INDEX

Symbols & Numbers

- * character, 255, 263
- 8N1 UART configuration, *158*
- 802.11 protocols, *288*
- 802.11w, 289

A

- AAAA records, 138–139
- A-ASSOCIATE abort message, 96
- A-ASSOCIATE accept message, 96
- A-ASSOCIATE reject message, 96
- A-ASSOCIATE request message, *96*
  - C-ECHO requests dissector, building, 101–105
  - defining structure, 112–113
  - overview, 96
  - parsing responses, 113–114
  - structure of, 101–102
  - writing contexts, 110–111
- ABP (Activation by Personalization), *324*, 326–327, 330–331
- Abstract Syntax, *111*
- access bits, 251–*252*
- access controls, testing, 49–50
- access points (APs), *287*
  - cracking WPA Enterprise, 304–305
  - cracking WPA/WPA2, 299–300
  - general discussion, 287–288
  - overview, 299
- access port, *60*
- account privileges, testing, 51
- ACK spoofing, 331
- Activation by Personalization (ABP), *324*, 326–327, 330–331
- active reconnaissance, *43*, 43–45
- active RFID technologies, *241*–242
- active spidering, *48*
- Activities, in Android apps, *361*
- activity tracking systems, *385*
- Adafruit CircuitPython
  - setting up, 318–319
  - writing LoRa sniffer, 320–322
- Adafruit FT232H Breakout, 401–402
- adb (Android Debug Bridge), 360, 402
- `adb logcat`, 367–368
- `adb pull` command, 361
- `AddPortMapping` command, 125–126, 130
- addressing layer, UPnP, *119*
- address search macro, *204*
- address space layout randomization, *345*
- admin credentials, Netgear D6000, 213–214
- ADV_IND PDU type, 271
- ADV_NONCONN_IND PDU type, 271
- advanced persistent threat (APT) attacks, *26*
- adversaries, 6
- `aes128_cmac` function, 325
- AES 128-bit keys, 323, 325, 326
- AFI (Application Family Identifier), *244*
- `-afre` argument, 135–136
- aftermarket security, 5
- Aircrack-ng, 289, 300–301, 402
- Aireplay-ng, 290
- Airmon-ng, 289–290, 297–298
- Airodump-ng, 290, 301
- Akamai, 118
- Akerun Smart Lock Robot app for iOS, 357
- alarms, jamming wireless, 375–379
- Alfa Atheros AWUS036NHA, 402
- altering RFID tags, 255–256
- Amazon S3 buckets, 209–210
- Amcrest IP camera, 147–152
- amplification attacks, 94
- analysis phase, network protocol inspections, 92–93
- Andriesse, Dennis, 218
- Android apps. *See also* smart treadmills, attacking
  - binary reversing, 362–363
  - dynamic analysis, 363–367
  - extracting APK, 361
  - MIFARE, attacking with, 256–257
  - network traffic, intercepting and examining, 367
  - overview, 360
  - preparing test environment, 360–361
  - security controls, 339–341

- Android apps, *continued*
  - side-channel leaks, 367–368
  - static analysis, 361–362
  - threats to, 337–338
- Android Debug Bridge (adb), 360, 402
- *AndroidManifest.xml* file, 361, 365–366
- Android Open Source Project (AOSP), *386–387*
- Android Package (APK) files, 360
  - abusing local file managers to install, 391–393
  - binary reversing, 362–363
  - extracting, 361
  - static analysis, 361–362
- Android Studio IDE, 360
- Android Verified Boot, *341*
- Android Virtual Device (AVD) Manager, 360
- Animas OneTouch Ping insulin pump security issue, 11–12
- Announcing phase
  - ippserver, 137–138
  - mDNS, 132
- antennas, RFID, 242–243
- Anti-collision loop command, 258–259
- anti-hacking laws, 12–13
- `"ANY"` query, 134–135
- AOSP (Android Open Source Project), *386–387*
- APK files. *See* Android Package (APK) files
- Apktool, 361, 402
- Apkx, 361
- app directory, inspecting, 366
- AppEUI (application identifier), 325
- AppKey, *323*, 325
- application analysis approach, 210–211
- Application Context, A-ASSOCIATE request message, 110–111
- application entity title, 102
- Application Family Identifier (AFI), *244*
- application identifier (AppEUI), 325
- application layer, LoRaWAN, 324
- application logs, inspecting, 351–352
- application mapping, 48
- application server, 50, *309*
- application signatures, 340
- application-specific attacks, LoRaWAN, 331
- AppNonce, 326
- AppSKey, *323*, 326
- APs. *See* access points (APs)
- APT (advanced persistent threat) attacks, 26
- ARC (Automatic Reference Counting), *346*
- Arduino, 402
  - coding target program in, 172–174
  - flashing and running program, 174–180
- Arduino Integrated Development Environment (IDE), *170*, 180
  - Heltec LoRa 32 development board, setting up, 309–314
  - setting up, 170–172
  - setting up controller-peripheral I^{2}C bus architecture, 201–202
- Arduino SAM boards, 171
- Arduino Uno microcontroller, 198–202
- A records, 138–140, 144
- A-RELEASE request message, 96
- A-RELEASE response message, 96
- `ar` parameter, 261–262
- asset-centric threat model, *30*
- association attacks, *291*–295
- Atheros AR7 devices, 225–226
- Atlasis, Antonios, 133
- `at` parameter, 261–262
- attacker-centric threat model, *31*
- attack trees, *28*–29
- Attify Badge, 403
- A-type messages, 96–97, 99
- authentication
  - BLE, 282–283
  - MIFARE cards, 258–259
  - mobile apps, 340–341
  - mutual, 94
  - nested authentication attack, *374*
  - web application testing, 49
- authorization, testing, 49–50
- AutoIP, 119
- automatic device discovery, *145*
- Automatic Reference Counting (ARC), *346*
- automating
  - firmware analysis, 215–216
  - RFID attacks using Scripting Engine, 263–264
  - static analysis of application source code, 346, 361
- AVD (Android Virtual Device) Manager, 360

B

- backdoor agent, *223*–228
- Baksmali, 368–369
- banner grabbing, *44*
- base station, *372*
- battery drain attacks, *42*
- baud rate, 162–163, *317*
- `b` command, 349
- beacon frames, *293*
- beacons, *270*
- Beagle I2C/SPI Protocol Analyzer, 403
- bed of nails process, *164*
- Bettercap, *276*
  - discovering devices and listing characteristics, 276–278
  - hacking BLE, 279–285
  - overview, 403
- BinaryCookieReader, 350–351, 403
- binary emulation, 216–217

- binary reversing
  - InsecureBankV2 app, 362–363
  - OWASP iGoat app, 355–356
- *bin/passwd* binary file, 213
- Binwalk, 212, 219, 403
- `binwalk` Nmap command, 70–71
- BIOS security testing, 41
- bit-flipping attacks, 327–330
- Black Magic Probe, 165
- Black Pill (STM32F103C8T6)
  - boot mode, selecting, 174–175
  - coding target program in Arduino, 172–174
  - connecting to computer, 179–180
  - connecting USB to serial adapter, 178
  - debugging target, 181–188
  - flashing and running Arduino program, 174–180
  - overview, 169–170, 412
  - UART pins, identifying with logic analyzer, 176–177
  - uploading Arduino program, 175–176
- BladeRF, 403
- BLE (Bluetooth Low Energy). *See* Bluetooth Low Energy (BLE)
- BLE CTF Infinity
  - authentication, 282–283
  - examining characteristics and descriptors, 281–282
  - getting started, 279–280
  - overview, 278
  - setting up, 279
  - spoofing MAC address, 283–285
- `ble.enum` command, 284
- `ble.show` command, 276
- `ble.write` command, 278
- BlinkM LED, 198–202, 404
- Bluetooth Low Energy (BLE), *269*. *See also* BLE CTF Infinity
  - BlueZ, 273–274
  - configuring interfaces, 274–275
  - discovering devices, 275–278
  - GAP, 271–272
  - GATT, 272
  - general discussion, 270–272
  - hardware, 273
  - listing characteristics, 275–278
  - overview, 269–270
  - packet structure, 271
- BlueZ, *273*–274
- Bolshev, A., 367
- Bolt, Usain, 400
- Bonjour, 138–139
- boot environment, security testing of, 41
- boot modes, ST-Link programmer, 174–175
- boundary scan, *164*
- breadboard, *169*
- breakpoints in debugging, setting, 349
- brokers, in publish-subscribe architecture, 73
- brute-force attack, 213–214
  - cloning MIFARE Classic cards, 252–253
  - preshared key attacks, 301
  - on RFID reader authentication control, 262–263
  - Wi-Fi Direct, 296–297
- BSSID, 288
- `bufsiz` variable, 173
- built-in security for IoT devices, 5
- Bundle container, *347*
- Burp Proxy Suite, 356–357
- Burp Suite, 404
- Bus Blaster, 404
- Bus Pirate, *190*
  - attacking I^{2}C with, 202–206
  - communicating with SPI chip, 194–195
  - overview, 190, 404
  - reading memory chip contents, 196
- BusyBox, *67*
- busybox file, *217*
- BYPASS command, JTAG, 164

