SYMBOLS
%n format specifier, 403

A
abootimg tool, 330
Abstract Namespace Socket, 165
access control mechanisms
(mitigations), 407–408
Access Point Name (APN), 137
Activities (Android applications),
36–37
Activities (IPC endpoint), 89–90
ActivityManager, 193–194
ad networks (attack surfaces), 146–147
ADB (Android Debugging Bridge)
access via TCP/IP, 140
ADB binaries, 227–228
ADB daemon, physical attacks via,
173
adb restore command race
condition, 80
adb root command, 218
adb daemon, 69
basics, 46–47
monitoring Android phones with,
386
tool, 63
ADBI framework, 492
Add Native Support menu item,
226–227
addresses
address lines, unexposed, 482
address space layout (kernels), 350
extracting (Linux kernel), 350–352
adjacency (networking), 137–139
Adleman, Leonard, 413
ADT Bundle, 213
ADT plug-in (Eclipse), 226, 486
Adventures in Bouncerland, 152
adware, 147
Aedla, 78
agent-proxy program, 346
ahh_setuid module, 324
AIDL (Android Interface Definition
Language), 51–52
alephzain, 80
allocated blocks, controlling heap
with (Android browser), 289–290
AllWinner SoC ARM core,
503
am command, 231
AndBug debugger, 112–113
Androguard framework, 95–96, 493–494

Android
Androguard framework, 95–96, 493–494
Androguard framework (Android-IA) project, 10
Android Secure Container (ASEC) files, 47
Android Studio, 487
Android-centric fork (Linux kernel), 49–50
AndroidManifest.xml file, 30, 35
Android.Troj.mdk Trojan, 151
application packages (APKs), 35
application Support Library, 17
applications, 34–39
building from source, 67
Compatibility Definitions, 63
Device Monitor, 212
dlmalloc allocator (heap exploitation), 269–271
emulator, 86
exposed UART on, 426–428
GDB binary, 245
heap debugging, 248–249
IDs (AIDs), 27–28
Interface Definition Language (AIDL), 51–52
logging system architecture, 53
Native Development Kit (NDK), 486
Software Development Kit (SDK), 93–94, 485–486
system architecture, 25–27
Update Alliance, 21
Android 4.0.1 linker case study (ROP)
executing arbitrary code from new mapping, 303–307
overview of, 300–301
pivoting stack pointer, 301–303
Android browser exploitation
controlling heap with allocated blocks, 289–290
controlling heap with CSS, 287–288
controlling heap with free blocks, 288–289
CVE-2011-3068 bug, 284–287
Android Developer Tools (ADT) bundle, 486–487
plug-in, 212
Android ecosystem
company history, 2
compatibility requirements, 17–18
complexities of, 15–16
device pool, 4–6
fragmentation of, 16
open source components, 7
public disclosures, 22–23
security vs. openness, 21–22
stakeholders. See stakeholders,
Android
update issues, 18–21
version history, 2–4
Android Framework
basics, 39–40
licensing, 12
overview of, 26
Android telephony stack
basics, 370–371
customization of, 371–372
AndroProbe, 246
Anonymous Shared Memory
(ashmem) (Linux kernel), 52, 167
anti-reversing epoxies, 482
aobj ARSCParser object, 106
AOSP (Android Open Source Project)
custom kernels for AOSP-supported devices, 325–326
getting kernel source, 317–319
Git repositories, 501–502
indexes of AOSP source code, 510
initializing, 215
native code debugging with, 227–233
native code debugging with non-AOSP devices, 241–243
Nexus devices supported by, 5
prebuilts directory, 229
Apache Ant, 223
Apache HTTP client libraries, 39
Apache Software License, 7
API permissions, 32
apktool (Java tool), 94, 494
app markets, 13
app permissions, 27, 84–86
Application Framework components (RIL), 371
application layer (OSI model), 136
application processor (smartphones), 369
application security
app permission issues, 84–86
information leakage through logs, 88–89
insecure data storage, 87–88
insecure transmission of sensitive data, 86
mobile security (app case study). See mobile security app (case study)
overview of, 83–84
SIP client (case study). See SIP client (case study)
unsecured IPC endpoints, 89–91
application testing tools, 496
app-locked device screen, 120
app.provider.query module, 125
apps
debugging with NDK, 222–226
malicious, 149
Verify Apps feature (Google), 150–151
web-powered mobile (attacks), 145–146
argv array, 281–282
Arithmetic Logic Unit (ALU) status flags, 341
ARM architecture
ABIs used on, 264
ARM ABI (Application Binary Interface), 295
ARM Linux debugger, 207–208
ARM9TDMI implementation, 292
licensing and designs, 10
ROP on. See ROP on ARM
separate instructions and data caches, 292–294
SOC families in ARM devices, 11
subroutine calls (ROP on ARM), 295–297
arm-eabi compiler, 322
ARP spoofing, 138
ashmem (Anonymous Shared Memory) (Linux kernel), 52
ASLR (Address Space Layout Randomization)
basics, 398–400
overcoming, 418–419
asroot exploit, 74
Asus
ASUS Transformer Prime, 79
open source repositories, 506
stock firmware (kernels), 312
attack phase (mobile security app), 117–120
attack surfaces (Android)
basics, 131–132
classifying, 134
local attack surfaces. See local attack surfaces
physical. See physical attack surfaces
physical adjacency, 154–161
remote. See remote attack surfaces
surface properties, 133
third-party modifications, 174
attacks
attack vectors, 130–131
overview of, 129–130
root access. See root access attack
history
automating
GDB client, 235
on-device tasks, 233–234

