575
- response length, 574–575
- scripting, 576–577
- Set-Cookie header, 575
- time delays, 575–576
- ERP. *See* enterprise resource planning software
- error messages
 - Apache, 628
 - ASP.NET, 628
 - database, 619–622
 - encryption oracle, 620–622
 - information disclosure, 619–620
 - databases, XSS in, 620
 - debugger, 425–426, 618–619
 - common, 619
 - dynamically generated, 434
 - engineering informative, 624–625
 - exploiting, 615–625
 - generic, 628
 - IE, 622
 - information disclosure, 615–625
 - generic, 628
 - Java, 628
 - keywords, 622
 - Microsoft IIS, 628
 - MS-SQL databases, 334–338
 - MySQL, 334–338
 - ODBC, 624
 - Oracle databases, 334–338
 - public information, 623
 - published content, 625
 - script, 616–617
 - search engines, 623
 - server, 619–622
 - SOAP injection, 388
 - source code, 623
 - SQL injection, 334–338
 - stack traces, 617–618
 - UNION operator, 306
 - VBScript, 616
 - verbose, 30–31, 624
- errors
 - conditional, SQL injection, 320–322
 - defense mechanisms handling
 - attackers and, 30–31

- unhandled, 30–31
- escaping
 - application logic flaws, 419–420
 - with backslash character, 419
 - JavaScript, script code
 - bypassing filters, 465–466
 - XSS, 420
- Etag string, 128–129
- eval function, 362, 722
 - script code bypassing filters
 - alternatives to, 466
- event handlers
 - HTML5, 458
 - script code in HTML with, 457–458
- Expires header, 42
- Extensible Markup Language (XML), 56. *See also* Simple Object Access Protocol; XML Path Language
- E4X, 463
- injection, 383–390
 - XXE, 384–386, 841
- interpreting, 387
- XSS exploits
 - JavaScript in, 478–479
 - sending cross-domain, 477–478
- Extract Grep function, 598

F

- fail-open login function, 185–186, 194
- failure messages, verbose, 166–169
- file extensions, 102–105
- file inclusion
 - hacker's methodology, 835–836
 - local, 382
 - remote, 381–382
 - flaw testing, 383
 - static resources, 382
 - vulnerabilities, 381–383
 - finding, 382–383
 - PHP, 381–382
- file path manipulation, 368–383. *See also* path traversal vulnerabilities
- filters
 - blocked characters, 311–312
 - conjunctive queries, 350
 - LDAP injection, 352–353
 - disjunctive queries, 350
 - LDAP injection, 351
 - exploiting defective, 313
 - HTML bypassing, 459–465
 - attribute delimiters, 461–462

- attribute names, 461
- attribute values, 462
- character sets, 464–465
- tag brackets, 462–464
- tag name, 460–461
- input, path traversal
 - vulnerabilities, 374–377
- LDAP, 350
- Oracle PL/SQL Exclusion List
 - bypassing, 692–694
- reflected XSS
 - defensive, 455–456
 - sanitizing, 468–471
 - signature-based, 456–457
 - sanitizing, reflected XSS, 468–471
- script code bypassing, 465–468
 - dot character alternatives, 466
 - dynamically constructed strings, 466
 - encoding, 468
 - eval function alternatives, 466
 - JavaScript escaping, 465–466
 - multiple technique
 - combination, 466–467
 - VBScript, 467
 - VBScript and JavaScript, 467–468
- simple match conditions, 350
- SQL injection bypassing, 311–313
- XSS
 - anti-, 452, 748
 - blacklist-based, 451–452
 - IE, 479–481
 - web browsers, 479–481
- financial services
 - application logic flaws, 412–416
 - ASPs, 658
- fingerprinting databases, SQL
 - injection, 303–304
- Firebug, 785
- Firefox, 459
 - browser extensions, 750
- cross-domain data capture, 521
- Firesheep tool, 234
- hacker's toolkit, 749–750
- Referrer header, 239
- Firesheep tool, Firefox, 234
- firewalls, 12
 - alerts, 33
 - WAFs, NULL bytes, 460
- first-order XSS. *See* reflected XSS
- 500 Internal Server Error, 49
- brute-force techniques, 85

- 503 Service Unavailable, 49
- Flash, 134–135
 - bytecode, 141
 - /crossdomain.xml, 525–526
 - LSOs, 553
 - same-origin policy, 525–526
 - serialized data, 137–138
- font-family property, 518–519
- forced browsing, application
 - logic flaws, 411
- forgotten password, 584
 - attackers using, 14
- format string vulnerabilities
 - attackers, 644
 - causes, 643
 - detecting, 644
 - hacker's methodology, 838
 - source code, 710
- forms
 - HTML, 58–59
 - authentication, 160–161
 - client-side control of user input with, 127–133
 - client-side data transmission with hidden, 118–120
 - disabled elements, 131–133
 - intercepting proxy modifying hidden, 119–120
 - length limits, 128–129
 - script-based validation, 129–131
 - web functionality, 58–60
- 400 Bad Request, 48
 - brute-force techniques, 84
- 401 Unauthorized, 48
 - brute-force techniques, 84–85
- 403 Forbidden, 49
 - brute-force techniques, 84–85
- 404 Not Found, 49
- 405 Method Not Allowed, 49
- 413 Request Entity Too Large, 49
- 414 Request URI Too Long, 49
- framebusting, UI redress
 - attacks, 514–515
- function callbacks, JavaScript hijacking, 520
- functional paths, web
 - application pages *versus*, 93–96
- functionality. *See* web functionality
- function-specific input
 - vulnerabilities, hacker's methodology, 836–841
- fuzzing, 572–573
 - Burp Intruder, 600–602
 - hacker's methodology, parameter, 824–827
 - integrated testing suites, 762–763
 - JAttack, 588–590
 - objective, 586–587
 - strings, 587
- G**
- general headers, 45
- generic error messages, 628
- GET method, 42
 - purpose, 264
- GET request, 40
 - XSS converting, 474–475
- getCurrentUserRoles
 - method, 261
- GIFAR files, 485–486
- Google, 89
 - Omitted Results, 90
 - querying, 90
- Google Translate (GT), 530–531
- Gray Hat Hacking* (Eagle & Harris & Harper & Ness), 634
- GT. *See* Google Translate
- H**
- hacker's methodology
 - access controls
 - insecure access, 823
 - limited access, 822–823
 - multiple accounts, 822
 - requirements, 821
 - analysis
 - attack surface mapping, 800
 - data entry points, 799
 - functionality, 798–799
 - technologies, 799–800
 - application logic flaws
 - attack surface, 842
 - incomplete input, 843
 - multistage functions, 842–843
 - transaction logic, 844
 - trust relationships, 844
 - application mapping, 795–798
 - debug parameters, 798
 - default content, 797
 - enumerating identifiers, 797–798
 - hidden content, 796–797
 - public information resources, 796
 - of tokens to sessions, 818
 - visible content, 795–796
- authentication
 - application logic flaws, 811–813
 - credentials, autogenerated, 809–810
 - credentials, unsafe
 - distribution, 810–811
 - credentials, unsafe transmission, 810
 - impersonation, 808–809
 - insecure web storage, 811
 - password guessing, 807
 - password quality, 806
 - password recovery, 807–808
 - “remember me” functions, 808
 - understanding, 805
 - username enumeration, 806–807
 - username uniqueness, 809
 - vulnerability exploitation for unauthorized access, 813
- back-end request injection, 841
- browser extensions, 802–804
 - ActiveX controls, 804
 - debugger, 803–804
 - decompiling, 802–803
- buffer overflow, 837–838
 - client-side
 - data transmission, 801
 - user input, 801–802
- cookie scope, 820–821
- CSRF, 820
- DOM, 849–850
- file inclusion, 835–836
- format string vulnerabilities, 838
 - fuzzing parameters, 824–827
 - guidelines, 793–794
- HTTP header injection, 830
- information leakage, 852
- input-based vulnerabilities, 824–836
 - function-specific, 836–841
- integer vulnerabilities, 838
- LDAP injection, 839–840
- local privacy attacks, 850–851
- miscellaneous checks, 849–852
- native software bugs, 837–838
- open redirection
 - vulnerabilities, 830–831
- OS command injection, 832–833
- path traversal vulnerabilities, 833–835
- reflected XSS, 829–830
- same-origin policy, 851–852