C

- CA (SSL certificate authority), 357
- cameras, IP. *See* IP cameras
- Capture the Flag (CTF). *See* BLE CTF Infinity
- CatWAN USB Stick, *309*, 404
  - turning into LoRa sniffer, 318–322
- `cbnz` command, 185–186
- C-ECHO messages, *96*–97
- C-ECHO requests dissector, building, 101–105
- central device, *270*
- Certificate Transparency, *37*
- CFAA (Computer Fraud and Abuse Act), 12–13
- characteristics, BLE, *272*
  - examining, 281–282
  - listing, 275–278
- `char-read-hnd <handle>` command, 282
- `charset` variable, 265
- `checkEmulatorStatus()` function, 368–369
- `check_fwmode` file, 71
- Chip Select (CS), 191
- ChipWhisperer, 404
- `chk` command, 252–253
- `chmod a+x <script_name>.js` command, 328
- `chmod` utility, 376
- Cipher-based Message Authentication Code (CMAC), *325*
- CIPO (Controller In, Peripheral Out), 191
- CIRCUITPY drive, 319, 320

- CircuitPython, 405
  - setting up, 318–319
  - writing LoRa sniffer, 320–322
- Cisco VoIP devices, imitating, 66–67
- classes, RFID tag, 243
- *classes.dex* file, 361
- Client, WS-Discovery, *145–146*
- client code, firmware update mechanisms for, 229–232
- client impersonation attacks, 94
- clients, enumerating and installing, 90
- client-side controls, *48*–49
- cloning RFID tags
  - high-frequency, 250–254
  - of keylock system, 372–375
  - low-frequency, 249
- cloud testing, 54
- Clutch, 344, 405
- CMAC (Cipher-based Message Authentication Code), *325*
- `cmd` struct, 85
- Code of Practice, UK, 14
- *code.py* file, 319, 320
- *com.android.insecureBankv2.PostLogin* file, 368
- combinator attack, *214*
- composition of IoT devices, 6
- Computer Fraud and Abuse Act (CFAA), 12–13
- `config_load "upnpd"` command, 123–124
- configuration files
  - finding credentials in firmware, 214–215
  - OpenOCD Server, 181–182
- `ConfigureConnection` command, 129
- *CONNACK* packet, MQTT, 75–76, 80, 82–84
- `connect <mac address>` command, 275
- *CONNECT* packet, MQTT, 74, 80–82
- `content update` command, 398
- contexts, DICOM, *103*–104
- contexts of IoT devices, 6
- `continue` command, GDB, 185
- continuity test, 161
- control data, in RFID tags, *243*
- control layer, UPnP, *120*
- Controller In, Peripheral Out (CIPO), 191
- Controller Out, Peripheral In (COPI), 191
- controller-peripheral I^2C bus architecture, setting up, 198–202
- control server, drug infusion pump, 19, 20
- control server service, *20*, 23–24
- cookies, reading, 350–351
- COPI (Controller Out, Peripheral In), 191
- CoreData framework, *348*
- cores, *171*
- crafting attacks, 152–153
- CRCs (cyclic redundancy checks), *243*, 313, 324
- credentials
  - finding in firmware configuration files, 214–215
  - firmware update services vulnerabilities, 233–234
  - WS-Discovery attacks, 153
- *Credentials.plist* file, 345
- crib dragging, *331*
- Cross-Site Request Forgery (CSRF) attacks, *49*
- cryptographic keys, 8
- CS (Chip Select), 191
- CSRF (Cross-Site Request Forgery) attacks, 49
- CTF. *See* BLE CTF Infinity
- CubicSDR, 376–378, 405
- cyclic redundancy checks (CRCs), *243*, 313, 324
- Cydia Impactor, 344

D

- Dalvik Executable (DEX) file formats, 361
- Damn Vulnerable ARM Router (DVAR), 235
- Damn Vulnerable IoT Device (DVID), 235
- Darkside attacks, *373*–374
- Dashboard APK, 397–398, 400
- databases of apps, inspecting, 366–367
- data bits, UART, 158
- Data container, *347*
- data encryption, testing, 53
- data link layer, *131*
- data protection, mobile app, 339–340
- datasheets, *37*
- Data Storage Format Identifier (DSFID), *244*
- DDoS (Distributed Denial of Service), 4–5
- Deauthenticate frames, *289*
- deauthentication attacks, 289–291
- debugging
  - assessment of interfaces, 42
  - Black Pill
    - with GDB, 183–188
    - setting up environment for, 170–172
  - on mobile apps, 348–349
- debug symbols, 183
- `#define` directives, 229
- `DeletePortMapping` command, 130
- denial of service attacks, *22*
  - ACK spoofing, 331
  - on control server service, 24
  - on drug library, 24
  - on firmware, 26
  - on hardware, 27
  - on IP cameras, 152–153
  - on operating system, 25
  - on pump service, 28

  - on restrictive user interface, 22–23
  - STRIDE threat classification model, 19
  - against wireless clients, 289–291
- dependent protocols, discovering, 90
- description layer, UPnP, *119*
- description XML file, *119*–120
- descriptors, BLE, *272*, 281–282
- DevAddr (end-device address), *324*, 326
- DevEUI (end-device identifier), 325
- device attestation, *18*
- device bootloaders, 211
- Device Discovery phase, Wi-Fi Direct, 296
- DevNonce, 325
- DEX (Dalvik Executable) file formats, 361
- Dex2jar, 361, 405
- dialout group, adding username to, 310
- `dicom_protocol .dissector()` function, 102
- `dicom.associate()` function, 114–115
- `dicom.pdu_header_encode()` function, 113
- DICOM pings, *96*–97
- DICOM protocol, *95*. *See also* DICOM service scanner
  - C-ECHO requests dissector, building, 101–105
  - general discussion, 95–97
  - Lua Wireshark dissector, developing, 99–101
  - traffic, generating, 97
- DICOM service scanner, *105*
  - A-ASSOCIATE request messages, 110–114
  - codes and constants, defining, 106–107
  - final script, writing, 113–114
  - functions for sending and receiving packets, 108–109
  - Nmap Scripting Engine library, creating, 106
  - overview, 105
  - packet headers, creating, 109–110
  - script arguments, reading in Nmap Scripting Engine, 112
  - socket creation and destruction functions, 107–108
- `dicom.start_connection()` function, 107–108
- dictionary attack, *49*
- differential power analysis, *42*
- Digital Millennium Copyright Act (DMCA), 12–13
- digital signing, 94
- directives, Nmap service probe, 72
- `disassemble` command, GDB, 184–185
- Disassociate frames, *289*
- discovering BLE devices, 275–278
- discovery layer, UPnP, *119*
- `dissector()` function, 99–100
- dissectors
  - C-ECHO requests, building, 101–105
  - Lua Wireshark, 99–101
  - testing Wireshark, 91
- Distributed Denial of Service (DDoS), 4–5
- DMCA (Digital Millennium Copyright Act), 12–13
- `dmesg` command, 246
- Domain Name System Service Discovery (DNS-SD), *131*
  - conducting reconnaissance with, 133–134
  - general discussion, 132–133
  - man-in-the-middle attacks
    - mDNS poisoner, creating, 141–144
    - mDNS poisoner, testing, 144–146
    - typical client and server interactions, 139–140
    - victim client, setting up, 138–139
    - victim server, setting up, 136–138
  - overview, 132
- `Dot1Q()` function, 64
- double tagging attacks, 63–65
- downgrade attacks, 94
- downtime, *52*
- DREAD Classification Scheme, 29–30
- Drozer, 363–365, 405
- drug infusion pump
  - architecture of, 19–21
  - identifying threats
    - attack trees, using, 28–29
    - control server service, 23–24
    - drug library, 24
    - firmware, 25–26
    - operating system, 25
    - overview, 21–22
    - physical equipment, 26–27
    - pump service, 27–28
    - RUI, 22–23
- DSFID (Data Storage Format Identifier), *244*
- DTP (Dynamic Trunking Protocol), *61*
- *dumpedkeys.bin* file, 253–254
- `dump` parameter, 253–254
- `dumptoemul` script, 263
- DVAR (Damn Vulnerable ARM Router), 235
- DVID (Damn Vulnerable IoT Device), 235
- dynamic analysis
  - firmware, 221–223
  - InsecureBankV2 app, 363–367
  - OWASP iGoat app, 347–353
- dynamic patching
  - jailbreak detection, avoiding with, 357–358
  - root detection, avoiding with, 369–370
- Dynamic Trunking Protocol (DTP), *61*