B
Babel fish, JTAG, 437
back-porting, 20
backtrace GDB command, 252
Baker, Mike, 74
Baksmali disassembler, 493
Barra, Hugo, 20
Baseband Attacks: Remote Exploitation
of Memory Corruptions in Cellular
Protocol Stacks, 480
baseband communication, rild
interaction with, 375
baseband interface (smartphones), 167
baseband processors (attack surfaces),
156–157
basebands (smartphones), 369
Bassel, Larry, 410
BCM3349 series chip, 447
Beagle device (Total Phase), 464
Beagle I2C (Total Phase), 498
Beagle USB (Total Phase), 498
beaming data, 159
Bergman, Neil, 88
bin arrays, 270
binaries, altering (exploit mitigations),
416–417
Binder driver (Linux kernel)
attack surfaces, 166–167
basics, 50–52
IPC and, 310
Binwalk, 487
binwalk tool, 316, 475
Bionic C runtime library (Android),
248
Bionic library, 42
Block View tool, 461
blown debug interfaces, 480
Bluetooth (attack surfaces), 157–158
BluetoothOppService, 38
Board Support Packages (BSPs),
502–503
boot command, 332
boot images
creating, 329–331
extracting kernels from, 315
boot loaders
boot partition (NAND flash
memory), 58
locked/unlocked, 62–65
passwords/hot keys/silent terminals,
480–481
rooting with locked/unlocked, 65–69
U-Boot, 468–469
unlock tools, 70
boot partitions
flashing, 333
going images of, 310–311
NAND flash memory, 58
recovery partition and, 314, 329–330
writing directly to, 334–335
boot process, 60–62
booted systems, gaining root access
on, 69
boot.img file, 315
booting
custom kernels, 331–336
customized boot sequences, 481–482
Borgaonkar, Ravi, 142
Bouncer system (attack surfaces),
151–152
break command (AndBug), 116
breakpoints
interdependent, 250
setting in “Hello World” module,
347–348
Broadcast Receivers
basics, 37
fuzz testing. See fuzzing Broadcast Receivers
handling implicit Intent messages
with, 89
onReceive method and, 101
browser attack surfaces, 143–145
browser exploitation, Android. See
Android browser exploitation
BrowserFuzz, 188, 193–194, 197
Bus Pirate device, 465–468, 497
bus resets (USB devices), 198
Busybox binary, 165–166, 491
BusyBox tool, 231
Butler, Jon, 190

C
The C++ Programming Language
(Addison Wesley), 272
C++ virtual function table pointers,
271–273
Caches
cache partition (NAND flash memory), 59
instructions and data (ARM), 292–294
calloc function, 395
canhazaxs tool, 162–163
carriers (stakeholders), 12
Case, Justin, 87
cat binary on Android, 400
CDD (Compatibility Definition Document), 18
cellular modem (smartphones), 369
certificate pinning, 146
Chainfire SuperSU, 66
chip passwords, 480
Chip Quik, 472, 498
chips, removing, 471–474
Chrome for Android browser
fuzzing. See fuzzing Chrome for Android
Google Play updates for, 144–145
client-side attack surfaces, 143–148
coalescing with blocks, 270–271
code
behind sockets, finding, 165–166
Code Aurora forum (Qualcomm), 23
Code Division Multiple Access (CDMA), 154
code signing, 392–394, 422
Common Attack Pattern Enumeration and Classification (CAPEC), 130
Common Vulnerabilities and Exposures (CVE) project, 23, 352–353
Common Vulnerability Scoring System (CVSS), 130
company history (Android), 2
Compatibility Definition Document (CDD), 327
compatibility requirements (Android), 17–18
Compatibility Test Suite (CTS), 349
Complex Instruction Set Computing (CISC), 299
components, identifying hardware, 456–458
CONFIG_KALLSYMS configuration option, 350
CONFIG_SEC_RESTRICT_FORK kernel option, 412
CONFIG_SEC_RESTRICT_SETUID kernel option, 412
CONFIG_SECURE_MEMORY_RWX kernel configuration, 410–411
Configurations
configuring kernel, 321–322, 349
configuring parameters for enabling KGDB, 344
and defenses (networking), 136–137
Package on Package (PoP), 458
Conover, Matthew, 394
consumers, features desired by, 14
ContainerNode HTML element, 257
Content Providers
 basics, 38–39
discovery of URIs (SIP client), 121–122
 exported attribute of, 413
 vulnerability of, 89
Cook, Kees, 409, 421
core services
 Android Debugging Bridge (ADB), 46–47
debuggerd, 46
 init command, 42–44
 other services, 47–49
 overview of, 42
 Property Service, 44–45
 Radio Interface Layer (RIL), 45–46
 Volume Daemon (vold), 47
Cowan, Crispin, 400
-crash dumps, debugging with, 208–211
-crash reports, kernel (debugging), 337–338
cross-site request forgery (CSRF or XSRF), 144
cross-site scripting (XSS), 144
CSipSimple application, 120–127
CSS, controlling heap with (Android browser), 287–288
CTS (Compatibility Test Suite), 18
Current Program Status Register (CPSR), 242, 296
custom allocators (heap exploitation), 269
custom debugger, writing, 245
custom firmware teams, 509
custom hardware interfaces, 479
custom kernels
 booting, 331–336
building, 325–329
configuring kernel, 321–322
creating boot images, 329–331
obtaining source code, 316–320
setting up build environment, 320–321
using custom kernel modules, 322–325
custom recovery images, 63–65
custom ROMs, 13–14
customized boot sequences, 481
CVE-2011-3068 bug (Android browser), 284–287
CyanogenMod, 13
Cydia Substrate for Android, 493

D
Dalvik code debugging
attaching to Dalvik processes, 220–221
debugging existing code, 217–221
faking debug devices, 218–220
“Hello World” app example, 213–215
overview of, 212–213
showing framework source code, 215–217
Dalvik Debug Monitor Server (DDMS), 212
Dalvik virtual machine
 basics, 40–41
 Java Virtual Machine and, 98
 overview of, 26
data
 binary/proprietary, 479–480
data cache (ARM), 292–294
Data Center interface, 464
data link layer (OSI model), 136
insecure storage of, 87–88
insecure transmission of sensitive, 86
overcoming execution problems, 419
preventing execution of, 396–398
data vulnerability
I²C/SPI/UART serial interfaces, 463–469
overview of, 459–460
USB interfaces, 459–463
Debootstrap, 245
debug interfaces
connecting to custom UARTS, 455
Finding JTAG Pinouts, 452–456
Finding SPI and I²C Pinouts, 451–452
finding UART pinouts, 447–451
logic analyzers, 444–447
overview of, 443–444
debuggers
debugger daemon, 46, 195
JTAG, 438–439, 471
KGDB, 343–348
debugging
alternative techniques for, 243–246
anti-debugging, 482
blown debug interfaces, 480
with crash dumps, 208–211
Dalvik code. See Dalvik code
debugging
debug statements, 243–244
Dynamic Binary Instrumentation (DBI) method, 245–246
gathering available information, 205–207
mixed code, 243
native code. See native code
debugging
on-device debugging, 244–245
remote, 211–212
toolchain selection, 207–208
debugging Linux kernel
live debugging with KGDB
debugger, 343–348
obtaining kernel crash reports, 337–338
Oops crash dumps, 338–343
overview of, 336–337
decompressing kernels, 316
defense in depth, 400
defenses and configurations
(networking), 136–137
developers, 13–14
development tools, 485–487
devices
automating on-device tasks, 233–234
custom kernels for AOSP-supported, 325–326
device mode services, fuzzing, 198
device pool (Android), 4–6
dismantling (attack surfaces), 169
extracting kernels from, 314–315
Facedancer, 463
faking debug devices, 218–220
fuzz testing on Android, 181–182
host mode, 198
interfacing with hardware, 424
JTAGulator device, 453–455
manufacturers, 11
native code debugging with non-AOSP, 241–243
RIL daemon on, 372–374
USB, 460
dex2jar project, 494
DHCP attacks, 138
diaggetroot exploit (Diag), 81
Discretionary Access Control (DAC), 407–408
dismantling devices (attack surfaces), 169
dlmalloc allocator, Android (heap exploitation), 269–271
dlmalloc memory allocator, 394
dmesg_restrict kernel setting, 409
DNS attacks, 138
do_ioctl function (Levitator), 358
document/media processing (attack surfaces), 147
domain parameter (sockets), 164
Donenfeld, Jason A., 78, 283
doPost method, 96
download mode, accessing, 61–62
Drake, Joshua J., 160, 162, 400
Drewry, Will, 395
drive-by attacks, 144
Drozer (Mercury) framework, 121, 496
dumb-fuzzing, 179–180
dynamic analysis (mobile security app), 109–117
Dynamic Binary Instrumentation (DBI)
framework, 492
method, 245–246
dynamic linkers, 300

