Index

A

AbsInt, 300
absolute time, limiting dependence on, 190-192
abstract interpretation, 299
access control lists (ACLs)
    advanced authorization controls, 79
    data isolation and, 168
    graceful degradation and, 223
    key isolation and, 166
    safe proxies and, 38
access controls
    designing for recovery, 211
    Google's corporate network and, 212
    understandability and, 105
access denials, 73
accidental errors, recovery from, 185
accountability, risk taking and, 479
ACLs (see access control lists)
active entities, defined, 100
activists, as attackers, 21
AddressSanitizer (ASan), 278
administrative APIs, 66, 70
advanced authorization controls, 79, 81-86
    business justifications, 84
    multi-party authorization, 81
    three-factor authorization (3FA), 82-84
advanced mitigation strategies, 314-327, 330
    binary provenance, 314-317
    code signing, 316
    deployment choke points, 325
    post-deployment verification, 327
    provenance-based deployment policies, 317-319
    verifiable builds, 319-324
adversarial testing, 8
adversaries, understanding, 15-34
    attacker methods, 30-32
    attacker motivations, 16
    attacker profiles, 18
    hobbyists, 18
    risk assessment considerations, 32-33
    vulnerability researchers, 18
AFL (American Fuzzy Lop), 284
Agile development, 57
alternative components
    pitfalls, 173
    types, 169-172
ALTS (Application Layer Transport Security), 104
American Fuzzy Lop (AFL), 284
amplification attacks, 219
Android, 263
Android Keystore, 118
Android security team, 464
anonymization, 353
Anonymous (hacktivist group), 22
antivirus software, 354
anycast, 222
APIs (application programming interfaces)
    defined, 65
    least-privilege-based design, 65-67
    secure cryptographic APIs and Tink crypto framework, 117-119
    third-party insider threats, 27
    usability and understandability, 116-119
App Engine (see Google App Engine)
App Security Improvement (ASI), 300
application frameworks

defined, 112
for service-wide requirements, 112-113
Application Layer Transport Security (ALTS), 104
application logs, 355
application-level data recovery, 432
artifact, defined, 304
artificial intelligence
cyber attacks and, 24
protecting systems from automated attacks, 24
ASan (AddressSanitizer), 278
ASI (App Security Improvement), 300
AST pattern matching, 292-295
ATT&CK framework, 32
attack surface
binary provenance and, 316
redundancy and, 5
attacker methods, 30-32
categorizing of tactics, techniques, and procedures, 32
Cyber Kill Chain framework for studying, 31
DoS attacks, 218
threat intelligence framework for studying, 30
attacker profiles, 18
activists/hacktivists, 21
criminal actors, 22-24
governments, 19-21
hobbyists, 18
insiders, 24-30
vulnerability researchers, 18
attackers (see adversaries, understanding)
auditing
automated systems, 377
choosing an auditor, 69
collecting good audit logs, 68
least privilege and, 68-70
authentication
credential/secret rotation, 433-435
least-privilege policy framework for, 78-81
second-factor authentication using FIDO security keys, 133-135
understandability and, 104
authentication protocol, defined, 100
authorization
advanced controls, 79
auditing to detect incorrect usage, 68-70
avoiding potential pitfalls, 80
business justifications, 84
investing in a widely used authorization framework, 80
least-privilege policy framework for, 78-81
multi-party, 81
temporary access, 85
three-factor authorization (3FA), 82-84
authorized_keys file, 202
automated attacks, 24
automated code inspection tools, 291-296
automated response mechanisms, 157-158
failing safe versus failing secure, 158
human involvement in, 159
automated testing, 240
automation
code deployment, 308
cyber attacks and, 24
response mechanism deployment, 157
awareness campaigns, 477
awareness, culture of, 476-479
AWS Key Management Service, 118

B

batteries-included frameworks, 112
BeyondCorp architecture, 430
Device Inventory Service tools, 356
location-based trust and, 165
zero trust networking model, 63
Bigtable, 345
binary provenance, 314-317, 432
BIOS, 207
blameless postmortems, 472, 480
blast radius, controlling, 159-166
failure domains, 167-172
location separation, 162-166
role separation, 162
time separation, 166
Blue Teams, 465-468, 465
breakglass mechanism
code deployment and, 329
graceful failure and, 74
least-privilege-based design, 67, 74
as safety net, 486
budget, for logging, 357
bug bounties (Vulnerability Reward Programs), 18, 446, 468-470
bugs, compromises versus, 390
builds

defined, 304
verifiable (see verifiable builds)