- script injection, 835
- session management
 - token insecure transmission, 817
 - token system log disclosure, 817–818
 - tokens tested for meaning, 815–816
 - tokens tested for
 - predictability, 816–817
 - understanding, 814–815
- sessions
 - fixation, 819
 - terminating, 818–819
- shared hosting, 845–846
- SMTP injection, 836–837
- SOAP injection, 839
- SQL injection, 827–829
- stored procedures, 831–832
- weak SSL ciphers, 851
- web servers, 846–849
 - dangerous HTTP methods, 847
 - default content, 847
 - default credentials, 846
 - native software bugs, 848
 - proxy server functionality, 847
 - virtual hosting, 847–848
 - WAFs, 848–849
- work areas, 791–793
- XPath injection, 840–841
- XXE injection, 841
- hacker's toolkit, 747
 - custom scripts, 786–789
 - Curl, 788
 - Netcat, 788–789
 - Stunnel, 789
 - Wget, 788
 - Firebug, 785
 - Hydra, 785–786
 - integrated testing suites, 751–773
 - components, 752–769
 - types, 751
 - Nikto, 785
 - web browsers, 748–750
 - Chrome, 750
 - Firefox, 749–750
 - IE, 748–749
 - Wikto, 785
- Hammad, Sherief, 322
- Harper, Allen, 634
- Harris, Shon, 634
- HEAD functions, 43
- HEAD method, 265
- heap overflows, 635–636
- Heasman, John, 634
- hex encoding, 69–70
- hidden content
 - discovering, 80–93
 - brute-force techniques, 81–85
 - inference from published content, 85–89
 - Nikto, 93
 - public information, 89–91
 - user-directed spidering, 81–83
 - web server leveraged for, 91–93
 - Wikto, 92–93
- hacker's methodology,
 - application mapping, 796–797
- hidden HTML form fields
 - client-side data transmission
 - with, 118–120
 - intercepting proxy modifying, 119–120
- hidden parameters, application
 - mapping, 96–97
- hijacking
 - JavaScript, 519–520
 - E4X, 523–524
 - function callbacks, 520
 - JSON, 521
 - preventing, 524
 - variable assignment, 522
- sessions, 436
- Holyfield, Brian, 138
- horizontal access controls, 258
- horizontal privilege escalation, 259, 416
- Host header, 41
- hosting. *See* shared hosting
- HP OpenView, 359
- HPI. *See* HTTP parameter injection
- HPP. *See* HTTP parameter pollution
- HTML. *See* hypertext markup language
- HTML5
 - Ajax, 487
 - event handlers, 458
 - local privacy attacks, 554
 - same-origin policy, 528–529
 - script pseudo-protocols, 458
 - web functionality, 64–65
- HTTP. *See* hypertext transfer protocol
- HTTP header injection
 - causes, 531–532
 - cookies, 533
 - exploiting, 532–535
 - attackers, 534–535
 - hacker's methodology, 830
 - HTTP response splitting, 534–535
 - input validation, 536
 - preventing, 536
- HTTP parameter injection (HPI), 390
 - causes, 393–394
 - HPP, 394–395
- HTTP parameter pollution (HPP)
 - client-side, 548–550
 - HPI, 394–395
- HTTPRECON, 102
- HTTPS, 49
 - integrated testing suites,
 - intercepting proxies, 755–758
 - login function, 170
 - man-in-the-middle attacks, 566–568
 - proxy servers, 50
 - session tokens, 234–236, 250
- HTTPWatch tool, IE, 748
- Hydra, 785–786
- hyperlinks, web functionality, 58
- hypertext markup language (HTML). *See also* HTML5
 - ActiveX controls modification, 557
 - bypassing filters, 459–465
 - attribute delimiters, 461–462
 - attribute names, 461
 - attribute values, 462
 - character sets, 464–465
 - tag brackets, 462–464
 - tag name, 460–461
 - encoding, 68–69
 - developer mistakes, 494–495
 - forms, 58–59
 - authentication, 160–161
 - client-side control of user input with, 127–133
 - client-side data transmission
 - with hidden, 118–120
 - disabled elements, 131–133
 - intercepting proxy
 - modifying hidden, 119–120
 - length limits, 128–129
 - script-based validation, 129–131
 - injection, cross-domain data
 - capture, 516–517
 - reflected XSS limiting, 495–496
 - script code introduced in
 - dynamically evaluated CSS styles, 459
 - event handlers, 457–458
 - script pseudo-protocols, 458
 - scripttags, 457

- stored XSS limiting, 495–496
 - tag pairs, XSS, 422
 - web functionality with, 58
 - hypertext transfer protocol (HTTP). *See also* HTTP header
 - injection
 - access controls testing, 278
 - authentication, 50–51
 - sessions avoided with, 208–209
 - benefits, 5
 - cookies, 19, 47
 - client-side data transmission, 121
 - session management tokens, 207–208, 234–236
 - fingerprinting, 102
 - hacker's methodology, web servers, 847
 - headers
 - application mapping, input entry points, 100–101
 - general, 45
 - request, 45–46
 - response, 46
 - security assumptions, 123
 - HPI, 390
 - causes, 393–394
 - HPP, 394–395
 - client-side, 548–550
 - man-in-the-middle attacks, 566–568
 - messages, 40–42
 - methods, 42–44
 - origins, 39
 - proxy servers, 49–50
 - requests, 40–41
 - dissecting, 107–108
 - input sources, 52
 - URL, 40, 42
 - responses, 41–42
 - splitting, 534–535
 - server-side redirection, 390–392
 - exploiting, 391–392
 - SSL and, 49
 - status codes, 48–49
 - enumerating identifiers, 574
 - TCP protocol, 40
 - hypothesis testing, statistical, 219–222
- I**
- ID field, 295
 - IDA Pro, 153
 - iDefense, 558
 - identifier-based functions
 - access controls, 261–262
 - application logs, 262
 - identifiers. *See* enumerating identifiers
 - IE. *See* Internet Explorer
 - IEWatch tool, 79, 748
 - If-Modified-Since, 128–129
 - If-None-Match, 128–129
 - iframe, 511–515
 - IIS, Microsoft
 - error messages, 628
 - ISAPI extensions, 688
 - path traversal vulnerabilities, 691–692
 - impersonation, authentication, 178–180
 - hacker's methodology, 808–809
 - in-band delivery, XSS, 449–450
 - inducing actions, 501
 - request forgery
 - CSRF, 8, 244, 251, 504–511
 - OSRF, 502–503
 - UI redress attacks, 508, 511–515
 - basic form, 511–513
 - framebusting, 514–515
 - mobile devices, 515
 - preventing, 515
 - variations, 513
 - XSS attack payloads, 445–446
 - inference
 - information disclosure, 626–627
 - search engines, 626
 - SQL injection, 319–324
 - infinite loops, 29
 - information disclosure
 - error messages, 615–625
 - generic, 628
 - inference, 626–627
 - leaks
 - client-side, 629
 - preventing, 627–629
 - protecting, 628–629
 - published content, 625
 - information leakage, 8
 - authentication preventing, 195–196
 - hacker's methodology, 852
 - information disclosure
 - client-side, 629
 - preventing, 627–629
 - information_schema, 309–310
 - initialization vector (IV), 685
 - injection
 - back-end request, 841
 - client-side, 531–550
 - SQL, 547–548
 - code, 288
 - cookie
 - attacker methods, 536–537
 - session fixation, 537–540
 - CSS, cross-domain data capture, 517–519
 - e-mail header, 398–399
 - HPI, 390
 - causes, 393–394
 - HTML, cross-domain data capture, 516–517
 - HTTP header
 - attackers exploiting, 534–535
 - causes, 531–532
 - cookies, 533
 - exploiting, 532–535
 - hacker's methodology, 830
 - HTTP response splitting, 534–535
 - input validation, 536
 - output validation, 536
 - preventing, 536
 - interpreted language, 288–290
 - LDAP, 349–354
 - conjunctive queries filters, 352–353
 - exploiting, 351–353
 - flaws, 353–354
 - hacker's methodology, 839–840
 - preventing, 354
 - vulnerabilities, 350–351
 - login function bypassed, 288–290
 - NoSQL, 342–344
 - MongoDB, 343–344
 - OS commands, 358–368
 - ASP.net, 360–361
 - dynamic code execution, 362
 - dynamic code execution, vulnerabilities, 366–367
 - flaws, 363–366
 - hacker's methodology, 832–833
 - metacharacters, 420
 - Perl language, 358–360
 - preventing, 367–368
 - shell metacharacters, 363, 365
 - source code, 708
 - spaces, 366
 - time delay, 363–364
 - script
 - hacker's methodology, 835
 - preventing vulnerabilities, 368
 - SMTP, 397–402
 - flaws, 400–401
 - hacker's methodology, 836–837
 - preventing, 402
 - SOAP, 386–388
 - banking application, 387–388