E

- EAP over LAN (EAPOL) handshake, 299–*300*
- EAP-TLS, 304–305
- EAP-Tunneled-TLS (EAPTTLS), 304–305
- eavesdropping, *331*
  - LoRaWAN, 331
  - on tag-to-reader communication, 260–261
- Eclipse Mosquitto software, 75
- economics of IoT manufacturing, 6
- EEPROM flash memory chips, dumping with SPI, 192–196
- `eget` command, 256
- electronic health record (EHR), *19*
- elevation of privilege, *23*
  - on control server service, 24
  - on drug library, 24
  - on firmware, 26
  - on hardware, 27
  - on operating system, 25
  - on pump service, 28
  - on restrictive user interface, 23
  - smart treadmills, attacking, 394
  - STRIDE threat classification model, 19
- ELF (Executable and Linkable Format) file, *183*
- `eload` parameter, 255, 265
- emergency stop key, *398*
- *.eml* file, loading in Proxmark3 memory, 265
- emulation, firmware, 216–221
- Enabled Protocols window, Wireshark, 91–92
- encryption
  - checking for, 94
  - mobile app filesystem, 339–340
  - testing, 53
- end-device address (DevAddr), *324*, 326
- end-device identifier (DevEUI), 325
- endianness of protocol, 93
- EPSON's iPrint application, 363
- escalating privileges. *See* elevation of privilege
- `eset` parameter, 255
- ESP32 development board, 273, 309–314
- ESSID, 288
- *etc/passwd* file, 213, 221
- `Ether()` function, 64
- eventing layer, UPnP, *120*
- EvilDirect attack, *297*–299
- Evil Twin attack, *291*–292
- exacqVision, 147–152
- `Exclude` Nmap service probe directive, 72
- Executable and Linkable Format (ELF) file, *183*
- executable binary, inspecting for memory protections, 345–346
- exploitation, protocol or service, 47
- Exported Activities, in Android apps, 361
- Extensible Authentication Protocol (EAP) over LAN (EAPOL) handshake, 299–*300*
- External Entity (XXE) attacks, *121*
- EXTEST command, JTAG, 164

F

- FACT (Firmware Analysis and Comparison Tool), 406
- fail-open conditions, *49*
- faking cameras on network
  - analyzing requests and replies in Wireshark, 147–149
  - emulating cameras, 149–152
  - setting up, 147
- `fallback` Nmap service probe directive, 72
- FBE (file-based encryption), *339*–340
- FCC ID online database, 37–38
- `fchk` command, 253
- FCntDown frame counter, 330
- FCntUp frame counter, 330
- FDE (full disk encryption), *339*–340
- federal laws affecting IoT research, 12–13
- Fernandes, E., 368
- `fetchButtonTapped` function, 358–359
- FFmpeg, 384
- FHDR (frame header), 324
- file-based encryption (FBE), *339*–340
- File Manager application, treadmill browser, 392–393
- file structure, iOS, 347
- filesystem
  - access controls, testing, 53
  - firmware, 212–216
  - mobile app, 339–340
- `find` command, 347
- fingerprinting, *44*, 67–71
- firewalls
  - disabling in firmware, 222
  - punching holes through, 121–126
- FIRMADYNE, *216*, 218–221, 227, 405
- Firmwalker, 215–216, 405
- firmware, *25*. *See also* firmware update mechanisms; Wi-Fi modem router hacking
  - backdooring, 223–228
  - general discussion, 208
  - identifying threats to, 25–26
  - obtaining, 209–211
  - security testing, 42
- Firmware Analysis and Comparison Tool (FACT), 406
- firmware-mod-kit, 226
- firmware update mechanisms, *228*
  - client code, 229–232

  - compilation and setup, 229
  - general discussion, 228
  - running update service, 232–233
  - vulnerabilities of, 233–235
- fixed header, MQTT *CONNECT* packet, *80*–82
- flags, 355
- flash memory chips, dumping with SPI, 192–196
- `flashrom` Linux utility, 195–196
- flooding attacks, 94
- flow diagrams, 38–39
- forced browsing, *50*
- `ForceTermination` command, 129
- `fork()` command, 224
- Forshaw, James, 92, 116
- Fourier transforms, 47
- four-way handshake, WPA/WPA2, 299–300
- FPort, 324
- frame header (FHDR), 324
- frameworks, *8*–10
- Frida instrumentation framework, 406
  - jailbreak detection, avoiding, 357–358
  - root detection, avoiding, 369–370
  - treadmill software and physical buttons, disabling, 398–399
- FRMPayload, 324, 327
- `fs` command, 355
- `fswatch` application, 347–348
- FTDI FT232RL, 406
- full disk encryption (FDE), *339*–340
- fuse, *32*
- `fuzz()`function, 266
- fuzzing
  - overview, 94
  - RFID, using custom scripting, 264–268

G

- GAP (Generic Access Profile), *271*–272
- Garcia, Daniel, 118, 128
- Garg, Praerit, 18
- gateways, LoRaWAN, *309*
- GATT (Generic Attribute Profile), *272*
- GATTTool, *275*, 406
  - discovering devices and listing characteristics, 275–276
  - hacking BLE, 279–285
  - reading and writing characteristics, 278
- GDB, *172*, 406
  - debugging with, 183–188
  - installing, 172
- `gdb-multiarch` command, 183
- Geiger, Harley, 12–13
- Generic Access Profile (GAP), *271*–272
- Generic Attribute Profile (GATT), *272*
- Generic Attribute Profile Tool (GATTTool). *See* GATTTool
- `GetAutoDisconnectTime` command, 129
- `GetConnectionTypeInfo` command, 128
- `GetExternalIPAddress` command, 130
- `GetGenericPortMappingEntry` command, 129
- `GetIdleDisconnectTime` command, 129
- `GetLinkLayerMaxBitRates` command, 129
- `GetNATRSIPStatus` command, 129
- `GetPassword` command, 129
- `GetPPPAuthenticationProtocol` command, 129
- `GetPPPCompressionProtocol` command, 129
- `GetPPPEncryptionProtocol` command, 129
- `GetSpecificPortMappingEntry` command, 129
- `GetStatusInfo` command, 129
- `GetUserName` command, 129
- `GetWarnDisconnectDelay` command, 129
- Ghidra, 185, 406
- `git` command, 226–227
- glitching attacks, *42*
- GND (ground line), 197, 199
- GND (Ground) port, UART, 159, 161–162, 178
- GNUcitizen, 118
- GNU Debugger (GDB), *172*
  - debugging with, 183–188
  - installing, 172
- Goldberg, Dave, 400
- Goode, Lauren, 4
- Google Dorks, 209
- Ground (GND) port, UART, 159, 161–162, 178
- ground line (GND), 197, 199
- group owner, *295*
- Group Temporal Key (GTK), *300*
- guidance documents, *8*–10

H

- HackRF One, 407
- HAL (Hardware Abstraction Layer), *396*
- `halt` command, 182
- hardcoded credentials, 233–234
- hardware
  - BLE, 273
  - identifying threats, 26–27
  - security testing, 40–43
  - smart treadmill design, 394–396
  - for Wi-Fi security assessments, 288
- Hardware Abstraction Layer (HAL), *396*
- Hardware Abstraction Layer APK, 396
- hardware folder, Arduino IDE, 170–171
- hardware integrity attacks, *32*
- Hashcat, 213–214, 302, 304, 407
- hashid, 213–214
- hashing algorithms, insecure, 234
- Hciconfig, *274*
- Hcxdumptool, 302–303, 407
- `hcxpcaptool` command, 303
- Hcxtools, 302, 407