C

C++, 282
for publicly trusted CA, 237
sanitizing code, 267
CA/Browser Forum Baseline Requirements, 233
California Department of Forestry and Fire Protection, 403, 481
canaries, 484
Cantrill, Bryan, 337
CAPTCHA (Completely Automated Public Turing test) systems, 23, 224
CAs (see certificate authorities)
CASBs (cloud access security brokers), 356
casting, implicit, 266
Cellcom, 23
certificate authorities (CAs)
background on publicly trusted CAs, 233
build or buy decision, 235
complexity versus understandability, 238
data validation, 241
design, implementation, maintenance considerations, 236-241
at Google, 233-241
Google's business need for, 234
hardening with fixits, 240
programming language choice, 237
resiliency for CA key material, 240
securing third-party/open source components, 238
testing, 239
certificate revocation (see explicit revocation mechanism)
Certificate Signing Requests (CSRs), 237
certification (security specialists), 459
certification validity database, 201
CFG (control-flow graph), 300
champions, IR team, 368
change
building a case for, 491
picking your battles, 493
reducing fear with risk-reduction mechanisms, 484-486
resistance to, 483
slowing down a change, 139
change budget, 159
change, designing for, 121-141
architecture decisions to make changes easier, 123-127
best practices for designing your change, 122
complications: when plans change, 138
containers, 124
factors influencing speed of change, 127-138
Heartbleed security bug example, 140
keeping dependencies up to date, 123
long-term change: external demand, 136-138
medium-term change: improvement to security posture, 132-135
microservices, 125-127
rebuilding, 124
releasing frequently using automated testing, 124
second-factor authentication using FIDO security keys, 133-135
short-term change: zero-day vulnerability, 129-132
types of security changes, 122
chaos engineering, 281
charter, IR team, 369
checksums, 338
China, 23
choke points, 325
Chrome security team, 445-453, 464
background, 445
designing for defense in depth, 451
helping users safely navigate the web, 450
security as team responsibility, 448
speed of detecting and fixing security flaws, 450
stages of evolution, 446-448
transparency and community engagement, 452
CI/CD (see continuous integration/continuous deployment)
CIA (confidentiality, integrity, availability) triad, 6
Cisco, 33
CL (communications lead), 410, 413
Clang-Tidy, 292-294
CLI (command-line interface), 40-42
client software, 228
cloud access security brokers (CASBs), 356
cloud assets

compromised cloud instances, 438
identifying/inventorying, 356
Cloud Key Management Service (KMS), 118
ClusterFuzz, 289
code
deploying (see deploying code)
testing (see testing (code))
writing (see writing code)
code inspection tools, automated, 291-296
code reviews, 307
code signing, 316
Code Spaces, 11
Codenomicon, 140
collaborative debugging, 348
Colombia, 23
Columbia Disaster Investigation Board, 481
command-line interface (CLI), 40-42
common object model, 99
communication
crisis management and, 406-410
emergency access and, 212
foundation for trust, 7
hedging, 407
hypothetical crisis management example, 411
keeping the right people informed with the right levels of detail, 409
meetings in crisis management situations, 408
misunderstandings, 407
overcommunication and transparency when advocating for change, 488
preparing, 413
when email or instant messaging system is compromised, 372
when taking a break from debugging, 346
communications lead (CL), 410, 413
community engagement, 452
compartmentalization, 160-162
(see also blast radius, controlling)
location separation, 162-166
role separation, 162
complexity
breaking down, 95
evolution and, 9
in evolving systems, 9
least privilege and, 87
managing (see understandability)
understandability versus, 94
concolic testing, 299
confidentiality
isolation of, 165
reliability/security intersection, 6
configuration distribution
custom HTTP receiver (in-process), 77
custom HTTP receiver (sidecar), 77
custom OpenSSH ForceCommand, 76
in least-privilege environment, 74-78
POSIX API via OpenSSH, 75
software update API, 76
tradeoffs, 77
configuration-as-code, 310
conformance checks, 54
containers, 124
continuous integration/continuous deployment (CI/CD), 58
implementing verifiable builds, 322-324
provenance-based deployment policies, 317
unit tests, 272
continuous validation
designing for recovery, 209
exercising emergency components as part of normal workflows, 176
Google's CA, 241
injecting anticipated changes in behavior, 176
key rotation cycle measurement, 178
oversubscribing but preventing complacency, 178
resilient design and, 174-179
scenarios for, 176-179
splitting when you cannot mirror traffic, 177
validation focus areas, 175
control plane, 127
control-flow graph (CFG), 300
coordinated vulnerability disclosure (CVD), 391
costs
adding reliability/security to existing systems, 49
computing resources consumed by failure, 153
differentiating costs of failures, 152-154
logging, 357
recovery speed's effect on, 154
reliability/security failures, 8
resilience solutions, 179