**E**

Eclipse
attaching to system_process, 220–221
debugging “Hello World” with, 213–217
native code debugging with, 226–227
overview of, 486
EEPROM (Electrically Erasable Programmable Read-Only Memory), 470
electronic mail (attack surfaces), 148
emulator, Android, 86
endpoints (USB interfaces), 171–172
eng build configuration, 217
epilogue code, 264
epoxies, anti-reversing, 482
Etoh, Hiroaki, 401
evaluation kits, 442–443
exec-shield technique, 396
Executable and Linker Format (ELF)
binary, 228
explicit Intents, 89
Exploid exploit (udev daemon), 74
exploit mitigations
access control mechanisms, 407–408
address space layout randomization (ASLR), 398–400
classifying, 392
code signing, 392–394
disabling mitigation features, 415–417
format string protections, 401–403
fortifying source code, 405–407
future of, 420–422
hardening measures, 411–414
heap hardening, 394
history of core Android mitigation support, 414–415
kernel protection, 408–411
overcoming, 418–420
overview of, 391
preventing data execution, 396–398
protecting against integer overflows, 394–396
read-only relocations, 403–404
sandboxing, 404–405
stack protection, 400–401
exploiting Linux kernel
extracting addresses, 350–352
levitator.c. See levitator.c
exploit (case study)
Motochopper, 356–358
overview of, 348
sock_diag bug case study, 352–356
typical Android kernels, 348–350
exposed network services, 140–141
exposed serial interfaces, 426–428
extracting addresses (Linux kernel), 350–352
extracting kernels
basics, 310–311
from boot images, 315
decompressing kernels, 316
from devices, 314–315
from stock firmware, 311–313
Exynos (Samsung), 505–506
exynos-abuse exploit (Exynos 4 processor), 80–81

**F**

Facedancer device, 463, 498
factory images (Nexus devices), 5
Index

G–G 531

faking debug devices, 218–220
fastboot
booting kernels with, 332–333
protocol, 61–62, 67
utility, 487–488
file permission attacks, 79
file system (attack surfaces), 162–163
file system permissions (Unix), 32–34
filters, intent, 36
Firefox browser for Android, 88
firmware
accessing unobtrusively, 469–471
custom firmware teams, 509
destructively accessing, 471–474
extraction/flashing tools, 487–491
image storage, 471
First-In-Last-Out (FILO) lists, 274
flashing (download mode), 61
FLIRT (Fast Library Identification and Recognition Technology), 477–478
Force Close modal dialog, 187
format string protections, 401–403
FormatGuard: Automatic Protection From printf Format String Vulnerabilities, 401
FORTIFY_SOURCE mitigation, 405–407
fragmentation of Android ecosystem, 16
Framaroot one-click rooting
application, 80
Framework sockets, 279–280
free blocks, controlling heap with
(Android browser), 288–289
Freeman, Jay, 78, 283
FreeType library, 42
full retro, 404
fuzz testing
on Android devices, 181–182
background, 177–179
crafting malformed inputs, 179–180
eмуllating modem for, 379–382
identifying targets, 179
monitoring test results, 181
processing inputs, 180–181
SMS on Android, 382–390
fuzzing Broadcast Receivers
delivering inputs, 185
generating inputs, 184–185
identifying targets, 183–184
monitoring testing, 185–188
fuzzing Chrome for Android
generating inputs, 190–192
monitoring testing, 194–197
overview of, 188
processing inputs, 192–194
selecting technologies to target, 188–190
fuzzing USB attack surfaces
challenges, 198
generating inputs, 199–201
monitoring testing, 202–204
overview of, 197–198
processing inputs, 201–202
selecting target mode, 198–199