- error messages, 388
 - finding and exploiting, 389
 - hacker's methodology, 839
 - preventing, 27, 390
 - SQL, 7, 14
 - advanced exploitation, 314–324
 - API methods, 291
 - application logic flaws, 420–422
 - blind, 626
 - bugs, 298–302
 - client-side, 547–548
 - column name, 301–302
 - conditional errors, 320–322
 - database code components, 741–742
 - defense in depth, 342
 - DELETE statements, 297–298
 - double hyphen, 293
 - error messages, 334–338
 - exploitation tools, 328–331
 - filter bypassing, 311–313
 - fingerprinting databases, 303–304
 - hacker's methodology, 827–829
 - inference, 319–324
 - input validation
 - circumvented, 312
 - INSERT statements, 295–296
 - JavaScript errors, 299
 - numeric data, 299–301, 315–316
 - ORDER BY clause, 301–302
 - out-of-band channel, 316–319
 - parameterized queries, 339–341
 - preventing, 27, 338–342
 - query structure, 301–302
 - second-order, 313–314
 - SELECT statements, 294–295
 - source code, 705–706
 - string data, 298–299
 - syntax, 332–334
 - time delays, 322–324
 - UNION operator, 304–308
 - UNION operator data
 - extraction, 308–311
 - UPDATE statements, 296–297
 - URL encoding, 300–301
 - vulnerability exploitation, 292–294
 - Trojan, XSS attack payloads, 444–445
 - XML, 383–390
 - XXE, 384–386, 841
 - XPath, 344–349
 - blind, 347–348
 - flaws, 348–349
 - hacker's methodology, 840–841
 - informed, 346–347
 - preventing, 349
 - input. *See also* user input
 - “accept known good” approach, 24
 - application mapping, entry points for
 - HTTP headers, 100–101
 - out-of-band channels, 101
 - request parameters, 99
 - URL file paths, 98–99
 - blog applications, 22
 - boundary validation, 25–28, 313
 - canonicalization, 28–29
 - defense mechanisms, 21–29
 - approaches to, 23–25
 - filters, path traversal
 - vulnerabilities, 374–377
 - hacker's methodology,
 - application logic flaws and incomplete, 843
 - insertion, stored XSS, reflected
 - XSS eliminating dangerous, 495
 - multistep validation, 28–29
 - “reject known bad” approach, 23–24
 - safe data handling approach, 25
 - sanitization approach, 24–25
 - semantic checks, 25
 - validation, 21–22, 313
 - application logic flaws
 - invalidating, 420–422
 - circumventing, 312
 - DOM-based XSS, 497
 - HTTP header injection, 536
 - problems, 26
 - stored XSS, reflected XSS, 492–493
 - varieties, 21–23
 - input-based vulnerabilities,
 - hacker's methodology, 824–836
 - function-specific, 836–841
 - INSERT statements
 - SQL injection, 295–296
 - WHERE clause, 295
 - insurance, application logic
 - flaws, 412–413
 - integer vulnerabilities
 - causes, 640
 - detecting, 642–643
 - hacker's methodology, 838
 - overflows, 640–641
 - signedness errors, 641–642
 - source code, 709–710
 - integrated testing suites
 - fuzzing, 762–763
 - hacker's toolkit, 751–773
 - components, 752–769
 - types, 751
 - intercepting proxies
 - alternatives, 771–773
 - common features, 758–759
 - HTTPS, 755–758
 - web browser configuration, 752–755
 - manual request tools, 765–767
 - shared functions and utilities, 768–769
 - shared token analyzers, 767
 - Tamper Data, 772
 - TamperIE, 772–773
 - vulnerability scanners, 764–765
 - standalone, 773–784
 - web spidering, 760–762
 - work flow, 769–771
 - intercepting proxies
 - evolution, 751
 - integrated testing suites
 - alternatives, 771–773
 - common features, 758–759
 - HTTPS, 755–758
 - web browser configuration, 752–755
 - Internet. *See* World Wide Web
 - Internet Explorer (IE), 239, 459
 - anti-XSS filters, 748
 - error messages, 622
 - HTTPWatch tool, 748
 - IEWatch tool, 79, 748
 - reflected XSS, 435
 - TamperIE, 772–773
 - userData, 554
 - web application hacker's toolkit, 748–749
 - XSS filter, 479–481
 - Internet forums, public
 - information, 91
 - interpreted language injection, 288–290
 - IP address availability, 100
 - IV. *See* initialization vector
- ## J
- Jad, Java, 141
 - decompiling, 148–150
 - .jad files, 148–150
 - .jar files, 141
 - JAttack

- data harvesting, 585–586
- enumerating identifiers, 577–583
- extract function, 598
- fuzzing, 588–590
- strength, 590
- Java
 - API methods
 - database access, 714–715
 - dynamic code execution, 715
 - file access, 713
 - OS command execution, 715–716
 - potentially dangerous, 713–716
 - sockets, 716
 - URL redirection, 716
 - applets, 134
 - decompiling browser extensions, 146–150
 - bytecode, 141
 - debuggers, 151–152
 - error messages, 628
 - Jad, 141
 - decompiling, 148–150
 - same-origin policy, 527
 - security configuring, 716–717
 - serialized data, 136–137
 - session interaction, 712–713
 - terminology, 53
 - tiered architectures, 648
 - user input, 711–712
 - API methods, 712
 - web container, 53
 - web functionality, 53–54
- Java Servlet, 53
- Java Virtual Machine (JVM), 134
 - web server software
 - vulnerabilities, 690
- `java.io.File`, 713
- `java.net.Socket`, 716
- JavaScript
 - browsing history stolen with, 560
 - client-side, validation with, 130–131, 156
 - decompiling browser
 - extensions, original
 - bytecode manipulation, 144
 - DOM, 440
 - DOM-based API methods, 740
 - escaping, script code
 - bypassing filters, 465–466
 - hijacking, 519–520
 - E4X, 523–524
 - function callbacks, 520
 - JSON, 521
 - preventing, 524
 - variable assignment, 522
 - `$js` function, 344
 - length limits, 471
 - logging keystrokes, 560
 - open redirection
 - vulnerabilities, 546
 - port scanning, 561, 566
 - script code bypassing filters
 - using VBScript and, 467–468
 - SQL injection, errors in, 299
 - third-party applications
 - currently used, 560–561
 - web functionality, 61
 - XSS, 436–438
 - XSS exploits executing, in XML responses, 478–479
 - JavaScript Object Notation (JSON)
 - cross-domain requests, 477
 - JavaScript hijacking, 521
 - web functionality, 63
 - JavaSnoop, 151–152
 - JBoss Application Server, 674–676
 - Jetty, 218
 - Dump Servlet, 672
 - Jitko worm, 530–531
 - `$js` function, JavaScript, 344
 - JMX, 674–676
 - JRun, Allaire, 690–691
 - JSON. *See* JavaScript Object Notation
 - `.jsp` file extension, 107
 - JSwat, 151–152
 - JVM. *See* Java Virtual Machine

K

 - Kamkar, Samy, 219
 - keystrokes, logging, 560
 - Klein, Amit, 248

L

 - LAMP server, 650–651, 666
 - languages. *See* interpreted language
 - lazy load approach, data transmission, 626
 - LDAP. *See* Lightweight Directory Access Protocol
 - leaks. *See* information leakage
 - length limits
 - JavaScript, 471
 - reflected XSS, 471–473
 - Ley, Jim, 444
 - Lightweight Directory Access Protocol (LDAP)
 - filters, 350
 - injection, 349–354
 - conjunctive queries filters, 352–353
 - disjunctive queries filters, 351
 - exploiting, 351–353
 - flaws, 353–354
 - hacker's methodology, 839–840
 - preventing, 354
 - vulnerabilities, 350–351
 - uses, 349–350
 - Linder, Felix, 634
 - Litchfield, David, 320, 327, 693
 - `LOAD_FILE` command, 328
 - local file inclusion, 382
 - tiered architectures, 652–654
 - local privacy attacks
 - autocomplete, 552
 - browsing history, 552
 - Flash LSOs, 553
 - hacker's methodology, 850–851
 - HTML5, 554
 - IE userData, 554
 - persistent cookies, 550
 - preventing, 554–555
 - Silverlight Isolated Storage, 553
 - testing, 550
 - Local Shared Objects (LSOs), 553
 - Location header, 531–532
 - enumerating identifiers, 575
 - location-based access controls, 266
 - logging keystrokes, 560
 - logic. *See* application logic flaws
 - login function, 18–19, 160
 - account suspension, 197–198
 - application logic flaws, 426–427
 - race conditions, 427
 - attackers, 164–165
 - authentication
 - brute-forcible, 162–165
 - verbose failure messages, 166–169
 - concurrent, 250
 - cookies, 163
 - fail-open, 185–186, 194
 - HTTPS, 170
 - injection bypassing, 288–290
 - multistage, 186–190, 194–195
 - attackers, 188
 - common myth, 187
 - purpose, 186–187
 - random questions, 189–190, 194–195
 - secondary challenge, 173, 200