Heffner, Craig, 163
Heltec LoRa 32 development board, *309*
  overview, 309, 407
  programming as LoRa sender, 310–313
  setting up, 309–310
  testing LoRa sender, 310–314
hf 14a raw command, 258–259
hf-mf-B46F6F79-data.bin file, 254
hf mf command, 251
hf mf ecfill command, 374
hf mf mifare command, 373–374
hf mf nested command, 374
hf mf rdsc command, 253
hf mf sim command, 262–263, 375
hf parameter, 248
hf search command, 258, 372–373
Hickory Smart app, 351
hidden content, *48*
hidden Wi-Fi networks, *288*
HID Global ProxCard, 244, 246
hid parameter, 249
high-frequency RFID
  antennas for, 243
  cloning tags, 250–254
  general discussion, 245
  identifying with Proxmark3, 248–249
HiLetgo USB logic analyzer, 176–177
Hippocratic Oath for Connected Medical Devices, 9
HMAC-MD5, 234
Homebrew package, 347–348
host configuration review, *50*–54
host discovery, *43*
HTTP caches, 350
Huawei HiLink app, 353
Hydrabus, 407

I

I^{2}C. *See* Inter-Integrated Circuit (I^{2}C)
I Am The Cavalry framework, 5–6, 9
IDA Pro, 408
IDE (Integrated Development Environment), Arduino. *See* Arduino Integrated Development Environment (IDE)
identification data, in RFID tags, *243*
idle state, UART, 158
IDOR (Insecure Direct Object References), *54*
IGD (Internet Gateway Device) protocol. *See* Internet Gateway Device (IGD) protocol
iGoat mobile app
  binary reversing, 355–356
  dynamic analysis, 347–353
  injection attacks, 353–354
  IPAs, extracting and re-signing, 343–344
  jailbreak detection, avoiding, 357–360
  keychain storage, 354
  network traffic, intercepting and examining, 356–357
  overview, 341–342
  static analysis, 344–346
  test environment, preparing, 342–343
iI command, 355
implicit header mode, 322
incline of treadmills, remotely controlling, 394–398
info functions command, GDB, 183–184
info registers command, GDB, 185
information disclosure, *22*
  on control server service, 24
  on drug library, 24
  on firmware, 26
  on hardware, 27
  on operating system, 25
  on pump service, 28
  on restrictive user interface, 22
  STRIDE threat classification model, 19
information-gathering phase, network protocol inspections, 90–91
Information Object Definitions (IODs), *110*–111
information property list file, *344*–345
init command, 182
injection attacks
  OWASP iGoat app, 353–354
  SQL, 24, 120, 354
  XSS, 353–354
injuries due to treadmill attacks, 400
input validation, 50
InsecureBankV2 app
  binary reversing, 362–363
  dynamic analysis, 363–367
  extracting APK, 361
  intercepting and examining network traffic, 367
  overview, 360
  preparing test environment, 360–361
  side-channel leaks, 367–368
  static analysis, 361–362
Insecure Direct Object References (IDOR), *54*
insecure hashing algorithms, 234
insecurity canaries, 14
insulin pumps, 11–12, 16
Inter-Integrated Circuit (I^{2}C), *189*
  Bus Pirate, attacking with, 202–206
  controller-peripheral bus architecture, setting up, 198–202
  general discussion, 197–198
  hardware for communicating with, 190–191
  overview, 189

Internet Gateway Device (IGD) protocol, *121*
  abusing UPnP through WAN interfaces, 126–131
  punching holes through firewalls, 121–122, 124–125
Internet of Things (IoT), *3–4*
IODs (Information Object Definitions), *110*–111
iOS apps
  binary reversing, 355–356
  dynamic analysis, 347–353
  injection attacks, 353–354
  IPAs, extracting and re-signing, 343–344
  jailbreak detection, avoiding, 357–360
  keychain storage, 354
  network traffic, intercepting and examining, 356–357
  overview, 341–342
  security controls, 339–341
  static analysis, 344–346
  test environment, preparing, 342–343
  threats to, 337–338
iOS App Store Package (IPA), *343*–344
IoT (Internet of Things), *3–4*
IoT devices, identifying on networks
  fingerprinting services, 67–71
  Nmap service probes, writing new, 71–73
IoT security
  expert perspectives, 12–16
  frameworks, standards, and guides, 8–10
  hacking techniques, 6–8
  importance of, 4–5
  insulin pump security issue, 11–12
  traditional security versus, 5–6
IPA (iOS App Store Package), *343*–344
IP cameras
  faking on network
    analyzing requests and replies in Wireshark, 147–149
    emulating cameras, 149–152
    setting up, 147
  Nmap service probes, writing, 71–73
  playing back stream from
    analyzing network traffic, 380–382
    extracting video stream, 382–385
    overview, 379–380
    streaming protocols, 380
  service fingerprinting, 67–71
  WS-Discovery attacks, 152–153
`IP()` function, 64
ippserver, *136*–137, 140, 145
`iptables` utility, 222
iRemocon- WiFi app, 367
`iwconfig` command, 299

J

JADX, 361, 408
jailbreak detection, avoiding, 357–358
jamming wireless alarms, 375–379
Jarsigner, 369
John the Ripper, 213–214, 408
Join-Accept, 326
joining LoRaWAN networks, 324–327
Join-Request, 325–326
Joint Test Action Group (JTAG), *157*
  boundary scan commands, 164
  hardware tools for communicating with, 165
  identifying pins, 166–167
  overview, 157–158, 164
  Test Access Port (TAP), 164–165
JTAGenum utility, 167
JTAGulator, *166*, 408
jumper pins, *175*
jumper wires, 169
Junior, M., 366

K

KARMA attack, *292*
keychain service API, *354*
keychain storage, iGoat app, 354
key generation and management, LoRaWAN, 330
keylock systems, cloning RFID tag of, 372–375
Keytool, 369
kiosk mode, *40*–41
Known Beacons attack, *292*–295
Kohnfelder, Loren, 18
Kr00k, 291

L

laws affecting IoT research, 12–13
legitimate RFID reader attack, 262–263
`lf` parameter, 248
LimeSDR, 408
`list` command, GDB, 184–185
LLDB, 348–350, 408
`llvm clang` static analyzer, 346
`load_seed_tag()` function, 265, 266
local-link protocols, 131–132
local-scope helper function, *79*
lockout mechanisms, *51*
locks, assessing, 41
log files, sensitive, 234
logging, UPnP, 121
logic analyzer, *162*, 163, 176–177
logic flaws, 50

- Long Range (LoRa), *307–308*. *See also* LoRaWAN protocol
  - CatWAN USB stick, turning into LoRa sniffer, 318–322
  - Heltec LoRa 32, setting up, 309–314
  - LoStik, setting up, 314–318
  - overview, 308–309
  - physical layer, 323–324
  - sending packets, 311–313
- loop()function, 173–174, 313
- LoRa.endPacket() function, 313
- *lora-packet* library, 328–329
- LoRa.print() function, 313
- LoRaWAN protocol
  - ACK spoofing, 331
  - application-specific attacks, 331
  - bit-flipping attacks, 327–330
  - eavesdropping, 331
  - general discussion, 308–309
  - joining networks, 324–327
  - key generation and management, 330
  - overview, 318–322
  - packet format, 323–324
  - replay attacks, 330–331
- LoStik, *309*, 314–318, 409
- low-frequency RFID
  - antennas for, 243
  - cloning tags, 249
  - general discussion, 244–245
  - identifying with Proxmark3, 248–249
- Low-Power Wide Area Network (LPWAN), *307*–309. *See also* Long Range (LoRa); LoRaWAN protocol
- Lua, *95*. *See also* DICOM service scanner
  - enabling in Wireshark, 97–99
  - general discussion, 95
  - generating DICOM traffic, 97
  - prototyping with, 93
  - Wireshark dissector, developing for DICOM protocol, 99–101

M

- MAC address, spoofing, 283–285
- MAC header (MHDR), *324*
- MAC layer, LoRaWAN, 324
- mac pause command, 316
- MACPayload, 324
- macros, I^2C library, 204–205
- main() function, 142, 229–232
- make command, 226
- Malith, Osanda, 224
- managed service accounts, *52*
- management frame, *288*
- man-in-the-middle attacks, *23*. *See also* playing back IP camera stream
  - on control server service, 23
  - iOS apps, 356–357
  - mDNS or DNS-SD
    - mDNS poisoner, creating, 141–144
    - mDNS poisoner, testing, 144–146
    - typical client and server interactions, 139–140
    - victim client, setting up, 138–139
    - victim server, setting up, 136–138
  - obtaining firmware through, 211
- manuals, system, 37
- manufacturer data, *244*
- match Nmap service probe directive, 72
- MCU (microcontroller unit), *211*
- MD5 hash, 232
- MDM (Mobile Device Management), *386*
- mDNS. *See* multicast Domain Name System (mDNS)
- MDNS class, creating, 143–144
- mDNS poisoner
  - creating, 141–144
  - testing, 144–146
  - typical client and server interactions, 139–140
  - victim client, setting up, 138–139
  - victim server, setting up, 136–138
- MDNS_poisoner function, 142
- mDNS reflection DDoS attack, 94
- mdw command, 182
- medical device security
  - insulin pump security issue, 11–12
  - patient perspectives on, 14–16
- memory corruption bugs, 120
- Message Integrity Code (MIC), *300*, 323–326
- Message Queuing Telemetry Transport (MQTT)
  - publish-subscribe architecture, 73–74
  - test environment, setting up, 75–76
  - testing Ncrack module against, 86–87
  - writing authentication-cracking module in Ncrack, 77–86
- mfkey64 tool, 262
- mfkeys script, 263
- MHDR (MAC header), *324*
- MIC (Message Integrity Code), *300*, 323–326
- microcontroller hacking
  - boot mode, selecting, 174–175
  - coding target program in Arduino, 172–174
  - connecting to computer, 179–180
  - connecting USB to serial adapter, 178
  - debugging environment, setting up, 170–172
  - debugging target, 181–188
  - flashing and running Arduino program, 174–180
  - STM32F103C8T6 target device, 169–170
  - tools for, 168–169