G
gadgets
combining into chains (ROP on ARM), 297–299
Gadget Framework, 172
gadget stack chaining, 294–295
identifying potential (ROP on ARM), 299–300
from leaf procedures, 298
master, 302–303
Galaxy Nexus, 140, 229, 336
GDB builds, custom, 245
GDB client
auto-generated script for, 223–224, 226
automating, 235
connecting to GDB Server, 230–232
gdbclient command, 232–233
non-AOSP devices and, 242
symbols and, 237–240
generate_assignment function, 192
generate_var function, 191
generative methods (smart-fuzzing), 180
Gerrit code review system (Google), 9, 13, 502
get_symbol function (Levitator), 358
getNeighboiringCellInfo method, 85
getpwuid function, 29–30
getList function, 266
getString method, 103–104, 117
gfree exploit, 70
giantpune, 81
GingerBreak exploit, 76–77, 275–279
Git repository, 319
Global Offset Table (GOT), 278–279
Global System for Mobile communications (GSM), 154
GNU Public License (GPL), 42
GoodFET device, 468, 497
Goodspeed, Travis, 456, 463, 468
Google
ClientLogin authentication, 86
Google Glass, 4, 161
Google Play, 9
Nexus devices, 4–5, 62–63
open source repositories, 501–502
role as Android brand owner, 8–10
Single Sign On (SSO) system, 148
Google infrastructure (attack surfaces)
Bouncer system, 151–152
Google Play, 148–149
GTalkService, 152–154
malicious apps, 149
overview of, 148
third-party app ecosystems, 149–151
Google Play
app developers and, 17
as remote attack surface, 148–149
GOT (Global Offset Table), 403
GPS (attack surfaces), 155–156
Grand, Joe, 453
grep, 94, 112
GSM (Global System for Mobile communication), 142
GSM AT command-based vendor-RIL, 380–381
GTalkService (attack surfaces), 152–154
A Guide to Kernel Exploitation: Attacking the Core, 348
gzip command, 316
H
Hacking Exposed Wireless, 158
half-day vulnerabilities, 21, 145
handleBlockEvent in void
implementation, 276
handlePartitionAdded function,
276–278
hardening measures, 414–414, 420–421
hardware
breakpoints, 250
hacking tools, 496–499
support services (attack surfaces), 168
vendors (stakeholders), 10–12
hardware attacks
accessing firmware unobtrusively,
470–472
analyzing binary image dumps,
474–478
anti-reversing epoxies, 482
binary/proprietary data, 479–480
blown debug interfaces, 480
boot loader passwords/hot keys/
silent terminals, 480–481
chip passwords, 480
custom hardware interfaces, 479
customized boot sequences, 481
destructively accessing firmware,
471–474
finding debug interfaces. See debug
interfaces
IPC interface, 428–431
identifying components, 456–458
image encryption/obfuscation/anti-debugging, 482
intercepting/monitoring/injecting data. See data vulnerability
interfacing with hardware devices, 424
JTAG. See JTAG (Joint Test Action
Group)
One-Wire (1-Wire) interface, 428–431
overview of, 423–424
pitfalls, 479–482
SPI interface, 428–431
UART (Universal Asynchronous
Receiver/Transmitter) serial
interfaces, 424–428
unexposed address lines, 481
heap debugging, Android, 248–249
heap exploitation
Android dlmalloc allocator, 269–271
C++ virtual function table pointers,
271–273
custom allocators, 269
RenderArena allocator, 273–275
use-after-free scenarios, 268–269
heap hardening, 394
heap memory, kernel, 349–350
heap.txt file, 394
Heimdall command-line tool, 488
Heimdall open source program, 334
“Hello World” app (Dalvik code
debugging), 213–215
Hex-Rays Decompiler, 496
hooking/instrumentation tool, 492–493
host mode (devices), 198
hosts, USB, 460, 462–464
Hotz, George, 431
HTC
HTCJ Butterfly device, 81
open source repositories, 507
stock firmware (kernels), 312
tools, 489–490
HTML5, 189
I
I²C (Inter-Integrated Circuit) serial
interface
basics, 428–431
finding pinouts, 451–452
sniffing, 464–465
IDA (Interactive Disassembler)
IDA Pro tool, 156, 207
importing binary images into,
476–478
overview of, 496, 499
IDCODE scans, 454–455
IEI (Information Element Identifier),
378–379
image encryption/obfuscation/anti-
debugging, 482
implicit intents, 36, 89
init command (Linux), 42–44
init configuration files, 174
init process, 60
Injecting SMS Messages into Smart
Phones for Vulnerability Analysis, 380
injection (SIP client), 125–126
Injectord (SMS message injection),
382–386
inputs (fuzzing)
comparing/minimizing (root cause
analysis), 247–248
crafting malformed, 179–180
delivering (Broadcast Receivers),
185
generating (Broadcast Receivers),
184–185
generating (Chrome for Android),
190–192
generating (USB attack surface),
199–201
processing (Chrome for Android),
192–194
processing (USB attack surface), 201–202
processing overview, 180–181
insecure data transmission, 86
Inside the C++ Object Model (Addison-Wesley), 272
insmod command, 324–325
instructions cache (ARM), 292–294
integer overflows, protecting against, 394–396
Intel, 503
Intel Android Developer, 10
IntentFuzzer application, 183, 184–185
Intents (Android applications), 35
interdependent breakpoints, 250
INTERNET permission, 32
Internet structure, 135
interworking (modes), 296
IPC
permissions, 34
unsecured endpoints, 89–91
iSEC Intent Sniffer/Intent Fuzzer tools, 496
isPinLock, 103, 115

J
jad (Java Decompiler), 494–495
Jasmin assembly format, 493
Java
  Debug Wire Protocol (JDWP), 112, 212
  Native Interface (JNI) method, 222
  Virtual Machine, 98
JD-GUI Java decompiler, 495
JEB decompiler, 495
Jelinek, Jakub, 403, 405
jigs (cables), 455
J-Link debugger (Segger), 438–439, 497
JTAG (Joint Test Action Group)
  Babel fish, 437
d debuggers, 438–439, 471
evaluation kits, 442–443
finding pinouts, 452–455
JTAG: Attacks and Defenses, 480
JTAGulator device, 453–455, 497
misconceptions, 432–437
OpenOCD (Open On Chip Debugger), 439–442
overview of, 431–432
Juice Jacking attacks, 173, 413

K
kallsymprint tool, 351
Karri, Ramesh, 480
kernel, Android Linux. See Linux
  Kernel (Android)
keexec program, 333
KGDB debugger, 343–348
Kies desktop software (Samsung), 488
Kies system application, 90
KillingInTheNameOf exploit (ashmem subsystem), 76
King, Russell, 309
kp_tr_restriction kernel setting, 409
Krahmer, Sebastian, 74–76
Kralevich, Nick, 412