- secret questions, 189
- session management, 206
 - tokens, 539–540
- timing differences, 168–169
- username enumeration, 166–169
- logout function, session management, 242, 250
- logs. *See* system log disclosure, session tokens
- LSOs. *See* Local Shared Objects

M

- macros, request, 604–606
- `magic_quotes-gpc` directive, 734
- `mail()` command, 398–399
- mail services. *See* e-mail; SMTP injection
- man-in-the-middle attacks, 566–568
- manual request tools, integrated testing suites, 765–767
- mapping. *See* application mapping
- Mavituna, Ferruh, 566
- McDonald, John, 634
- meaningful token attackers, 212
- memory management, web server software, 687–689
- metacharacters, OS command injection, 420. *See also* shell metacharacters
- Microsoft. *See also* Internet Explorer
 - Asirra puzzles, 612
 - IIS
 - error messages, 628
 - ISAPI extensions, 688
 - path traversal vulnerabilities, 691–692
 - security, 431–432
 - SiteLock Active Template Library, 559
- mobile devices
 - applications, 4
 - UI redress attacks, 515
- `mod_isapi`, Apache, 688
- `mod_proxy`, Apache, 688
- MongoDB, NoSQL injection, 343–344
- `MOVE` method, 679–680
- MS-SQL databases
 - attackers, 326–327
 - automated exploitation, 330
 - batch queries, 317
 - default lockdown, 326–327
 - error messages, 334–338
 - out-of-band channels, 317
 - syntax, 332–334
 - `WAITFOR` command, 322–323
- multistage functions
 - access controls, 262–263
 - testing, 271–273
- banking application, 263
- hacker's methodology, application logic flaws, 842–843
- login, 186–190, 194
 - attackers, 188
 - common myth, 187
 - purpose, 186–187
 - random questions, 189–190, 194–195
- multistep validation, input, 28–29
- MySpace, stored XSS, 442–443, 446
- MySQL
 - attackers, 328
 - comments, 303–304, 312
 - double hyphen, 293
 - error messages, 334–338
 - out-of-band channels, 319
 - path traversal vulnerabilities, 651
 - sleep function, 323
 - syntax, 332–334
 - tiered architectures extracting, 650–652
 - UDFs, 328

N

- naming schemes
 - application mapping, 85–86
 - brute-force exercise, 88
 - identifying, 87
 - static resources, 87
- native client components, 153
- native compiled applications
 - buffer overflow, 634–640
 - examples, 633
 - format string vulnerabilities, 643–644
 - integer vulnerabilities, 640–643
 - testing for, 633–634
- native software bugs
 - hacker's methodology, 837–838
 - web servers, 848
 - source code, 709–710
- NBFS. *See* .NET Binary Format for SOAP
- negative price method, 120
- Ness, Jonathan, 634

- .NET
 - encryption, 686
 - padding oracle, 685–687
- .NET Binary Format for SOAP (NBFS), 138
- Netcat, 788–789
- NETGEAR router, 562
- network disclosure, session tokens, 234–237
- network hosts, attackers, 561–562
- network perimeter, web application security and new, 12–14
- `nextPayload` method, 578
- NGSSoftware, 640
- Nikto
 - hacker's toolkit, 785
 - hidden content, 93
 - maximizing effectiveness, 797
- non-HTTP services, 562–563
- NoSQL
 - advantages, 343
 - data stores, 342–343
 - injection, 342–344
 - MongoDB, 343–344
- `notNetgear` function, 562
- `nslookup` command, 365
- NTLM protocol, 50
- NULL bytes
 - attackers, 23–24
 - WAFs, 460
 - XSS, 460
- NULL value, 306–307
- numeric data
 - limits, 417
 - SQL injection into, 299–301, 315–316

O

- obfuscation
 - bytecode, decompiling browser extensions, 144–146
 - custom schemes, 109
- OCR. *See* optical character recognition
- ODBC. *See* open database connectivity
- off-by-one vulnerabilities, 636–638
- OllyDbg, 153
- Omitted Results, Google, 90
- 100 Continue, 48
- on-site request forgery (OSRF), 502–503
- onsubmit attributes, 130
- opaque data
 - attackers, 124

- client-side data transmission, 123–124
 - open database connectivity (ODBC), 624
 - open redirection vulnerabilities
 - causes, 540–541
 - finding and exploiting, 542–546
 - hacker's methodology, 830–831
 - JavaScript, 546
 - preventing, 546–547
 - rickrolling attacks, 541
 - source code, 707–708
 - URLs, 542
 - absolute prefix, 545–546
 - blocking absolute, 544–545
 - user input, 543–544
 - OpenLDAP, 352
 - operating system commands (OS commands)
 - ASP.NET API methods, 722–723
 - injection, 358–368
 - ASP.net, 360–361
 - dynamic code execution, 362
 - dynamic code execution, vulnerabilities, 366–367
 - flaws, 363–366
 - hacker's methodology, 832–833
 - metacharacters, 420
 - Perl language, 358–360
 - preventing, 367–368
 - shell metacharacters, 363, 365
 - source code, 708
 - spaces, 366
 - time delay, 363–364
 - Java API methods, 715–716
 - Perl language API methods, 738
 - PHP API methods, 731
 - optical character recognition (OCR), 611
 - OPTIONS functions, 43
 - OPTIONS method, 679–680
 - OPTIONS request, 528
 - Oracle
 - databases
 - attackers, 327
 - 11g, 318
 - error messages, 334–338
 - out-of-band channels, 317–318
 - syntax, 332–334
 - time delays, 323–324
 - UNION operator, 307–308
 - PL/SQL Exclusion List, 676–677
 - web server software filter bypass, 692–694
 - web server, 676–677
 - The Oracle Hacker's Handbook* (Litchfield), 693
 - oracles. *See* encryption oracle
 - ORDER BY clause, 295
 - SQL injection, 301–302
 - Origin headers, 528–529
 - OS commands. *See* operating system commands
 - OSRF. *See* on-site request forgery
 - other user attackers, 431–432
 - out-of-band channels
 - application mapping, input entry points, 101
 - MS-SQL databases, 317
 - MySQL, 319
 - Oracle databases, 317–318
 - SQL injection, 316–319
 - unavailable, 319
 - out-of-band delivery, XSS, 450
 - output validation
 - DOM-based XSS, 497–498
 - HTTP header injection, 536
 - stored XSS, reflected XSS, 493–495
- P**
- padding oracle
 - attack, 626
 - .NET, 685–687
 - pageid parameter, 598
 - parameter-based access controls, 265–266
 - parameterized queries
 - provisos, 341
 - SQL injection, 339–341
 - parameters
 - application mapping, input entry points, 99
 - hidden, application mapping, 96–97
 - URL, client-side data transmission, 121–122
 - parseResponse method, 585, 589
 - passive scanning, 764–765
 - passwords
 - access controls attackers harvesting, 275–276
 - backdoor, 178–179
 - source code, 708
 - brute-force techniques for wiki, 424
 - change functionality, 171–172, 193
 - application logic flaws, 409–410
 - misuse, 199
 - username, 172
 - cleartext storage, 190–191
 - forgotten, 14, 584
 - functionality, 173–175
 - guessing, 160
 - techniques, 163–164
 - hacker's methodology, authentication
 - guessing, 807
 - quality, 806
 - recovery function, 807–808
 - hints, 174, 200
 - predictable initial, 183
 - real-world, 163
 - recovery
 - challenges, 173–174
 - hacker's methodology, authentication, 807–808
 - hints, 200
 - misuse, 199–200
 - secondary challenge, 200
 - time-limited URLs, 174–175
 - requirements, 192
 - resetting, 175
 - system-generated, 192
 - truncated, 180–181
 - weak, 161–162
 - path restriction cookies, 247–248
 - path traversal vulnerabilities
 - Apple iDisk Server, 690
 - application mapping, 371
 - attackers
 - circumventing obstacles, 374–377
 - successful, 374
 - targets, 370–371
 - causes, 368–369
 - chrooted file system, 380–381
 - custom encoding, 377–378
 - detecting, 372–374
 - initial testing, 372
 - exploiting, 379
 - finding, 370–378
 - hacker's methodology, 833–835
 - input filters, 374–377
 - Microsoft IIS, 691–692
 - MySQL, 651
 - preventing, 379–381
 - source code, 706–707
 - subtlety, 370
 - UNIX compared to Windows, 374