  - UART pins, identifying with logic analyzer, 176–177
  - uploading Arduino program, 175–176
- microcontroller unit (MCU), *211*
- MIFARE cards
  - access bits, 251
  - altering RFID tags, 255–256
  - attacking with Android app, 256–257
  - authentication protocol, 258, 259
  - cloning Classic cards, 250–254
  - cloning RFID tag of keylock system, 372–375
  - extracting private key from captured traffic, 261–262
  - MIFARE Classic memory map, 250
  - overview, 245
  - raw commands, reading with, 258
  - simulating RFID tags, 254–255
- MIFARE Classic Tool, 256–257
- mini ST-Link programmer, *168*
- MiniUPnP, setting up, 122–124
- Mirai botnet, 4–5, 6
- Miranda, 125, 130, 409
- mobile apps. *See also* iGoat mobile app; InsecureBankV2 app
  - architecture of, 336
  - general mobile device threats, 337
  - overview, 335–336
  - root detection, avoiding, 368–370
  - security controls, 339–341
  - security testing, 54
  - threats to, 337–338
- Mobile Device Management (MDM), *386*
- Mobile Security Framework (MobSF), 346, 409
- ModemManager, 247
- `modprobe` command, 63
- Moe, Marie, 15
- monitor mode, AP, *288*
- Moore, H.D., 118
- *.mpy* files, 319
- MQTT. *See* Message Queuing Telemetry Transport (MQTT)
- `MQTT_FINI` state, Ncrack, 79–80, 85–86
- `MQTT_INIT` state, Ncrack, 79–80, 84–86
- `mqtt_loop_read` function, 79, 83, 86
- `msearch` command, 125
- M-SEARCH request, *119*
- MU editor, 320–322
- multicast Domain Name System (mDNS), *131*
  - abusing Probing phase, 134–136
  - general discussion, 132
  - man-in-the-middle attacks
    - mDNS poisoner, creating, 141–144
    - mDNS poisoner, testing, 144–146
    - typical client and server interactions, 139–140
    - victim client, setting up, 138–139
    - victim server, setting up, 136–138
  - overview, 131–132
  - reconnaissance with, 133–134
- multimeters, *160*–162
- mutation-based fuzzing, 264
- mutual authentication, 94
- MyCar Controls mobile app, 356

N

- NAC (network access control), 18
- NarrowBand (NB-IoT), *308*
- NAT (network address translation), *121*
- native VLAN, 63, *63*
- NB-IoT (NarrowBand), *308*
- `ncat` Nmap command, 69
- Ncrack, *74*, 409
  - architecture of, 77
  - compiling, 77–78
  - initializing modules, 78–79
  - overview, 77
  - testing module against MQTT, 86–87
  - writing authentication-cracking module in, 77–86
- `ncrack_mqtt` function, 84–86
- *ncrack-services* file, 78
- Near-Field Communication (NFC), *245*, 296
- nested authentication attack, *374*
- Netgear D6000
  - dynamic analysis, 221–223
  - extracting filesystem, 212
  - firmware emulation, 216–221
  - overview, 211–212
  - statically analyzing filesystem contents, 213–216
  - support page, 211
  - web app, 223
- NetID (network identifier), 326
- `netstat` command, 222
- network access control (NAC), *18*
- network address translation (NAT), *121*
- network assessments
  - identifying IoT devices on networks, 67
    - with fingerprinting services, 67–71
    - Nmap service probes, writing new, 71–73
  - MQTT, attacking
    - overview, 73–74
    - test environment, setting up, 75–76
    - testing Ncrack module against MQTT, 86–87
    - writing authentication-cracking module in Ncrack, 77–86

- network assessments, *continued*
  - overview, 59
  - VLAN-hopping attacks
    - double tagging attacks, 63–65
    - imitating VoIP devices, 65–67
    - overview, 60
    - switch spoofing attacks, 61–63
    - VLANs and network switches, understanding, 60–61
- network identifier (NetID), 326
- network layer, *43*–47
- network printers, man-in-the-middle attacks on. *See* mDNS poisoner
- network protocols. *See also* DICOM protocol
  - analysis phase, 92–93
  - C-ECHO requests dissector, building, 101–105
  - information-gathering phase, 90–91
  - overview, 89–90
  - prototyping and tool development, 93
  - security assessments, 93–94
  - stages of attacks on, 45–47
- network server, *309*
- network sockets, setting up, 150
- network switches, 60–61
- network traffic
  - analysis of, 46
  - analyzing with Wireshark, 92
  - InsecureBankV2 app, 367
  - of IP camera stream, analyzing, 380–382
  - obtaining copies of, 92
  - OWASP iGoat app, 356–357
- `NewInternalClient` bug, 120
- NFC (Near-Field Communication), *245*, 296
- Nmap, 409
  - fingerprinting services, 67–71
  - writing new service probes, 71–73
- *nmap* library, 106–107
- Nmap Scripting Engine (NSE), 112. *See also* DICOM service scanner
- Nmap Scripting Engine (NSE) Library, 106
- *nmap-service-probes* file, 68, 70, 72
- `nmcli` command, 63
- node-applesign, 344
- node cloning, *32*
- Node.js, 327–328
- nodes, LoRaWAN, *309*
- NOTIFY message, 119
- NotPetya attack, 5
- `npm` package manager, 328
- `nr` parameter, 261–262
- NSE (Nmap Scripting Engine), 112. *See also* DICOM service scanner
- NSE (Nmap Scripting Engine) Library, 106
- *nsedebug* library, 106
- NSEdoc block format, 108
- `nsock_iod` variable, 85
- `nsock_read` variable, 85
- `nsock_write` variable, 85
- *Nsock* library, 77
- `nt` parameter, 261–262
- `NULL` probe, Nmap, 72
- NwkSKey, *323*, 326
- NXP cards, 245

O

- OhMiBod Remote app for Android, 367
- *ojbc.pl* script, 355
- ONVIF, 145
- open drain drivers, *197*
- open mode, Wi-Fi networks, *288*
- OpenOCD (Open On-Chip Debugger), *165*
  - installing, 171–172
  - overview, 165, 409
- open-source intelligence (OSINT), *37*–40
- Open Web Application Security Project (OWASP), 9–10. *See also* iGoat mobile app
- OpenWrt
  - compiling backdoor agent, *225*–226
  - test UPnP server, setting up, 122–125
- operating system, 25, 44
- Orthanc, *97*
- OSINT (open-source intelligence), *37*–40
- Otool, 345–346, 410
- over-the-air (OTA) updates, 26
- Over-the-Air Activation (OTAA) method, *323*–326, 330–331
- OWASP (Open Web Application Security Project), 9–10
- OWASP iGoat project. *See* iGoat mobile app
- OWASP IoTGoat, 235
- OWASP Zed Attack Proxy (ZAP), 410

P

- pacemakers, 15
- packet format
  - LoRaWAN, 323–324
  - UART, 158
- packet injection capabilities, *288*
- packet structure, BLE, 271
- Padding Oracle on Downgraded Legacy Encryption (POODLE) attack, 94
- pads, UART, *159*
- Pairwise-Master Key (PMK), 300
- Pairwise Master Key Identifier (PMKID) field, *299*, 301–304
- Pairwise Transient Key (PTK), 300
- parallel communication protocols, *158*
- parity bit, UART, 158
- parsing variable-length fields, 103–104