L
Lais, Christopher, 74
Lanier, Zach, 84
Larimer, Jon, 77, 358
launchMode attribute, 37
ldpreloadhook tool, 492
Lea, Doug, 394
least privilege principle, 55
levitator exploit (PowerVR driver), 77
levitator.c exploit (case study)
determining root cause, 360–362
fixing the exploit, 362–364
getting source code, 360
overview of, 358–359
running existing exploit, 359–360
LG
LGBinExtractor command-line tool, 489
mobile support tool, 489
open source repositories, 507
Optimus Elite (VM696), 60–61
stock firmware (kernels), 313
lib.so binary, 406
libraries (user-space native code), 41–42
libsysutils library, 279
Linaro project, 510
Linux
capabilities, 28
exposed UART on, 426–428
Linux kernel (Android)
debugging. See debugging Linux kernel
exploiting. See exploiting Linux kernel
extracting kernels. See extracting kernels
future hardening efforts, 420–421
A Guide to Kernel Exploitation: Attacking the Core, 348
A Heap of Trouble: Breaking the Linux Kernel SLOB Allocator, 350
overcoming protections, 419–420
overview of, 309–310
protection of, 408–411
running custom kernel code. See custom kernels
telephony stack and (RIL), 371
tweaking configurable parameters, 417
Understanding the Linux Kernel, 339
Linux kernel modifications
Android-centric fork, 49–50
Anonymous Shared Memory (ashmem), 52
Binder driver, 50–52
logger driver, 53–55
Paranoid Networking, 55
pmem custom driver, 53
lit exploit (Diag), 81
loadable kernel modules (LKM)s, 322
local area networks (LANs), 137–138
local attack surfaces
baseband interface (smartphones), 167
Binder driver (Linux kernel), 166–167
file system, 162–163
hardware support services, 168
overview of, 161
shared memory, 167
sockets, 164–166
system calls, 163
locked boot loaders, 68–73
locked/unlocked boot loaders, 62–65, 393
logcat, 109
logger driver (Linux kernel), 53–55
logic analyzers, 444–447
logically (network) adjacent relationships, 137
logs
information leakage through, 88–89
kernel, 337
lsusb and libusb library, 171–172

M
main buffer (logger), 53
main function (Levitator), 358
Makris, Andreas, 80
malicious apps, 149
managers, Android Framework, 39–40
Mandatory Access Control (MAC), 407
Man-in-the-Middle (MitM) attacks, 86, 138, 144
market share, Android, 5
Marvell, 503–504
master gadgets, 302
Media Access Control (MAC) addresses, 138
Media Transfer Protocol (MTP) specification, 199–201
media/document processing (attack surfaces), 147
MediaTek, 504
mem_write function (Linux kernel), 78
memcpy implementation, 301, 304–305
memory corruption exploits
heap exploitation. See heap exploitation
overview of, 263–264
stack buffer overflows, 264–267
MemoryFile class, 52
mempodroid exploit (Linux kernel), 78–79, 283–284
MicroSD cards for firmware storage, 471
Miller, Barton, 177
Miller, Charlie, 152, 160, 380, 431
Miner, Rich, 2
MIPS Technologies, 11
mixed code debugging, 243
mkbootimg utility (AOSP), 315
mmap function, 303–304
mmap system calls, 398–399
mobile apps, web-powered (attacks), 145–146
mobile security app (case study)
attack phase, 117–120
dynamic analysis, 109–117
overview of, 91
profiling phase, 91–93
static analysis phase, 93–109
mobile technologies (attack surfaces), 142
modems
emulating for fuzzing, 379–382
fuzzing SMS on Android, 382–390
modular arithmetic, 395
modules, custom kernel, 322–325
monitoring
fuzz testing results, 181
fuzz testing results (Broadcast Receivers), 185–188
fuzz testing results (Chrome for Android), 194–197
fuzz testing results (USB attack surfaces), 202–204
Motochopper exploit (case study), 356–358
Motorola
open source repositories, 507
stock firmware (kernels), 313
tools, 490–491
Moulu, Andre, 90–91
Müller, Michael, 173
Mulliner, Collin, 246, 380
Multifunction Composite Gadget, 172
Multimedia Messaging Service (MMS), 142, 371
MultiMediaCard (MMC) block request filter, 71
mutation fuzzing, 247–248
mutation techniques (dumb-fuzzing), 179–180

N
NAND flash, 15
NAND flash memory partition layout, 58
NAND locks, 14, 70–71
native Android tools, 491–492
native code debugging
with AOSP, 227–233
with Eclipse, 226–227
increasing automation, 233–235
with NDK, 222–226
with non-AOSP devices, 241–243
overview of, 221
with symbols, 235–241
native code, user-space. See user-space native code
NAT-PMP protocol, 141
NDK (Android Native Development Kit)
developing user-space native code with, 10
native code debugging with, 222–226
revision 4b, 398
Netlink messages, 352
NETLINK sockets, 275
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>netstat command</td>
<td>141</td>
</tr>
<tr>
<td>Network Address Translation (NAT)</td>
<td>137</td>
</tr>
<tr>
<td>networking capabilities</td>
<td>55</td>
</tr>
<tr>
<td>concepts</td>
<td>134–139</td>
</tr>
<tr>
<td>exposed network services</td>
<td>140–141</td>
</tr>
<tr>
<td>network layer (OSI model)</td>
<td>136</td>
</tr>
<tr>
<td>network paths</td>
<td>135</td>
</tr>
<tr>
<td>on-path attacks</td>
<td>138–139</td>
</tr>
<tr>
<td>OSI (Open Systems Interconnection) model</td>
<td>135–136</td>
</tr>
<tr>
<td>physically adjacent relationships</td>
<td>137</td>
</tr>
<tr>
<td>stacks (Linux kernel)</td>
<td>139–140</td>
</tr>
<tr>
<td>Nexus devices (Google)</td>
<td>4–5, 162</td>
</tr>
<tr>
<td>Nexus factory images, extracting kernel from</td>
<td>311–312</td>
</tr>
<tr>
<td>NFC (Near Field Communication) technology (attack surfaces)</td>
<td>159–161</td>
</tr>
<tr>
<td>Nmap port scanner</td>
<td>141</td>
</tr>
<tr>
<td>non-vendor-specific libraries</td>
<td>42</td>
</tr>
<tr>
<td>Non-Volatile Random Access Memory (NVRAM)</td>
<td>70</td>
</tr>
<tr>
<td>null Intent fuzzing</td>
<td>187–188</td>
</tr>
<tr>
<td>nvflash tool (NVIDIA)</td>
<td>489</td>
</tr>
<tr>
<td>NVIDIA open source repositories</td>
<td>504</td>
</tr>
<tr>
<td>proprietary recovery mode</td>
<td>489</td>
</tr>
</tbody>
</table>