- user input, 379–380
 - Payment Card Industry (PCI), 7
 - Perl language
 - API methods
 - database access, 737–738
 - dynamic code execution, 738
 - file access, 737
 - OS command execution, 738
 - potentially dangerous, 736–739
 - sockets, 739
 - URL redirection, 738
 - eval function, 362
 - OS command injection via, 358–360
 - security configuration, 739–740
 - session interaction, 736
 - shell metacharacters, 360
 - user input, 735–736
 - per-page tokens, 252–253
 - persistent cookies, 550
 - phishing attacks, 541, 707
 - PHP
 - API methods
 - database access, 729–730
 - dynamic code execution, 730–731
 - file access, 727–729
 - OS command execution, 731
 - potentially dangerous, 727–732
 - sockets, 732
 - URL redirection, 731–732
 - eval function, 362
 - file inclusion vulnerabilities, 381–382
 - mail() command, 398–399
 - safe mode, 666
 - security configuration, 732–735
 - magic_quotes-gpc directive, 734
 - register_globals directive, 733
 - safe_mode directive, 733–734
 - session interaction, 727
 - tiered architectures, 653–654
 - user input, 724–727
 - web functionality, 54–55
 - .php file extension, 108
 - phpinfo.php, 672
 - ping command, 364
 - PKC # 5 padding, 685
 - CBC, 686–687
 - Plain Old Java Object (POJO), 53
 - PL/SQL Exclusion List, Oracle, 676–677
 - web server software filter bypass, 692–694
 - POJO. *See* Plain Old Java Object
 - port scanning, Java Script, 561, 566
 - POST method, 43, 192
 - purpose, 264
 - POST request
 - Content-Length header, 581
 - XSS converting, 474–475
 - PostgreSQL, 323
 - Pragma header, 42
 - predictable initial passwords, 183–184
 - predictable tokens, 213–223
 - Burp Intruder, 213–214
 - concealed sequences, 213–215
 - time dependency, 215–217
 - weak random number generation, 218–219
 - testing quality, 219–223
 - preg_replace function, 730
 - prepared statements, 339–341
 - privacy attacks. *See* local privacy attacks
 - privilege
 - data stores, 287
 - DBA, 325–326
 - escalation
 - horizontal, 258, 416
 - vertical, 258, 416
 - multilayered model
 - access controls security, 280–283
 - attackers, 283
 - privs field, 295
 - proceeding to checkout, application logic flaws, 410–411
 - programmatic access controls, 282
 - PROPFIND method, 679
 - proxy history records, 769–771
 - proxy servers. *See also* intercepting proxies
 - hacker's methodology, web servers, 847
 - hidden HTML form
 - modification with intercepting, 119–120
 - HTTP, 49–50
 - HTTPS, 50
 - invisible, 138
 - web servers as, 682–683
 - proxy services
 - cross-domain data capture, 529–531
 - GT, 530–531
 - Jitko worm, 530–531
 - public information
 - error messages, 623
 - hacker's methodology, application mapping, 796
 - hidden content discovery with, 89–91
 - Internet forums, 91
 - search engines for, 89
 - web archives for, 89–90
 - published content
 - error messages, 625
 - hidden content discovery with inference from, 85–89
 - information disclosure, 625
 - PUT functions, 43
 - PUT method, 679–680
- ## Q
- quantity parameter, restricting, 128
 - queries
 - CGI, 735–736
 - conjunctive filters, 350
 - LDAP injection, 352–353
 - disjunctive filters, 350
 - LDAP injection, 351
 - parameterized
 - provisos, 341
 - SQL injection, 339–341
 - search engines, 90
 - SELECT queries, UNION operator, 304–305
 - structure, SQL injection, 301–302
- ## R
- race conditions, 427
 - Rails 1.0, 55
 - RBAC. *See* role-based access control
 - real-world
 - application logic flaws, 406–407
 - CSRF flaw, 505
 - passwords, 163
 - XSS, 442–443
 - recompiling, source code to
 - bytecode
 - within browser, 142–143
 - outside browser, 143

- redirection attacks. *See* open redirection vulnerabilities
 - referer-based access controls, 266
 - Referrer header, 41–42
 - client-side data transmission, 122
 - Firefox, 239
 - XSS exploiting via, 475–476
 - reflected XSS, 434–438
 - Apache, 442
 - cookies, 437–438
 - delivering, 448–449
 - DOM XSS converted from, 472–473
 - exploiting, 435–438, 474
 - filters
 - defensive, 455–456
 - sanitizing, 468–471
 - signature-based, 455–456
 - finding and exploiting, 452–481
 - hacker's methodology, 829–830
 - IE, 435
 - length limits, 471–473
 - preventing, 492–496
 - HTML limitations, 495–496
 - input insertion, 495
 - input validation, 492–493
 - output validation, 493–495
 - “remember me” function, 437
 - steps, 436–437
 - stored XSS compared to, 439–440
 - user input testing, 453
 - script introduction, 454–455
 - register_globals directive, 733
 - “reject known bad” approach, input, 23–24
 - RememberMe cookie, 407–408
 - “remember me” functions
 - application logic flaws, encryption oracle, 407
 - authentication, 175–176, 193
 - hacker's methodology, 808
 - cookies, 175–176
 - encrypting, 177
 - reflected XSS, 437
 - remote attackers, 427
 - remote black-box testing, 427
 - remote file inclusion, 381–382
 - flaw testing, 383
 - remoting, 70
 - representational state transfer (REST), URLs, 44–45
 - spidering, 74–75
 - request forgery
 - CSRF, 8, 244, 504–511
 - anti-CSRF tokens, 508–509, 516–517
 - authentication, 507–508
 - exploiting flaws, 506–507
 - hacker's methodology, 820
 - preventing flaws, 508–510
 - real-world flaws, 505
 - session management, 251
 - XSS defeating anti-CSRF tokens, 510–511
 - OSRF, 502–503
 - request headers, 45–46
 - “request in browser,” Burp Suite, 272–273
 - request macros, Burp Suite, 604–606
 - response headers, 46
 - REST. *See* representational state transfer
 - reverse strokejacking, 560
 - rickrolling attacks, 541
 - Rios, Billy, 485
 - robots.txt, 74
 - role-based access control (RBAC), 282
 - rolling your own insurance, application logic flaws, 412–413
 - Ruby on Rails (Ruby), 55
 - WEBrick, 690
- S**
- safe data handling approach, input, 25
 - “safe for scripting” registration, ActiveX controls, 555–557
 - safe_mode directive, 733–734
 - same-origin policy, 524–525
 - browser extensions, 525–527
 - Flash, 525–526
 - Java, 527
 - Silverlight, 526–527
 - hacker's methodology, 851–852
 - HTML5, 528–529
 - web functionality, 64
 - sanitization approach, input, 24–25
 - sanitizing filters, 468–471
 - scanning. *See* vulnerability scanners
 - Schuh, Justin, 634
 - ScreenName cookie, 407–408
 - scripts. *See also* cross-site scripting
 - deliberate backdoor, 660–661
 - enumerating identifiers, 576–577
 - error messages, 616–617
 - hacker's toolkit custom, 786–789
 - Curl, 788
 - Netcat, 788–789
 - Stunnel, 789
 - Wget, 788
 - HTML form validation, 129–131
 - injection
 - hacker's methodology, 835
 - preventing vulnerabilities, 368
 - reflected XSS user input testing to introduce, 454–455
 - session token attacker, 217
 - script code
 - bypassing filters, 465–468
 - dot character alternatives, 466
 - dynamically constructed strings, 466
 - encoding, 468
 - eval function alternatives, 466
 - JavaScript escaping, 465–466
 - multiple technique combination, 466–467
 - VBScript, 467
 - VBScript and JavaScript, 467–468
 - HTML introducing
 - dynamically evaluated CSS styles, 459
 - event handlers, 457–458
 - script pseudo-protocols, 458
 - scripttags, 457
 - script pseudo-protocols, 458
 - search engines
 - error messages, 623
 - inference, 626
 - public information, 89
 - queries, 90
 - search function
 - application logic flaws, 422–424, 429
 - stored XSS, 439
 - SEARCH method, 679
 - secondary challenge
 - login function, 173, 200
 - password recovery, 200
 - second-order SQL injection, 313–314