third-party service providers, 51
credentials
defined, 100
revocation system, 200-204
rotation of, 433-435
criminal actors
as attackers, 22-24
protecting your systems from, 23
crises, incidents versus, 388-391
crisis management, 387-415
(see also disaster planning)
avoiding panic, 392
beginning of response, 392
closure, 414
communications, 406-410, 411, 413
compromises versus bugs, 390
coordinated vulnerability disclosure, 391
crises versus incidents, 388-391
email attack example, 399
establishing your incident team, 393
handovers, 402-405, 412, 413
hypothetical example, 410-414
intersection of security and reliability, 11
investigative process, 398-401
keeping control of the incident, 401-406
keeping the right people informed with the right levels of detail, 409
operational security, 394-397, 411
parallelizing the incident, 401
preparing communications and remediation, 413
reliability/security tradeoffs, 5
taking command of your incident, 391-401
trading good OpSec for the greater good, 397
triage, 388-390
when tools try to be helpful, 396
cross-site scripting (XSS), 254
cryptographic code, 117-119
cryptographic keys, 434
cryptography, 458
CSRs (Certificate Signing Requests), 237
culture, xxvii, 471-495
aligning goals and participant incentives, 484
balancing accountability and risk taking, 479
building a case for change, 491
building a culture of security and reliability, 471-495
building empathy, 489
changing through good practice, 483-490
culture of awareness, 476-479
culture of inevitability, 480
culture of review, 474-475
culture of sustainability, 481-483
culture of yes, 479
defining a healthy security/reliability culture, 473-483
escalations and problem resolution, 494
increasing productivity and usability, 486-488
leadership buy-in for security/reliability changes, 490-494
least privilege's impact on, 86
overcommunication and transparency, 488
picking your battles, 493
reducing fear with risk-reduction mechanisms, 484-486
safety nets as norm, 486
security and reliability as default condition, 473
culture of no, 479
culture of yes, 479
CVD (coordinated vulnerability disclosure), 391
Cyber Grand Challenge, 24
Cyber Kill Chain, 31
cyber warfare, 19

D

Dapper, 345
DARPA (Defense Advanced Research Projects Agency), 24
data corruption, 338
data integrity, 6
data isolation, 167
data plane, 127
data sanitization, 431
data summarization, 358
DDoS attacks (see distributed denial-of-service attacks)
debugging, 336-344
cleaning up code, 346
collaborative debugging, 348
correlation-versus-causation problem, 342
data corruption and checksums, 338

deleting legacy systems, 347
distinguishing common from uncommon bugs, 337
filtering out normal events from bugs, 340
hypothesis testing with actual data, 343
importance of regular practice, 344
improving access/authorization controls, 348
improving observability, 345
isolating the problem, 342
recording observations, 339
reliability of logging, 359
reproducing the bug, 341
rereading documentation, 344
robust, secure access to systems, 359
security investigations versus, 350
security issues, 359
setting aside time for debugging and investigations, 339
stopping when things start to go wrong, 347
taking a break, 346
techniques, 336-344
what to do when you're stuck, 344-348
declaring an incident, 393, 411
decompilers, 298
defacing of websites, 21
Defense Advanced Research Projects Agency (DARPA), 24
defense in depth, 145-150
Chrome security team (case study), 451
controlling blast radius, 159-166
Google App Engine analysis, 147-150
resilience and, 10
Trojan Horse attack, 145-147
degradation
controlling, 150-159
differentiating costs of failures, 152-154
DoS attacks and, 223
failing safe versus failing secure, 158
logs and, 358
resilience and, 150-159
response mechanism automation, 157-158
Delta Airlines, 8
denial-of-service (DoS) attacks, 217-229
amplification attacks, 219
attacker's strategy, 218
CAPTCHA implementation, 224
client retry behavior in self-inflicted attacks, 228
DDoS attacks versus, 220
defendable architecture, 220-222
defendable services, 222
defender's strategy, 219
designing for defense against, 220-222
graceful degradation, 223
mitigating, 223-226
mitigation system, 224
monitoring/alerting, 223
problems with failing open, 201
reliability/security intersection, 6
self-inflicted attacks, 226-228
strategic response, 225
user behavior in self-inflicted attacks, 226
deny lists, 194
dependencies, keeping up to date, 123
deploying code, 303-331
actionable error messages, 328
advanced mitigation strategies, 314-327, 330
automation for, 308
best practices, 307-311
binary provenance, 314-317
breakglass with, 329
code reviews, 307
code signing, 316
concepts and terminology, 304-305
creating unambiguous policies, 329
deployment choke points, 325
ensuring unambiguous provenance, 328
errors manifested during deployment, 343
maintaining confidentiality of secrets, 311
post-deployment verification, 327
practical advice, 328-330
provenance-based deployment policies, 317-319
securing against threat model, 311-314
supply chain issues, 328
threat model, 306
treating configuration as code, 310
trusting third-party code, 313
verifiable builds, 319-324
verifying artifacts, 309
deployment (generally)
definition, 305
response mechanism, 154-159
system, 10
Trojan Horse attack, 146
DER (Distinguished Encoding Rules), 237
design document template (Google), 47