- passive reconnaissance, *37*–40, 292
- passive RFID technologies, *241*–242
- passive spidering, *48*
- password expiration, *51*
- password history, *51*
- passwords
  - cracking Netgear D6000 admin credentials, 213–214
  - expiration, 51
  - firmware update services vulnerabilities, 233–234
  - history, 51
  - resetting in Android apps, 363–365
  - strength, testing, 51
  - uncovering by fingerprinting services, 67–71
- pasteboard, *353*
- patch levels, testing, 52
- patents, 38
- payload, MQTT *CONNECT* packet, *81*–82
- PBC (Push-Button Configuration), *296*, 297–299
- PCB. *See* printed circuit board (PCB)
- (P-DATA-TF) message, 96
- `pdc` command, 356
- `pdf` command, 356
- `pdu_header_encode()` function, 110
- PDUs (Protocol Data Units), 96–97, 271
- PEAP (Protected-EAP), 304–305
- penetration testing. *See* security testing methodology
- perception layer, testing, 47
- peripheral interfaces, *40*–41
- Personally Identifiable Information (PII), *53*
- PHDR (physical header), 324
- PHDR_CRC, 324
- PHI (Protected Health Information), *53*
- Philips HealthSuite Health Android app, 366
- Pholus, 410
  - abusing mDNS Probing phase, 134–136
  - reconnaissance with, 133–134
- PHYPayload, 324
- physical buttons of treadmills, disabling, 398–400
- physical entry into smart homes
  - cloning RFID tag of keylock system, 372–375
  - jamming wireless alarms, 375–379
- physical equipment, identifying threats to, 26–27
- physical header (PHDR), 324
- physical robustness, testing, 42–43
- PII (Personally Identifiable Information), *53*
- PIN entry, WPS, 296–297
- pinout, UART, *159*–162
- pins
  - Arduino Uno, 198
  - flash memory chips, 192–193, 194
  - JTAG, 166–167
  - UART, 159–162, 176–177
- playing back IP camera stream
  - analyzing network traffic, 380–382
  - extracting video stream, 382–385
  - overview, 379–380
  - streaming protocols, 380
- `PLAY` request, 382
- Plutil tool, 345
- PMK (Pairwise-Master Key), 300
- PMKID (Pairwise Master Key Identifier) field, *299*, 301–304
- pointer records (PTR), 132–133, 138, 139–140
- POODLE (Padding Oracle on Downgraded Legacy Encryption) attack, 94
- `portfwd` command, 399
- ports
  - for network protocols, 90
  - UART, 159–162
- `ports` Nmap service probe directive, 72
- preamble, radio, 323
- Preboot Execution Environment (PXE), *41*
- predictive text engine, *353*
- preferred network list, *292*
- Presentation Context, A-ASSOCIATE request message, 111
- presentation layer, UPnP, *120*
- preshared key attacks, 299
- principle of least privilege, *51*–52
- printed circuit board (PCB)
  - JTAG pins on, 166, 167
  - UART pins on, 159–160
- printers, man-in-the-middle attacks on. *See* mDNS poisoner
- privacy breaches, 32
- privilege, elevation of. *See* elevation of privilege
- PRNG (pseudorandom number generator), *373*
- `Probe` Nmap service probe directive, 72
- probe request, *292*
- Probing phase, mDNS, 132, 134–136
- Protected-EAP (PEAP), 304–305
- Protected Health Information (PHI), *53*
- Protocol Data Units (PDUs), 96–97, 271
- `ProtoField` class, 99
- `ProtoField.string` function, 102
- `Proto(name, description)` function, 99
- prototyping, *93*
- ProxCard, HID Global, 244, 246
- Proxmark3, 410
  - altering RFID tags, 255–256
  - attacking MIFARE with Android app, 256–257
  - automating RFID attacks using Scripting Engine, 263–264

Proxmark3, *continued*
- eavesdropping on tag-to-reader communication, 260–261
- extracting sector's key from captured traffic, 261–262
- high-frequency tag cloning, 250–254
- identifying low- and high-frequency cards, 248–249
- keylock system tags, cloning, 372–375
- legitimate RFID reader attack, 262–263
- low-frequency tag cloning, 249
- overview, 245–246
- RAW commands, 258–260
- RFID fuzzing using custom scripting, 264–268
- setting up, 246
- simulating RFID tags, 254–255
- updating, 246–248

pseudorandom number generator (PRNG), *373*,
PTK (Pairwise Transient Key), 300
PTR (pointer records), 132–133, 138–140
publish-subscribe architecture, *73*
pump service, identifying threats to, 27–28
punching holes through firewalls, 121–126
Pupy, 391, 393, 399, 410
`pupygen` command, 391
Push-Button Configuration (PBC), *296*–299
PuTTY, 319–320
PXE (Preboot Execution Environment), *41*
`pyserial` package, 315–316
Python 2, 128, 134

Q

Qark, 361, 410
QEMU (Quick Emulator), *216*–217, 411
QU bit, 132

R

`r2` command, 355
`rabin2` command, 355
Radare2, 355–356, 358, 411
Radcliffe, Jay, 11–12, 16
Radio Frequency Identification (RFID), *239*. *See also* Proxmark3
- altering tags, 255–256
- automating attacks using Scripting Engine, 263–264
- eavesdropping on tag-to-reader communication, 260–261
- extracting sector's key from captured traffic, 261–262
- fuzzing using custom scripting, 264–268
- general discussion, 240
- high-frequency tag cloning, 250–254
- high-frequency tags, 245
- identifying low- and high-frequency cards, 248–249
- keylock system tags, cloning, 372–375
- legitimate reader attack, 262–263
- low-frequency tag cloning, 249
- low-frequency tags, 244–245
- overview, 239
- passive and active technologies, 241–242
- radio frequency bands, 240–241
- RAW commands for tags, 258–260
- simulating tags, 254–255
- structure of tags, 242–244

radio jamming, *291*
`radio rx 0` command, 316–317
`radio set crc off` command, 316–317
`radio set sf sf7` command, 316–317
`radio set wdt 0` command, 316
RADIUS (Remote Authentication Dial-In User Service) server, *304*
`randomize()` function, 265–266
ransomware attacks, 5
rarity level, *68*
`rarity` Nmap service probe directive, 72
Raspberry Pi, transforming into radio transmitter, 378–379
rate limiting, *297*
RAW commands for RFID tags, 258–260
RBAC (role-based access control), 49–50
`rdbl` parameter, 253
Read-Only Memory (ROM), *338*
Realtek RTL2832U chipset, 376
Reaver, 297, 411
Receive (RX) port, UART, 159, 162, 178
`receive(dcm)` function, 109
reconnaissance
- active, *43*–45
- with DNS-SD, 133–134
- with mDNS, 133–134
- passive, *37*–40, 292

`recv_data()` function, 174
Remote Authentication Dial-In User Service (RADIUS) server, *304*
remotely controlling speed and incline of treadmills, 394–398
remote maintenance, security testing for, 53
remote shell access, getting, 391
replay attacks, *23*, *31*
- on control server service, 23
- LoRaWAN, 330–331
- overview, 31

Repository APK, 397–399
repudiation, *22*
- on control server service, 23
- on drug library, 23
- on firmware, 26

  on hardware, 27
  on operating system, 25
  on pump service, 28
  on restrictive user interface, 22
  STRIDE threat classification model, 19
Request all (REQA) command, 258–259
`RequestConnection` command, 129
`RequestTermination` command, 129
`restore` parameter, 254
restrictive user interface (RUI), *22*–23
`ret_code` field, 83
reverse engineering protocols, 46–47
revocation evasion attack, 7–8
RfCat, 411
RFID. *See* Radio Frequency Identification (RFID)
`rfm9x` class, 322
`rfm9x.receive()` function, 322
`rfm9x.rssi()` function, 322
RFQuack, 411
Robust Security Network (RSN), *301*
Rogers, David, 14
role-based access control (RBAC), *49*–50
ROM (Read-Only Memory), *338*
root detection, avoiding, 368–370
root user, 394
Rpitx, 378–379, 411
`rpitx` command, 378–379
RSN (Robust Security Network), *301*
RTCP protocol, 380
RTL-SDR DVB-T dongle, 375–376, 411
rtpdump file, 383–384
`rtpplay` command, 384
RTP protocol, 380
RTP stream, extracting, 383
RTP Tools, 384, 412
`RTSP DESCRIBE` request, 381
`RTSP OPTIONS` request, 381
RTSP protocol, 380
RUI (restrictive user interface), 22–23
RX (Receive) port, UART, 159, 162, 178