- second-order XSS. *See* stored XSS
- secret questions, login function, 189
- Secure Socket Layer (SSL)
 - client-side certification, 138
 - communication protection, 192
 - hacker's methodology check
 - for weak ciphers, 851
 - HTTP tunneled over, 49
 - security, 7–8
 - session tokens, 233
 - vulnerabilities of, 8
- security. *See also* defense mechanisms
 - access controls, 278–283
 - best practices, 279–280
 - central component approach, 280
 - multilayered privilege model, 280–283
 - pitfalls, 278–279
- application logic flaws, 428
- ASP.NET
 - configuration, 723–724
 - ViewState, 155
- ASPs, 665–667
 - component segregation, 667
 - customer access, 665–666
 - customer functionality segregation, 666
- authentication, 191–201
 - brute-force attack prevention, 196–199
 - subtleties, 195
- client-side, 431–432
- client-side data transmission, 154–156
 - logging and alerting, 156
 - validation, 155
- evolution, 432
- hardening, 695–696
- HTTP headers and
 - assumptions with, 123
- Java configuration, 716–717
- media focus on, 432
- Microsoft, 431–432
- myths, 433
- PCI standards, 7
- Perl language configuration, 739–740
- PHP configuration, 732–735
 - magic_quotes-gpc directive, 734
 - register_globals directive, 733
 - safe_mode directive, 733–734
- questions, 650
- reputation, 1
- session management, 248–254
- shared hosting, 665–667
 - component segregation, 667
 - customer access, 665–666
 - customer functionality segregation, 666
- SSL, 7–8
- tiered architectures, 654–656
- time and resources impacting, 11
- token generation, 210
- underdeveloped awareness of, 10
- web application, 1, 6–15
 - attackers, 6
 - developer understanding, 3
 - future, 14–15
 - key factors, 10–12
 - new network perimeter for, 12–14
 - user input threatening, 9–10
 - vulnerabilities, 7–8
- web server
 - configuration, 684
 - software, 695–697
- website evolution and, 2
- XSS, evolution, 433
- SELECT NULL value, UNION operator, 306–307
- SELECT queries, UNION operator, 304–305
- SELECT statements
 - SQL injection, 294–295
 - WHERE clause, 321
- self-registration, usernames, 182, 196
- semantic checks, input, 25
- semicolon character, batch function, 363
- serialization, 70
- serialized data
 - browser extensions
 - intercepting data transmission, handling, 136–138
 - Java, 136–137
 - Flash, 137–138
 - Silverlight, 138
- server error messages, 619–622
- Server header, 42
- server-executable files, 382
- servers. *See* web servers
- server-side
 - API redirection, 392
 - functionality
 - application mapping identifying, 106–110
- ASP.NET, 54, 103
- dissecting requests, 107–108
- Java, 53–54
- PHP, 54–55
- Ruby on Rails, 55
- SQL, 55–56
- web application behavior extrapolation, 109–110
- web application behavior isolation, 110
- web services, 56–57
- XML, 56
- HTTP redirection, 390–392
 - exploiting, 391–392
- technologies
 - application mapping identifying, 101–106
 - banner grabbing, 101
 - directory names, 105
 - file extensions, 102–105
 - HTTP fingerprinting, 102
 - session tokens, 105
 - third-party code components, 105
- sessions
 - ASP.NET, 719–720
 - fixation
 - cookie injection, 537–540
 - finding and exploiting, 539–540
 - preventing, 540
 - steps, 537–538
 - hacker's methodology
 - fixation, 819
 - terminating, 818–819
 - hacker's methodology, application mapping, tokens to, 818
 - hijacking, 436
 - HTTP authentication
 - alternative to, 208–209
 - Java, 712–713
 - Perl language, 736
 - PHP, 727
 - standalone vulnerability
 - scanners handling, 778–779
 - state information managed
 - without, 209
 - termination, 241–243
 - reactive, 253–254
 - web functionality, 66
- session management. *See also* access controls
 - alerts, 253
 - application logic flaws, 429
 - attackers, 20
 - cookies, liberal scope, 244–248

- CSRF, 251
- defense mechanisms handling
 - access with, 19–20
- duration, 241–243
- hacker's methodology
 - token insecure transmission, 817
 - token system log disclosure, 817–818
 - tokens tested for meaning, 815–816
 - tokens tested for
 - predictability, 816–817
 - understanding, 814–815
- logging, 253
- login function, 206
- logout function, 242, 250
- monitoring, 253
- security, 248–254
- state information, 206–209
- tokens
 - algorithm generating, 249
 - attacker scripts, 217
 - client-side exposure to
 - hijacking of, 243–244
 - concealed sequences, 213–215
 - eavesdroppers, 234
 - encrypting, 223–233
 - HTTP cookies, 207–208, 234–236
 - HTTPS, 234–236, 250
 - life cycle protection, 250–253
 - login function, 539–540
 - meaningful, 210–212
 - network disclosure, 234–237
 - per-page, 252–253
 - predictable, 213–223
 - server-side technology, 105
 - SSL, 233
 - strength, 248–249
 - system log disclosure, 237–239
 - time dependency, 215–217
 - transmitting, 538
 - URL transmission, 250
 - in URLs, 237–238
 - vulnerable mapping of, 240–241
 - weak random number
 - generation, 218–219
 - weak random number
 - quality testing, 219–223
 - weakness in generating, 210–233
 - weakness in handling, 233–248
 - XSS vulnerabilities, 243–244
 - uses, 205
- session riding. *See* request forgery
- session-handling mechanisms
 - Burp Suite
 - cookie jar, 603–604
 - request macros, 604–606
 - session-handling rules, 606–609
 - session-handling tracer, 609
 - supporting, 603–609
 - customized automation, 602–609
 - session-handling rules, 606–609
 - session-handling tracer, 609
 - SessionID parameter, 590
 - Set-Cookie header, 42, 47, 242, 244–245, 531
 - enumerating identifiers, 575
 - setString method, 340
- shared hosting, 656–657. *See also*
 - cloud computing
 - attackers, 658–665
 - access, 658–660
 - deliberate backdoor scripts, 660–661
 - between web applications, 660–663
 - hacker's methodology, 845–846
 - securing, 665–667
 - component segregation, 667
 - customer access, 665–666
 - customer functionality
 - segregation, 666
 - threats, 657
 - virtual hosting, 657
- shared token analyzers,
 - integrated testing suites, 767
- shared usernames, 181
- shell metacharacters, 359–360
 - application logic flaws, 419
 - OS command injection, 363, 365
 - Perl language, 360
 - types, 363
- The Shellcoder's Handbook* (Anley & Heasman & Linder), 634
- Shift-JIS character set, 464–465
- shutdown command, 315
- signature-based filters, reflected
 - XSS, 456–457
- signedness errors, 641–642
- Silverlight, 135
 - bytecode, 141
 - debuggers, 152
 - Isolated Storage, 553
 - same-origin policy, 526–527
 - serialized data, 138
 - Spy, 152
- simple match conditions filter, 350
- Simple Object Access Protocol (SOAP), 57
 - functions, 386
 - injection, 386–388
 - banking application, 387–388
 - error messages, 388
 - finding and exploiting, 389
 - hacker's methodology, 839
 - preventing, 27, 390
 - NBFS, 138
- site map records, 769–771
- SiteLock Active Template
 - Library, Microsoft, 559
- sleep function, MySQL, 323
- smartcards, authentication, 206
- SMTP injection, 397–402
 - flaws, 400–401
 - hacker's methodology, 836–837
 - preventing, 402
- sniper attack, Burp Intruder, 592
- SOAP. *See* Simple Object Access Protocol
- sockets
 - ASP.NET API methods, 723
 - Java, 716
 - Perl language API methods, 739
 - PHP API methods, 732
- source code
 - application logic flaws, 428
 - backdoor password, 708
 - browsing, 743
 - buffer overflow, 709
 - bytecode recompiling
 - within browser, 142–143
 - outside browser, 143
 - comments, 710–711
 - decompiling browser
 - extensions, 142–144
 - error messages, 623
 - format string vulnerabilities, 710
 - integer vulnerabilities, 709–710
 - native software bugs, 709–710
 - open redirection
 - vulnerabilities, 707–708
 - OS command injection, 708
 - path traversal vulnerabilities, 706–707
 - review
 - approaches, 702–704
 - black-box *versus* white-box, 702–703
 - methodology, 703–704
 - situations, 701