**O**

Oberheide, Jon | 77, 152, 154, 358 |
ODIN tool (Samsung) | 333–334, 488 |
OEMs devices, custom kernels for | 326–329 |
| devices, flashing boot partitions of | 333–336 |
| getting source code for | 319–320 |
| open source repositories | 506–508 |
| stock firmware (kernels) | 312–313 |
Oldani, Massimiliano | 348 |
Oliva, Paul | 220 |
on-device debugging | 244–245 |
One Laptop Per Child (OLPC) XO tablet | 504 |
One-Wire (1-Wire) serial interface | 428–431 |
on-path attacks (networking) | 138–139 |
onReceive method | 101–102, 114 |
On-the-Go (OTG) cable | 198 |
Oops crash dumps | 338–343 |
Opaque Binary Blobs (OBBs) | 47 |
Open Handset Alliance (OHA) | 2 |
Open Multimedia Applications Platform (OMAP) | 344 |
Open On-Chip Debugger (OpenOCD) software | 497 |
open source components (Android) | 7 |
Open Source Mobile Communications (Osmocom) project | 156–157 |
open source repositories custom firmware teams | 509 |
| Google | 501–502 |
| indexes of AOSP source code | 510 |
| individual sources | 510 |
| Linaro project | 510 |
| OEMs | 506–508 |
| overview of | 501 |
| Replicant project | 510 |
| SoC manufacturers | 502–506 |
| upstream sources | 508–509 |
opendir system call | 162 |
openness vs. security (Android) | 21–22 |
OpenOCD (Open On Chip Debugger) | 439–442 |
OpenSession operation code | 202 |
Optimized DEX files (ODEX) | 40–41 |
Ormandy, Tavis | 73 |
Ortega, Alfredo | 245 |
Osborn, Kyle | 173, 413 |
OSI (Open Systems Interconnection) model (networking) | 135–136 |
OTA (over-the-air) updates | 63 |
overgranting permissions | 85 |

**P**

Package on Package (PoP) configurations | 458–459 |
packages.xml | 31 |
PAGEEXEC technique | 396 |
pairing Android devices, 157
Paranoid Networking (Linux kernel), 55
Paris, Eric, 409
partial relo, 404
partition layouts (rooting), 58–60
passwords
  boot loaders, 480
  chip, 480
paths, network, 135
PDU (protocol data unit), 377, 389
Percoco, Nicholas, 152
Perla, Enrico, 348
permanent roots, 70–71
permissions
  Android, 30–34
  app, 27, 84–86
  READ_LOGS, 88
  UNIX file system, 32–34
  persisting soft-roots, 71–73
Personal Unblocking Key (PUK) (SIM cards), 142
personality system call (Linux), 416
PF_NETLINK socket, 165
PF_UNIX socket domain, 164–165
Phone applications component (RIL), 371
phone side SMS delivery, 382
PHP web app attack surfaces, 132
physical adjacency attacks, 154–161
physical attack surfaces
disassembly devices, 169
miscellaneous, 173–174
overview of, 168–169
USB wired interfaces, 169–173
physical layer (OSI model), 135
physically adjacent relationships
  (networks), 137
Pie, Pinkie, 190
pivoting stack pointers (Android linker case study), 301–303
platform keys, 35
pmem custom driver (kernel), 53
pointer and log restrictions (kernel), 409–410
Polaris Office application, 147
pop/push instructions (Thumb), 297
Position-independent executables (PIE), 416–417
POSIX functions, 29
power users, 14
pre-installed applications, 34–35
presentation layer (OSI model), 136
privilege reduction technique, 56
proc_register function, 364
process isolation technique, 56
processUnlockMsg, 105
profiling phase (mobile security app), 91–93
prologue code, 264
properties, attack surface, 133
Property Service, 44–45
ProPolice project, 401
protectionLevel attribute
  (signature), 36
ProtoBufs transport (Google), 152–153
Protocol Buffers (protobufs), 136
ps command, 173
psneuter exploit, 76
ptrace, 246
public disclosures (Android), 22–23
public exploits
  GingerBreak exploit, 275–279
  mempodroid exploit, 283–284
  overview of, 275
  zergRush exploit, 279–283
public-key cryptography, 35
pull-up resistors, 465
pure Google experience (Nexus devices), 5
push/pop instructions (Thumb), 297
PyUSB (Python), 201–202

Q
qlimit local variable, 281
Qualcomm, 505
Quick Response (QR) codes/voice commands, 161

R
Radare2 framework, 495
radio partition (NAND flash memory), 59
RageAgainstTheCage exploit (ADB daemon), 75
rand_num function, 192
READ_LOGS permission, 88
read-only memory regions (kernel), 410–411
Read-Only Relocations mitigation, 403–404
recovery images, stock/custom, 63–65
recovery partitions, 58, 314, 329–330
recovery.img file, 315
references & resources
by chapter, 511–522
general, 522
registerReceiver method, 37
Reiter, Andrew, 84
remote attack surfaces
client-side attack surfaces, 143–148
exposed network services, 140–141
Google infrastructure. See Google infrastructure (attack surfaces)
mobile technologies, 142
networking concepts, 134–139
networking stacks, 139–140
overview of, 134
remote debugging, 211–212
RenderArena allocator (heap exploitation), 273–275
RenderObject class, 287–289
RenderTree, 273
Replicant project, 510
repo tool (AOSP), 501–502
ret2libc technique, 294
Ridley, Stephen A., 447
RIL (Radio Interface Layer)
Android telephony stack, 370–372
architecture, 368–369
modem interaction. See modems overview of, 45–46, 367–368
RIL daemon (rild), 372–374
smartphone architecture, 369–370
SMS (Short Message Service). See SMS (Short Message Service) vendor-ril API, 374–375
Rivest, Ron, 413
Role-Based Access Control (RBAC), 407
ROMs, custom, 13–14
root access attack history
adb restore command race condition, 80
Exploid exploit (udev daemon), 74
exynos-abuse exploit (Exynos 4 processor), 80–81
file permission attacks, 79
GingerBreak exploit (vold daemon), 76–77
KillingInTheNameOf exploit (ashmem subsystem), 76
levitator exploit (PowerVR driver), 77
lit/diagetroot exploits (Dia), 81
mempodroid exploit (Linux kernel), 78–79
overview of, 73
RageAgainstTheCage exploit (ADB daemon), 75
symbolic link-related attacks, 79
Volez utility (recovery images), 74
Wunderbar/asroot bug (Linux kernel), 73–74
zerGRush exploit (libsutils), 78
Zysploit implementation (Zygote process), 75–76
root cause analysis
analyzing WebKit crashes, 250–260
Android heap debugging, 248–249
comparing/minimizing inputs, 247–248
interdependent breakpoints, 250
overview of, 246–247
watchpoints, 250
RootAdb app, 220
rooting devices
boot process, 60–62
gaining root access on booted systems, 69
locked/unlocked boot loaders, 62–65
NAND locks, 70–71
overview of, 57–58
partition layouts, 58–60
permanent roots, 70–71
persisting soft-roots, 71–73
root access attack history. See root access attack history
rooting with locked boot loaders, 68–73
rooting with unlocked boot loaders, 65–68
temporary roots, 70–71
ROP (Return Oriented Programming), 291–294
Android 4.0.1 linker (case study). See Android 4.0.1 linker case study (ROP)
history and motivation, 291–294
ROP on ARM
ARM subroutine calls, 295–297
basics, 294–295
combining gadgets into chains, 297–299
identifying potential gadgets, 299–300
Rosenberg, Dan, 79, 81, 356, 409
Rosenfeld, Kurt, 480
Rowley, Robert, 173, 413
RSD Lite tool (Motorola), 490
Rubin, Andy, 2
ruuveal utility (HYC), 490