S

S3Scanner, 209–210, 412
Saleae logic analyzer, 163, 176–177
SAMD21 microcontroller, 318
SAMPLE/PRELOAD command, JTAG, 164
sandbox, 340, 347
SCAN request, *270*
Scapy, 64–65, 412
Schlage door lock companion app, 368
SCK (Serial Clock), 191
SCL (serial clock line), 197, 199–200
screenshots, application, 352
Scripting Engine, Proxmark3, 263–264
`script list` command, 263
`script run` command, 263
`script run fuzzer` command, 267
SDA (serial data line), 197, 199–200
SD card directory, inspecting, 367
SDP (Session Description Protocol) file, 381–383
SDR (software defined radio), *375*–376
`search` parameter, 248
sector trailer, 250–*251*
secure boot, *341*
Secure Enclave, *341*
secure IPC, 340
secure mode, Wi-Fi networks, *288*
SecureRom, *341*
Secure RTP (SRTP) protocol, *385*
security breaches, 32
security testing methodology
  cloud testing, 54
  conceptual layers, 36
  hardware layer, 40–43
  host configuration review, 50–54
  mobile applications, 54
  network layer, 43–47
  overview, 35–37
  passive reconnaissance, 37–40
  web applications, 48–50
seed, *264*
Segger J-Link Debug Probe, *165*
Select card command, 258–260
selective jamming, *291*
semi-passive RFID technologies, *242*
`send_cmd` function, 317
`send(dcm, data)` function, 109
sensitive log files, 234
Serial Clock (SCK), 191
serial clock line (SCL), 197, 199–200
serial data line (SDA), 197, 199–200
Serial Monitor, *180*, 310, 313, 319–320
Serial Peripheral Interface (SPI), *189*
  dumping EEPROM flash memory chips with, 192–196
  general discussion, 191
  hardware for communicating with, 190–191
  overview, 189
serial protocols, *158*
Serial Wire Debug (SWD), *158*
  hacking devices through
    coding target program in Arduino, 172–174
    debugging environment, setting up, 170–172
    debugging target, 181–188
    flashing and running Arduino program, 174–180
    STM32F103C8T6 target device, 169–170
    tools for, 168–169

Serial Wire Debug (SWD), *continued*
hardware tools for communicating with, 165
overview, 158, 165
Serial Wire or JTAG Debug Port (SWJ-DP), *165*
ser Serial object, 317
server impersonation attacks, 94
server_ip variable, 230
server misconfiguration, testing for, 54
<service> tag, 120
Service Discovery phase, Wi-Fi Direct, 296
services, BLE, *272*
service scanner, DICOM. *See* DICOM service scanner
service scanning, 44
service version detection, 44
Session Description Protocol (SDP) file, 381–383
session management, 49
SetAutoDisconnectTime command, 129
SetConnectionType command, 128
SetIdleDisconnectTime command, 129
settings tampering attacks, *32*
setup() function, 173, 312–313
SETUP request, 381–382
SetWarnDisconnectDelay command, 129
SGX, *341*
shared memory segment, *397*
Shikra, 412
short-range radio, 239. See also specific technologies
showRootStatus() function, 368–369
side-channel analysis, *330*
side-channel leaks, 367–368
signal jamming attacks, *31*
signal-to-noise ratio (SNR), *375*
signatures, application, 340
signing, 94
SIMATIC WinCC OA Operator application, 348
sim parameter, 255
Simple Object Access Protocol (SOAP), *120*
simulating RFID tags, 254–255
skeleton file for mDNS poisoner, 141–143
skimming attacks, *23*
small outline integrated (SOIC) clips, 190, 193
smart door locks, circumventing, 372–375
smart homes. *See also* smart treadmills, attacking
cloning RFID tag of keylock system, 372–375
jamming wireless alarms, 375–379
overview, 371–372
playing back IP camera stream
analyzing network traffic, 380–382
extracting video stream, 382–385
overview, 379–380
streaming protocols, 380
smart lock systems, 7–8
smartQuotesType property, 354
smart treadmills, attacking
Android operating system for, 386–387
escalating privileges, 394
injuries due to, 400
installing APKs, 391–393
overview, 385
remotely controlling speed and incline, 394–398
remote shell access, 391
software and physical buttons, disabling, 398–400
UI restrictions, circumventing, 387–390
SMS messages, forcing devices to send, 351–352, 365–366
Snapshots folder, 352
sniff CDP mode, 66
sniffer macro, 204–205
sniffing LoRa traffic with CatWAN USB stick, 318–322
sniff mode, CDP, *66*
SNR (signal-to-noise ratio), *375*
SOAP (Simple Object Access Protocol), *120*
socket creation functions, 107–108
socket destruction functions, 107–108
socketserver framework, 142, 144
softmatch Nmap service probe directive, 72
software composition analysis, *52*
software defined radio (SDR), *375*–376
software fragmentation, *338*
software integrity control, *18*
software of treadmills, disabling, 398–400
software whitelisting, *18*
SOIC (small outline integrated) clips, 190, 193
speed of treadmills, remotely controlling, 394–398
SPI. *See* Serial Peripheral Interface (SPI)
spidering tools, *48*
spiflash.py script, 196
spoof CDP mode, 66
spoofing, *22*, *23*
on control server service, 23
on drug library, 24
on firmware, 25
on hardware, 27
MAC address, 283–285
on operating system, 25
on pump service, 28
on restrictive user interface, 22
STRIDE threat classification model, 18
spoof mode, CDP, *66*
spooftooph utility, 284

spoof with a pre-made packet mode, CDP, *66*

spreading factor, *313*, 316–317

spread spectrum, *308*

SQL injection attacks, *24*, 120, 354

Squashfs format, *71*

SRTP (Secure RTP) protocol, *385*

SRV record, 132–133, 135, 138–140, 144

SSL certificate authority (CA), 357

`sslports` Nmap service probe directive, 72

STA (station), *288*

stack-smashing protection, *346*

Stais, Ioannis, 385

standards, *8*–10

star-of-stars topology, *309*

start bit, UART, 158

state anti-hacking laws, 13

states, Ncrack, 79–80

static analysis

  of firmware filesystem contents, 213–216

  InsecureBankV2 app, 361–362

  OWASP iGoat app, 344–346

static patching

  jailbreak detection, avoiding with, 358–360

  root detection, avoiding with, 368–369

station (STA), *288*

stdnse library, 106

`step` command, 349

ST-Link programmer, *165*

  boot mode, selecting, 174–175

  connecting to computer, 179–180

  connecting USB to serial adapter, 178

  flashing and running Arduino program, 174–180

  UART pins, identifying with logic analyzer, 176–177

  uploading Arduino program, 175–176

STM32F103C8T6 (Black Pill)

  boot mode, selecting, 174–175

  coding target program in Arduino, 172–174

  connecting to computer, 179–180

  connecting USB to serial adapter, 178

  debugging target, 181–188

  flashing and running Arduino program, 174–180

  overview, 169–170, 412

  UART pins, identifying with logic analyzer, 176–177

  uploading Arduino program, 175–176

stop bit, UART, 158

`strcmp()` function, 185–186

streaming protocols, 380

STRIDE threat classification model, *18*

  attack trees, using, 28–29

  breaking architecture into components, 20–21

  identifying architecture, 19

  identifying threats

    control server service, 23–24

    drug library, 24

    firmware, 25–26

    operating system, 25

    overview, 21–22

    physical equipment, 26–27

    pump service, 27–28

    RUI, 22–23

  overview, 18–19

string library, 106

`string.pack()` function, 109–110

`string.unpack()` function, 109, 113–114

string values of application entity titles, extracting, 102

subscribers, in publish-subscribe architecture, 73

subtrees, adding to existing protocol trees, 102–103

SUID binaries, *394*

supplementary data, in RFID tags, *243*–244

supply chain attacks, *27*

(`-sV`)Nmap command, 68

SWD. *See* Serial Wire Debug (SWD)