- signatures of common
 - vulnerabilities, 704–711
- SQL injection, 705–706
- XSS, 704–705
- spidering
 - REST URLs, 74–75
 - user-directed, 77–80
 - benefits, 77
 - hidden content discovery
 - with, 81–83
 - web compared to, 79
 - web, 74–77
 - authentication, 76
 - integrated testing suites, 760–762
 - user-directed spidering
 - compared to, 79
- SQL. *See* Structured Query Language
- SQLMap, 322
- sql-shell option, 330–331
- SQLzoo.net, 292
- SSL. *See* Secure Socket Layer
- stack overflows, 634–635
- stack traces
 - ASP.NET, 617
 - error messages, 617–618
- standalone vulnerability
 - scanners, 773–784
 - automated *versus* user-directed, 784
 - customized automation, 780–781
 - dangerous effects, 779
 - individuating functionality, 779–780
 - limitations, 776–777
 - products, 781–782
 - technical challenges, 778–781
 - authentication and session handling, 778–779
 - using, 783–784
 - vulnerabilities detected, 774–776
 - vulnerabilities undetected, 775
- state information
 - session management, 206–209
 - without sessions, 209
 - web functionality, 66
- static resources
 - access controls, 263–264
 - account testing, 277
 - file inclusion, 382
 - naming schemes, 87
- static tokens, 240
- statistical hypothesis testing, 219–222
- status codes, HTTP, 48–49
- enumerating identifiers, 574
- storage. *See* web storage, cloud
- computing
- stored procedures
 - databases, 339
 - hacker's methodology, 831–832
- stored XSS, 438–440
 - attacker steps, 438–439
 - delivering, 449–450
 - e-mail testing, 483–484
 - finding and exploiting, 481–487
 - MySpace, 442–443, 446
 - preventing, 492–496
 - HTML limitations, 495–496
 - input insertion, 495
 - input validation, 492–493
 - output validation, 493–495
 - reflected XSS compared to, 439–440
 - search function, 439
 - uploaded files testing, 484–487
 - Ajax, 486–487
 - GIFAR files, 485–486
- string data
 - dynamically constructed,
 - script code bypassing filters, 466
 - manipulation, 316
 - SQL injection into, 298–299
- string-length() function, 348
- strncpy function, 642
- strokejacking, 511. *See also* user interface redress attacks
- reverse, 560
- Structured Query Language (SQL)
 - client-side injection, 547–548
 - comments, 312
 - injection, 7, 14
 - advanced exploitation, 314–324
 - API methods, 291
 - application logic flaws, 420–422
 - blind, 626
 - bugs, 298–302
 - client-side, 547–548
 - column name, 301–302
 - conditional errors, 320–322
 - database code components, 741–742
 - defense in depth, 342
 - DELETE statements, 297–298
 - double hyphen, 293
 - error messages, 334–338
 - exploitation tools, 328–331
 - filter bypassing, 311–313
 - fingerprinting databases, 303–304
 - hacker's methodology, 827–829
 - inference, 319–324
 - input validation
 - circumvented, 312
 - INSERT statements, 295–296
 - JavaScript errors, 299
 - numeric data, 299–301, 315–316
 - ORDER BY clause, 301–302
 - out-of-band channel, 316–319
 - parameterized queries, 339–341
 - preventing, 27, 338–342
 - query structure, 301–302
 - second-order, 313–314
 - SELECT statements, 294–295
 - source code, 705–706
 - string data, 298–299
 - syntax, 332–334
 - time delays, 322–324
 - UNION operator, 304–308
 - UNION operator data
 - extraction, 308–311
 - UPDATE statements, 296–297
 - URL encoding, 300–301
 - vulnerability exploitation, 292–294
 - web functionality, 55–56
 - structured tokens, 210–212
 - Stunnel, 789
 - SUBSTR(ING) functions, 324
 - suspension of account, 197–198
 - .swf files, 141
 - syntactic validation, 25
 - system log disclosure
 - hacker's methodology, session management, 817–818
 - session tokens, 237–239
 - vulnerabilities, 238

T

- tag brackets, HTML bypassing filters, 462–464
- tag name, HTML bypassing filters, 460–461
- scripttags, 457
- Tamper Data, 772
- TamperIE, 772–773
- TCP protocol, HTTP using, 40
- testing. *See* account testing; hacker's methodology; hacker's toolkit; statistical hypothesis testing
- third-party applications, 560–561

- 301 Moved Permanently, 48
 - 302 Found, 48
 - brute-force techniques, 84
 - 304 Not Modified, 48
 - tiered architectures, 647
 - attacks, 648–654
 - categories, 648–649
 - component segregation, 655–656
 - defense in depth, 656
 - Java, 648
 - layers, 648
 - PHP, 653–654
 - securing, 654–656
 - subverting, 650–654
 - decryption algorithms, 650
 - local file inclusion executing commands, 652–654
 - MySQL extraction, 650–652
 - trust relationships, 649–650
 - access, 649
 - minimize, 654–655
 - time
 - delays
 - enumerating identifiers, 575–576
 - Oracle databases, 323–324
 - OS command injection, 363–364
 - SQL injection, 322–324
 - session token generation, 215–217
 - time of check, time of use flaw (TOCTOU flaw), 505
 - TOCTOU flaw. *See* time of check, time of use flaw
 - tokens
 - anti-CSRF, 508–509
 - XSS defeating, 510–511
 - authentication, 160
 - Burp Sequencer testing
 - randomness of, 219–221
 - cloud computing attackers, 665
 - encrypting, 223–233
 - attackers, 232–233
 - Burp Intruder bit flipper, 228–231
 - CBC, 227–233
 - downloading, 231–232
 - ECB ciphers, 224–226
 - “reveal” encryption oracle, 232
 - generating strong, 248–249
 - hacker’s methodology,
 - application mapping, sessions to, 818
 - hacker’s methodology, session management
 - insecure transmission, 817
 - system log disclosure, 817–818
 - tested for meaning, 815–816
 - tested for predictability, 816–817
 - per-page, 252–253
 - session management
 - algorithm generating, 249
 - attacker scripts, 217
 - client-side exposure to hijacking of, 243–244
 - concealed sequences, 213–215
 - eavesdroppers, 234
 - encrypting, 223–233
 - HTTP cookies for, 207–208, 234–236
 - HTTPS, 234–236, 250
 - life cycle protection, 250–253
 - login function, 539–540
 - meaningful, 210–212
 - network disclosure, 234–237
 - per-page, 252–253
 - predictable, 213–223
 - security, generation of, 210
 - server-side technologies, 105
 - strength, 248–249
 - system log disclosure, 237–239
 - transmitting, 538
 - URL transmission, 250
 - in URLs, 237–238
 - vulnerable mapping of, 240–241
 - weakness in generating, 210–233
 - weakness in handling, 233–248
 - XSS vulnerabilities, 243–244
 - shared analyzers, integrated testing suites, 767
 - SSL, 233
 - static, 240
 - structured, 210–212
 - time dependency, 215–217
 - weak random number generation, 218–219
 - weak random number quality testing, 219–223
 - TRACE functions, 43
 - transaction logic, 844
 - Trojan injection, XSS attack payloads, 444–445
 - trust relationships
 - hacker’s methodology, application logic flaws, 844
 - tiered architectures
 - access, 649
 - exploiting, 649–650
 - minimize, 654–655
 - XSS attack payloads exploiting, 446–447
 - try-catch blocks, 30
 - 200 OK, 48
 - 201 Created, 48
- ## U
- UDFs. *See* user-defined functions
 - UI redress attacks. *See* user interface redress attacks
 - uid parameter, 584, 590
 - unhandled errors, 30–31
 - Unicode encoding, 67–68
 - Burp Intruder, 375
 - uniform resource identifier (URI), 44
 - open redirection
 - vulnerabilities, absolute prefix, 545–546
 - uniform resource locator (URL)
 - account activation, 184
 - application mapping, input entry points, 98–99
 - buffer overflow and length of, 639
 - bytecode, 140
 - encoding, 67
 - SQL injection, 300–301
 - truncating, 378
 - format, 44
 - HTTP requests, 40, 44
 - open redirection
 - vulnerabilities, 542
 - absolute prefix, 545–546
 - blocking absolute, 544–545
 - parameters, client-side data transmission, 121–122
 - passwords recovery with time-limited, 174–175
 - redirection
 - ASP.NET API methods, 723
 - Java API methods, 716
 - Perl language API methods, 738
 - PHP API methods, 731–732
 - REST, 44–45
 - spidering, 74–75
 - session tokens, 237–238, 250
 - translation attacks, 396–397
 - UNION operator
 - Boolean conditions, 329
 - error messages, 306
 - NULL value, 306–307
 - Oracle databases, 307–308