design tradeoffs, 43-59
  balancing requirements, 49-53
  cost of adding reliability and security to existing systems, 49
  feature requirements, 44
  features versus emergent properties, 45-47
  Google design document template, 47
  initial versus sustained velocity, 56-59
  managing tensions/aligning goals, 54-56
  microservices and Google web application framework, 54
  nonfunctional requirements, 45
  objectives/requirements, 44-48
  payment processing example, 50-53
developers, least privilege and, 87
Device Inventory Service tools, 356
DevOps, 498
DevSecOps, xix, 498
dictionaries, fuzz engines and, 281
digital forensics, 398-401
disassemblers, 298
disaster planning, 363-385
  configuring systems, 374
  defining "disaster", 364
  dynamic response strategies, 364
  prestaging systems and people before an incident, 373-376
  processes and procedures, 376
  real-world examples from Google, 383-385
  risk analysis, 366
  setting up an incident response team, 367-373
  testing systems and response plans, 376-382
  training, 375
Disaster Recovery Testing (DiRT) program, 12, 384
disaster risk analysis, 366, 499
distinct failure domains, 167-172
  alternate component pitfalls, 173
  alternate component types, 169-172
  benefits of splitting system into, 168
  data isolation, 167
  functional isolation, 167
  high-availability components, 170
  high-capacity components, 170
  low-dependency components, 170-172
  resilience and, 10
distributed denial-of-service (DDoS) attacks, 218
  (see also denial-of-service (DoS) attacks)
  Anonymous's attack on Israeli websites, 22
  DoS attacks versus, 220
  reliability/security intersection, 6
DNS (Domain Name System) queries, 357
documentation
  culture of awareness and, 477
  maintaining access to, 373
  rereading, 344
dogfooding, 485
DoS extortion, 217
dumb fuzzing, 281
dynamic program analysis, 277-280
dynamic type checking, 264

E

elections, hacking of, 23
Elliptic Curve Cryptography (ECC), 151
email
  communicating when system is compromised, 372
  crisis management email attack example, 399
embargoed vulnerabilities, 128
emergency access
  access controls, 211
  communication channels, 212
  designing for recovery, 210-214
  responder habits, 213
emergent properties
  aligning security/reliability goals, 55
  feature requirements versus, 45-47
  reliability and security as, 46
empathy, 489
encryption
  defense in depth and, 10
  log data, 353
encryption keys (see key rotation)
Envoy HTTP proxy, 380
epoch, 191
error messages, 328
Error Prone, 259, 292, 296
errors, threat modeling and, 29
escalations, problem resolution and, 494
espionage, 19
EternalBlue, 147
evolution of systems, 9
exception handling, 328
explicit revocation mechanism, 200-204

avoiding risky exceptions, 203
centralized service to revoke certificates, 201
failing open, 201
handling emergencies directly, 202
removing dependency on accurate notions of time, 202
revoking credentials at scale, 203
exponential backoff, 228, 249
external researchers, 468-470