switch spoofing attack, *61*–63

`switch` statement, 85

SWJ-DP (Serial Wire or JTAG Debug Port), *165*

Sybil attack, *32*

symbolic links, 216–*217*

synchronous communication protocol, *191*

`syslog` function, 230–231

system manuals, 37

T

table library, 106

Tag-Connect interface, *167*

tagged port, *60*

tag-to-reader communication, eavesdropping on, 260–261

TAGulator, 166–167

`tag` variable, 265

tampering, *22*

  on control server service, 23

  on drug library, 23

  on firmware, 26

  on hardware, 27

  on operating system, 25

  protection and detection, 41–42

  on pump service, 28

  on restrictive user interface, 22

  STRIDE threat classification model, 18

tamper-resistant hardware, 341

TAP (Test Access Port), *164*–165

Target Service, WS-Discovery, *145*–146

TCK (Test clock input), 164

TCP SYN flood attack, 94
tcpwrappedms Nmap service probe directive, 72
TDI (Test data input), 164–165
tdnse.get_script_args() function, 112
TDO (Test data output), 164–165
technological protection measures (TPMs), *12*–13
TEE (Trusted Execution Environment), 41, *341*
Test Access Port (TAP), *164*–165
Test clock input (TCK), 164
Test data input (TDI), 164–165
Test data output (TDO), 164–165
test environment, setting up, 75–76
test hook clips, 190, 194
testing
  dissectors, 104
  firmware update service, 232–233
  LoRa sender, 310–314
  mDNS poisoner, 144–146
testing methodology. *See* security testing methodology
Test mode select (TMS) input, 164–165
test point interfaces, 42
test probes, *159*
Test reset (TRST) input, 164–165
text messages, forcing devices to send, 351–352, 365–366
thick clients, *210–211*
thinning the binary, 355
threat modeling, *17*. *See also* STRIDE threat classification model
  common threats, 31–33
  DREAD Classification Scheme, 29–30
  issues in, 18
  other types of, 30–31
  overview, 17
timescales, 6
Time-to-Live (TTL) value, 132
timing markers, *177*
Titan M, *341*
TMS (Test mode select) input, 164–165
tool development, 93
tools. See specific tools
topics, in publish-subscribe architecture, 73
topology mapping, *44*
totalwaitms Nmap service probe directive, 72
TP-Link Kasa app, 366
TPMs (technological protection measures), 12–13
traditional versus IoT security, 5–6
Transfer Syntax, *111*
transistor-transistor logic (TTL), *168*
transmission power, *313*
Transmit (TX) port, UART, 159, 162, 178
treadmills, attacking. *See* smart treadmills, attacking
TRST (Test reset) input, 164–165
trunk link, *61*
trunk port, *60*
trust boundaries, *20*–21
Trusted Execution Environment (TEE), 41, *341*
trusted platform module, *341*
TrustZone, *341*
TTL (Time-to-Live) value, 132
TTL (transistor-transistor logic), *168*
TX (Transmit) port, UART, 159, 162, 178
TXT record, 133, 138–140, 144

U

UART. *See* Universal Asynchronous Receiver-Transmitter (UART)
UART bridge VCP driver, 310
Ubertooth One, 412
UDP_server, creating, 142
UEFI (Unified Extensible Firmware Interface) Secure Boot, *41*
UF2 (USB Flashing Format), *318*
UID (Unique Identifier), *244*, 255, 258
UI restrictions, circumventing, 387–390
UK Code of Practice, 14
Ultra Narrowband (UNB), *308*
Umap, 118, 413
  abusing UPnP through WAN interfaces, 126–131
unencrypted communication channels, 234
Unified Extensible Firmware Interface (UEFI) Secure Boot, *41*
Unique Identifier (UID), *244*, 255, 258
Universal Asynchronous Receiver-Transmitter (UART), *157*
  baud rate, identifying, 162–163
  hacking devices through
    coding target program in Arduino, 172–174
    debugging environment, setting up, 170–172
    debugging target, 181–188
    flashing and running Arduino program, 174–180
    STM32F103C8T6 target device, 169–170
    tools for, 168–169
  hardware tools for communicating with, 158–159
  overview, 157
  packet format, 158
  ports, identifying, 159–162, 176–177
Universal Plug and Play (UPnP)
  abusing through WAN interfaces, 126–131
  common vulnerabilities, 120–121

    history of vulnerabilities, 118
    other types of attacks, 131
    overview, 118
    punching holes through firewalls, 121–126
    UPnP stack, 119–120
`u` parameter, 255
update mechanisms, firmware. *See* firmware update mechanisms
`upload` command, 399
UPnP. *See* Universal Plug and Play (UPnP)
UPnProxy, 118
URL schemes, 344–*345*, 351–352
USB Flashing Format (UF2), *318*
USB ports, assessment of, 40–41
USB-to-serial adapter, *168*, 176–177
USB-to-serial interface. *See* Bus Pirate
user accounts, testing, 51
user authentication, mobile app, 340–341
User Info Context, A-ASSOCIATE request message, 111
user knowledge, 39–40
user-level segregation, *49*–50
username enumeration, *49*
user passwords, resetting, 363–365
user security awareness, 32–33
USRP, 413
UUIDs, 278

V

`validate()` function, 174, 184–187
Valsamaras, Dimitris, 385
variable header, MQTT *CONNECT* packet, *80*–82
variable-length fields, parsing, 103–104
Vcc (Voltage) port, UART, 159, 162
`vconfig` command, 63
vendors, obtaining firmware from, 208–209
version intensity, 68
Vibease Wireless Remote Vibrator app, 367
video management servers, *145*
    attacks on, 152–153
    faking network cameras, 147–152
Vim, 315
Virtual Local Area Networks (VLANs), *60*–61
VLAN-hopping attacks
    double tagging attacks, 63–65
    imitating VoIP devices, 65–67
    overview, 60
    switch spoofing attacks, 61–63
VLAN tagging, *61*
VMware, 122–123
Voice over Internet Protocol (VoIP) devices, imitating, 65–67
VoIP Hopper, 65–67, 413
Voltage (Vcc) port, UART, 159, 162
vulnerability scanning, 46
`VV` command, 359

W

WAN interfaces, abusing UPnP through, 126–131
WannaCry attack, 5–6
Watchdog Timer, 316–317
web applications, assessment of, 48–50
web application sessions, *49*
Web Services Dynamic Discovery (WS-Discovery), *145*
    crafting attacks, 152–153
    faking cameras on network
        analyzing requests and replies in Wireshark, 147–149
        emulating cameras, 149–152
        setting up, 147
WebView, 350
    circumventing UI restrictions on treadmills, 387–390
    XSS vulnerabilities, 353–354
`wget` command, 212, 226
whole firmware emulation, 218–221
Wi-Fi
    attacks against APs
        cracking into WPA/WPA2 Enterprise, 304–305
        cracking WPA/WPA2, 299–300
        overview, 299
    attacks against wireless clients
        association attacks, 291–295
        deauthentication attacks, 289–291
        denial-of-service attacks, 289–291
        overview, 288–289
        Wi-Fi Direct, 295–299
    general discussion, 287–288
    hardware for security assessments, 288
    testing methodology, 305–306
Wi-Fi Direct, attacks against, 295–299
Wi-Fi modem router hacking
    dynamic analysis, 221–223
    extracting filesystem, 212
    firmware emulation, 216–221
    overview, 211–212
    statically analyzing filesystem contents, 213–216
Wifiphisher, 294–295, 297–298, 413
Wi-Fi Protected Access (WPA/WPA2), 47, 299–300
Wi-Fi Protected Setup (WPS), 296–297
Wired Equivalent Privacy (WEP), 47, 299
wireless alarms, jamming, 375–379
wireless clients, attacks against
    association attacks, 291–295
    deauthentication and denial-of-service attacks, 289–291

wireless clients, attacks against, *continued*
Evil Twin attack, 291–292
KARMA attack, 292
Known Beacons attack, 292–295
overview, 288–289
Wi-Fi Direct, 295–299
wireless protocol testing, 47
Wireshark
About Wireshark window, 98
additional documentation, 91
ADV_IND packet in, 271
DICOM traffic, generating, 97
Enabled Protocols window, 91–92
IP camera network traffic in, 380–381
Lua, enabling in, 97–99
Lua dissector, developing, 99–101
network traffic, analyzing with, 92
overview, 413
RTP stream, extracting, 383
SRV record in, 133
testing dissectors, 91
traffic dumps
of DHCP frame in voice network, 67
of MQTT *CONNACK* packet, 76
of MQTT CONNECT packet, 74
WS-Discovery requests and replies, analyzing in, 147–149
world-writeable logs, *23*
WPA Enterprise, cracking into, 304–305
WPA/WPA2 (Wi-Fi Protected Access), 47, 299–300
WPA/WPA2 four-way handshake, *299*–300
WPS (Wi-Fi Protected Setup), 296–297
`wrbl` parameter, 256
Write-Ahead-Logging mechanism, *354*
WS-Discovery. *See* Web Services Dynamic Discovery (WS-Discovery)

X

`xcodebuild` command, 343
Xcode IDE, 342
`xcode-select` command, 342
Xcrun, 351–352
`xcrun` command, 342–343
XSS injection attacks, 353–354
XXE (External Entity) attacks, *121*

Y

Yersinia, 61–63, 413
Yushkevich, I., 367

Z

ZAP (OWASP Zed Attack Proxy), 410
zero-configuration networking, *117*, *See also specific protocols*
Zipalign, 369