- provisos, 305–306
- SELECT NULL value, 306–307
- SELECT queries, 304–305
- SQL injection, 304–308
 - data extraction, 308–311
- UNIX
 - chrooted file system, 381
 - Windows path traversal
 - vulnerabilities compared to, 374
- UPDATE statements, 296–297
- uploaded files, stored XSS
 - testing, 484–487
 - Ajax, 486–487
 - GIFAR files, 485–486
- URI. *See* uniform resource identifier
- URL. *See* uniform resource locator
- US-ASCII, 464
- user access. *See* access
- user input. *See also* input
 - ASP.NET API methods for, 718–719
 - client-side controls, 117
 - browser extensions, 133–153
 - hacker's methodology, 801–802
 - HTML forms, 127–133
 - Java, 711–712
 - API methods, 712
 - open redirection
 - vulnerabilities, 543–544
 - path traversal vulnerabilities, 379–380
 - Perl language, 735–736
 - PHP, 724–727
 - reflected XSS testing, 453
 - script introduction, 454–455
 - web application security
 - threatened by, 9–10
- user interface redress attacks (UI redress attacks), 508, 511–515
 - basic form, 511–513
 - framebusting, 514–515
 - mobile devices, 515
 - preventing, 515
 - variations, 513
- User-Agent header, 41, 52
 - targeting, 100
- userData, IE, 554
- user-defined functions (UDFs), 328
- user-directed spidering, 77–80
 - benefits, 77
 - hidden content discovery with, 81–83

- web spidering compared to, 79
- _username buffer, 635–637
- usernames
 - access controls attackers
 - harvesting, 275–276
 - attackers, 168
 - e-mail address, 167, 196
 - enumeration, 166–169
 - hacker's methodology,
 - authentication
 - enumerating, 806–807
 - uniqueness, 809
 - nonunique, 181–182
 - password change functionality, 172
 - predictable, 182–183, 197
 - self-registration, 182, 196
 - shared, 181
 - sources, 169
 - system-generated, 192
 - UTF-7, 464
 - UTF-16, 464–465
 - UTL-HTTP package, 317–318

V

- ValidateForm function, 130
- VALUES clause, 295–296
- variable assignment, JavaScript
 - hijacking, 522
- VBScript
 - error messages, 616
 - script code bypassing filters, 467
 - JavaScript with, 467–468
 - web functionality, 61
- vendor patches, web servers, 695
- verbose debugger messages, 425
- verbose error message, 30–31, 624
- verbose failure messages, 166–169
- vertical access controls, 258
- vertical privilege escalation, 258, 416
- ViewState, ASP.NET
 - attackers, 127
 - Base64 encoding, 125–126
 - Burp Suite, 126
 - client-side data transmission, 124–127
 - purpose, 125
 - security, 155
- virtual defacement, XSS attack
 - payloads, 443–444
- virtual hosting
 - Apache, 683

- hacker's methodology, web
 - servers, 847–848
- shared hosting, 657
- web servers misconfigured, 683
- virtual machines (VMs), 145
 - sandbox, 153
- virtual private network (VPN), 659
- VMs. *See* virtual machines
- VPN. *See* virtual private network
- vulnerability scanners
 - integrated testing suites, 764–765
 - standalone, 773–784
 - standalone, 773–784
 - automated *versus* user-directed, 784
 - customized automation, 780–781
 - dangerous effects, 779
 - individuating functionality, 779–780
 - limitations, 776–777
 - products, 781–782
 - technical challenges, 778–781
 - using, 783–784
 - vulnerabilities detected, 774–776
 - vulnerabilities undetected, 775

W

- WAFs. *See* web application firewalls
- WAITFOR command, MS-SQL, 322–323
- WAR files, 673–676
- warez, distributing, 2
- WayBack Machine, 89
- WCF. *See* Windows Communication Foundation
- weak passwords, 161–162
- web 2.0, 14
 - vulnerabilities, 65
- web application firewalls (WAFs)
 - bypassing, 698
 - hacker's methodology, web
 - servers, 848–849
 - NULL bytes, 460
 - web servers, 697–698
- web applications. *See also*
 - hacker's methodology;
 - hacker's toolkit
 administrative functions in, 35–36

- ASP attackers between, 660–663
- behavior
 - extrapolating, 109–110
 - isolating, 110
- benefits, 5–6
- business, 4
- cloud computing, 5
- custom development, 10
- data store reliance of, 287
- deceptive simplicity, 10–11
- evolution, 2–3
- framework flaws, 685–687
- functions, 4–5
 - increasing demands on, 12
- managing, 35–36
- overextended, 11–12
- pages, functional paths *versus*, 93–96
- security, 1, 6–15
 - attackers, 6
 - developer understanding, 3
 - future, 14–15
 - key factors, 10–12
 - new network perimeter for, 12–14
 - user input threatening, 9–10
 - vulnerabilities, 7–8
- shared hosting attackers
 - between, 660–663
- technologies developing, 6
- third-party, 560–561
- threats to, 3
 - rapidly evolving, 11
- XPath subverting logic of, 345–346
- web archives, public
 - information, 89–90
- web browsers. *See also* browser extensions; Firefox; Internet Explorer
 - attackers, 559–568
 - browsing history, 552
 - bugs, 563
 - capabilities, 5–6
 - DNS rebinding, 563–564
 - exploitation frameworks, 564–566
 - BeEF, 565–566
 - XSS Shell, 566
 - hacker's toolkit, 748–750
 - Chrome, 750
 - Firefox, 749–750
 - IE, 748–749
 - integrated testing suites, intercepting proxies configuring, 752–755
 - XSS filters, 479–481
 - web container, Java, 53
 - web functionality
 - client-side, 57–65
 - Ajax, 62–63, 384
 - browser extension technologies, 65
 - CSS, 60–61
 - DOM, 62
 - forms, 58–60
 - HTML, 58
 - HTML5, 64–65
 - hyperlinks, 58
 - JavaScript, 61
 - JSON, 63
 - same-origin policy, 64
 - VBScript, 61
 - server-side, 51–57, 103, 106–110
 - ASP.NET, 54, 103
 - Java, 53–54
 - PHP, 54–55
 - Ruby on Rails, 55
 - SQL, 55–56
 - web services, 56–57
 - XML, 56
 - sessions, 66
 - state information, 66
 - web servers, 669–670
 - CMS, 92
 - configuration
 - security, 684
 - vulnerabilities, 670–684
 - default content, 92, 671–677
 - debug functionality, 671–672
 - hacker's methodology, 847
 - JMX, 674–676
 - powerful functions, 673–674
 - sample functionality, 672–673
 - default credentials, 670–671
 - hacker's methodology, 846
 - directory listing, 677–679
 - Allaire JRun, 690–691
 - flaws, 694
 - hacker's methodology, 846–849
 - dangerous HTTP methods, 847
 - default content, 847
 - default credentials, 846
 - native software bugs, 848
 - proxy server functionality, 847
 - virtual hosting, 847–848
 - WAFs, 848–849
 - hidden content discovery
 - leveraging, 91–93
 - JBoss Application Server, 674–676
 - misconfigured virtual hosting, 683
 - Oracle, 676–677
 - as proxy servers, 682–683
 - software
 - Allaire JRun, 690–691
 - Apple iDisk Server, 690
 - defense in depth, 696–697
 - encoding and
 - canonicalization, 689–694
 - JVM, 690
 - memory management, 687–689
 - Microsoft IIS path traversal vulnerabilities, 691–692
 - Oracle PL/SQL Exclusion
 - List filter bypass, 692–694
 - resources, 694
 - Ruby WEBrick, 690
 - securing, 695–697
 - security hardening, 695–696
 - vendor patches, 695
 - vulnerabilities, 684–697
 - vulnerabilities, 91–92
 - WAFs, 697–698
 - WebDAV methods, 679–681
 - web services, 56–57
 - Web Services Description Language (WSDL), 57
 - web spidering, 74–77
 - authentication, 76
 - integrated testing suites, 760–762
 - user-directed spidering
 - compared to, 79
 - web storage
 - cloud computing, 665
 - hacker's methodology, authentication insecure, 811
 - Web-based Distributed Authoring and Versioning (WebDAV)
 - overflows, 689
 - web server methods, 679–681
 - WebDAV. *See* Web-based Distributed Authoring and Versioning
 - WEBrick, Ruby, 690
 - websites
 - attacker-created, 448–449
 - evolution, 51
 - security and evolution of, 2
 - web.xml file, 716–717
 - Wget, 788