S
safe_iop library, 395–396, 422
SAFEDROID project, 421
Saleae Logic Analyzer, 445–449, 497
Samsung
devices, flashing, 488
Galaxy Nexus, 59
Galaxy S III, 336
open source repositories, 505–506, 508
stock firmware (kernels), 313
sandboxing
Android’s sandbox, 27–30
basics, 404–405
future implementation of, 420
SAX XML parser, 39
sbf_flash utility (Motorola), 490
SBF-ReCalc tool (Motorola), 490–491
Scapy packet manipulation tool, 200
SD cards, 33–34, 471
sdcard_rw group, 28
sealime Loadable Kernel Module, 71
Sears, Nick, 2
SecureRandom class, 413
security
vs. openness (Android), 21–22
application. See application security
Google security announcements, 22–23
researchers, 15
RIL daemon and, 374
State of Security in the App Economy: Mobile Apps Under Attack, 150
updates, 19–20
Why Eve and Mallory Love Android: An Analysis of Android SSL (In)
Security, 146
security boundaries/enforcement
Android permissions, 30–34
Android’s sandbox, 27–30
overview of, 27
Segerdahl, Olle, 199–200
Segger J-Link debugger, 438–439
SELinux, 408
Sense and Touchwiz user interface, 12
Service Loading (SL) request, 142
Services, Android, 38
Services, unsecured (IPC endpoints), 89–90
session layer (OSI model), 136
setarch program, 416
setpropex system properties editor, 491
Shamir, Adi, 413
shared memory (attack surfaces), 167
sharedUserId attribute
   (AndroidManifest.xml), 35
sideload feature (Android 4.1), 67
SIGPIPE signal, 210
SIM-unlocked devices, 4
Single Sign On (SSO) system (Google), 148
SIP client (case study)
   discovery of Content Provider URIs, 121–122
   Drozer security testing framework, 121
   injection, 124–127
   overview of, 120
   snarfing, 122–125
Skip Operation packet, 203
Skype client for Android, 87–88
SLAB/SLUB allocators, 349–350
Smali assembler, 493
Smali format, 94
smart-fuzzing, 180
smartphone architecture, 369–370
Smashing the Stack for Fun and Profit, 265
SMS (Short Message Service)
   fuzzing SMS on Android, 382–390
Injecting SMS Messages into Smart Phones for Vulnerability Analysis, 380
   message format, 376–379
   overview of, 375–376
   phone side SMS delivery, 382
   Protocol Data Unit (PDU), 101
   Protocol Data Units (PDUs), 118–119
   sending/receiving messages, 376
   SmsReceiverService, 38
   using as attack vector, 142
   SMSC (Short Message Service Center), 376
   snarfing (SIP client), 122–124
   sniffing
      I2C/SPI/UART, 464–465
      USB, 460–462
   SoC manufacturers, 502–506
   sock_diag bug (case study), 352–356
   sockets (attack surfaces), 164–166
   soft root method, 69
   soft-roots, persisting, 71–73
   software breakpoints, 250
   S-ON lock feature, 412
   Sony
      mobile division open source repositories, 508
      stock firmware (kernels), 313
      source code, fortifying, 405–407
      source-level debugging (symbols), 240–241
   specifications of hardware components, 456–457
   Speers, Ryan M., 462
   Spengler, Brad, 74, 408, 421
   SPI (Serial Peripheral Interface)
      EEPROM memory, 470
      finding pinouts, 451–452
      serial interface basics, 428–431
      sniffing, 464–465
   splash partition (NAND flash memory), 58
   spoofing attacks, 138
   SQL injection, 126
   SQLite database engine, 491–492
   SQLite library, 42
   stacks
      networking (Linux kernel), 139–140
Smashing the Stack for Fun and Profit, 265
stack buffer overflows (memory corruption), 264–267
stack protections, 400–401, 418
StackGuard protection, 400–401
Stack-Smashing-Protector (SSP), 401
stakeholders, Android
  carriers, 12
developers, 13–14
Google, 8–10
  hardware vendors, 10–12
overview, 7–8
users, 14–15
stat system call
State of Security in the App Economy:
  Mobile Apps Under Attack, 150
statements, debug, 243–244
static analysis phase (mobile security app), 93–109
static analysis tools, 493–496
stock firmware, extracting kernels from, 311–313
stock recovery images, 63–65
stock ROMs, 313
storage of data, 87–88
strace utility (on-device debugging), 244, 492
strcpy function, 405
su binary, 65, 67
subroutine calls (ROP on ARM), 295–297
Subscriber Identity Module (SIM) cards, 137
SuperMUTT device, 463
SuperPro (Zeltek), 472–473, 498
Supervisor Mode Access Protection (SMAP), 421
Supervisor Mode Execution Protection (SMEP), 421
surface properties (attacks), 133
surfaces, attack. See attack surfaces (Android)
symbolic link-related attacks, 79
symbols
debugging ARM binaries with, 206–207
native code debugging with, 235–241
syringe technique (Goodspeed), 457
sysctls (kernel parameters), 417
system architecture, Android. See Android
system buffer (logger), 54
system calls (attack surfaces), 163
system logs, 208–209
system partition (NAND flash memory), 58
system_server process, 41
System-on-Chip (SoC) manufacturers, 11
T
table pointers, virtual function (vtable), 272
tagcode key, 108–109, 117
targets (fuzzing)
  basics of identifying, 179
  identifying (Broadcast Receivers), 183–184
  selected modes (SB attack surfaces), 198–199
  selected technologies (Chrome for Android), 188–190
T-bits, 296
TCP sequence number prediction, 140
telephony stack, Android. See Android
  telephony stack
temporary roots, 70–71
test points (PCBs), 456
Texas Instruments (TI), 504–505
third-party app ecosystems (attack surfaces), 149–151
third-party modifications (attack surfaces), 174
Thumb execution mode (ARM), 296–297, 299–300
Time Stamp field (SMS), 378
Tinnes, Julien, 73
T-Mobile G2, 71
tombstone files, 209–211
TOMOYO implementation, 408
toolchain selection (debugging), 207–208
Total Phase Data Center software, 460–462
TP-PID field (SMS), 377
transport layer (OSI model), 136
TriangleAway tool, 333
Typed Arrays feature (Chrome for Android), 189–192

U
UARTs (Universal Asynchronous Receiver/Transmitter) serial interfaces
basics, 424–428
connecting to custom, 455
finding UART pinouts, 447–451
sniffing, 464–465
U-Boot, 468–469, 480
UDH (User Data Header), 377–379
umask functionality, 412
undefined behavior, 247
undergranted permissions, 85
Understanding the Linux Kernel, 339
unexposed address lines, 481
Universal Flash Programmers, 472
Universal Software Radio Peripheral (USRP), 156
UNIX domain sockets, 275
UNIX file system permissions, 32–34
Unlimited.io exploits, 70–71
unlink technique, 394
unlock portals, 63
unlocked/locked boot loaders, 62–68
unruu utility (HTC), 490
Unstructured Supplementary Service Data (USSD) facilities, 142
update issues, 18–21
update packages, 64
UPnP protocol, 141
upstream repository sources, 508–509
USB interfaces, 459–463
USB Complete: The Developer’s Guide, 459
wired interfaces (attack surfaces), 169–173
use-after-free scenarios (heap exploitation), 268–269
User Data Header (UDH) (SMS), 378–379
userdata partition (NAND flash memory), 58
userdebug builds, 217
user-installed applications, 34–35
users, Android, 14–15
user-space components (RIL), 371
user-space native code
core services. See core services libraries, 41–42
user-space software
exploiting Android browser. See Android browser exploitation
memory corruption exploits. See memory corruption exploits
public exploits. See public exploits

V
valgrind tool, 181
vectors, attack, 130–131
vendors
balancing security and openness, 21–22
vendor-ril API, 372, 374–375, 380–381
vendor-specific libraries, 42
Verify Apps feature (Google), 150–151
versions, Android
adoption rate, 6
back-porting, 20
history of, 2–4
openness of, 7
versions, kernel, 348–349
virtual functions, 271–273
Virtual Private Networks (VPNs), 137
vold daemon, 275
Volez utility (recovery images), 74
Volume Daemon (vold), 47
Volume Manager daemon, 78
VolumeManager dispatcher class, 276
Von Neumann architecture, 396
vulnerability analysis
determining root causes. See root
cause analysis
judging exploitability, 260–261
overview of, 246

W
Walker, Scott, 71, 76
watchpoints (breakpoints), 250
watering hole attacks, 144
WebKit
analyzing crashes, 250–260
library, 42, 236
RenderArena allocator, 273–275
specific allocator (RenderArena), 273
virtual function call example, 272–273
web browser engine, 21
web-powered mobile apps (attack
surfaces), 145–146
websites for downloading
Android Debug Bridge (ADB) tool, 63
Android Dynamic Binary
Instrumentation Toolkit (adbi), 246
apktool, 94
ARM9TDMI™ Technical Reference
Manual, 292
boot loader unlock portal, 66
boot loader unlock tools, 70
catalog of tools, 485–499
Chainfire SuperSU, 66
Compatibility Definition Document
(CDD), 18
dashboard updates, 5
diaggetroot exploit, 81
exynos-abuse exploit, 80
fastboot client utility, 61
fuzzing MTP tool, 199
gfree exploit, 71
GingerBreak exploit, 76
Injectord source code, 380
JTAGulator schematics/firmware,
453–454
KillingInTheNameOf exploit, 76
levator exploit, 77
LGExtract tool, 313
library for creating SMS messages,
383
mempodroid exploit, 79
MIPS Technologies, 11
open source repositories, 501–510
patch for setting breakpoints, 224
psneuter exploit, 76
RootAdb app, 220
TriangleAway tool, 333
Unlimited.io exploits, 70
zergRush exploit, 78
websites for further information
3GPP SMS standard, 376
adb command, 47
Android code names/tags/build
numbers, 2
Android Compatibility Definitions,
63
Android market share, 5
Android on Intel Architecture
(Android-IA) project, 10
Android security issues, 22
AOSP, initializing, 215
Apache Software License, 7
ARM, 11
Bluetooth subsystem in Android, 158
browser compatibility, 189
Common Weakness Enumeration
(CWE) project, 246
Dalvik documentation, 41
Debootstrap, 245
device-specific repositories, 317
Facedancer21 units, 463
factory images for Nexus devices, 311
Google ClientLogin findings, 86
GSM AT command set, 375
A Heap of Trouble: Breaking the Linux Kernel SLOB Allocator, 350
heaptut.txt file, 394
Intel Android Developer, 10
Java Debug Wire Protocol (JDWP), 212
jigs for Android devices, 455
Linux capabilities, 29
Linux kernel documentation, 410
Mozilla bug tracker, 89
native Android GDB binaries, 245
Nexus 4 cable, 455
NFC on Android, 159
OHA members, 2
permission mapping research, 85
Replicant project, 375
seccomp-bpf sandbox on Android, 420
security tests (CT), 18
SMS online services, 386
SMS standard, 379
Wunderbar emporium exploit, 74
WebView browser engine, 146–147
Weimer, Florian, 395
Weinmann, Ralph Phillip, 480
weird machine programming, 264
White, Chris, 2
Wicherski, George, 160, 246, 400
wide area networks (WANs), 137
WiFi networks (attack surfaces), 158–159
Wi-Fi Protected Access (WPA), 158
WiFiManager class, 84
Wired Equivalent Privacy (WEP), 158
Wireless Application Protocol (WAP), 142
wireless communications (attacks)
  baseband processors, 156–157
  Bluetooth, 157–158
  Google Glass, 161
  GPS, 155–156
  NFC communications technology,
    159–161
  overview of, 154–155
  WiFi networks, 158–159
  Wise, Joshua, 76
write-four primitives, 278
Wunderbar/asroot bug (Linux kernel), 73–74

X
Xeltek devices, 472–473
XN exploit mitigation, 292
Xperia Firmware, 313
XPosed framework, 492–493

Y
Yet another free() exploitation technique, 271

Z
zergRush exploit, 78, 279–283, 418
zero page protection (kernel), 410
zImage binary file, 310
Zimperlich exploit (Zygote process), 75–76
Zygote process, 41, 87, 419
Zysploit implementation (Zygote process), 75–76