F

Facetime privacy bug, 397
failing closed (secure)
failing safe versus, 158
security/reliability tradeoffs, 158
failing open (safe)
failing secure versus, 158
revocation system, 201
security/reliability tradeoffs, 158
failing static, 225
failover strategies, 172, 381
failure domains (see distinct failure domains)
failures, cost of
computing resources consumed by, 153
differentiating costs of, 152-154
effect on user experience, 153
speed of mitigation, 154
failures, system-wide, 381
false positives/negatives, 291
fault injection, 379
fear
reducing with risk-reduction mechanisms, 484-486
resistance to change and, 483
feature requirements, 44
FIDO security keys, 133-135
firmware
capturing state for updates, 206
rollbacks, 199
first-party insiders, 25
fixits, 240
ForceCommand, 76
forensic timeline, 399
forensics, digital, 398-401
forward-only MASVN, 199
Fourth Industrial Revolution, 497
Frama-C, 300
frameworks, software development, 55
(see also application frameworks)
access control policies, 105
benefits of, 245
lessons for evaluation/construction, 256-259
reliability/security benefits of, 55
reliability/security enforcement, 244-249
rollout strategy, 258
RPC backends, 247-249
simple, safe, reliable libraries for common tasks, 257
understandability and, 112-113
full-stack frameworks, 112
functional isolation, 167
functional requirements, 44
fuzz engines, 281-285
fuzz testing (fuzzing), 271, 280-290
Chrome security team and, 449
ClusterFuzz, 289
continuous fuzzing, 289
example fuzzer, 286-289
fixits and, 240
how fuzz engines work, 281-285
"known safe" functions, 283
OSS-Fuzz, 290
security/reliability benefits, 280
writing effective fuzz drivers, 285

G

games, for developing culture of awareness, 476
General Electric (GE), 25
GFE (Google Front End), 126
global network failure, 171
Gmail, 94
Go, 237
Go Race Detector, 268
goals, aligning, 54-56
emergent-property requirements, 55
microservices and Google web application framework, 54
participant incentives and, 484
Google
DiRT exercise testing emergency access, 384
disaster planning at, 383-385
earthquake response test, 383
embedding security at, 463-465
industry-wide vulnerabilities in Linux kernel, 384
password manager incident, 3
reliability- and security-related sections of design doc template, 47

safe proxies case study, 37-42
security/reliability education, 476
smart system for intake, 464
sustainable reliability and security culture at, 482
Tool Proxy, 40-42
Google App Engine
culture of yes and, 479
defense in depth and, 147-150
runtime layers, 149
threat modeling, 148
Google design document template, 47
Google Front End (GFE), 126
Google Sanitizers, 268, 278
Google Search, 29
governments
as attackers, 19-21
cyber attacks as domestic activity monitoring, 20
intelligence gathering, 19
military purposes of attacks, 19
protecting systems from nation-state actors, 20
graceful degradation
DoS attacks and, 223
logs and, 358
resilience and, 150-159
graceful failure, 74
Gregg, Brendan, 343

H

Hacker Camp, 490
hacking (origin of term), 18
hacktivists
as attackers, 21
protecting systems from, 22
handovers, 402-405, 412, 413
hardware security module (HSM), 239
health, of team members, 482
Heartbleed security bug, 140, 283
hedging, 407
hermetic builds, 321
hero mode, 403
HIDS (host intrusion detection system), 354
high-availability components, 170
high-capacity service, 170
hobbyists, as attackers, 18
Honggfuzz, 285
host intrusion detection system (HIDS), 354
host management, 204-206
HSM (hardware security module), 239
HTTPS, 57, 136-138
human resource testing, 380

I

IC (incident commander), 367, 393
idempotent operations, 99
identifiers, 101
identities
Google production system model, 102
understandable, 100-104
IMAG (see Incident Management at Google)
imminent risk, 397
immutability, logging design for, 352
implicit casting, 266
implicit type conversions, 266
in-memory state, 205
incentives, aligning goals with, 484
Incident Command System, 12, 393
incident commander (IC), 367, 393
incident management (see crisis management; disaster planning)
Incident Management at Google (IMAG)
crisis management, 387, 392
crisis response, 12
IR team training, 375
incident response (IR) team, 367-373
avoiding single points of failure, 368
communicating when email or instant messaging system is compromised, 372
communications, 406-410
creating/staffing, 367-373
developing response plans, 371-373
establishing team charter, 369
handovers, 402-405, 412
identifying team members and roles, 367-369
keeping control of the incident, 401-406
maintaining access to documentation and update information, 373
morale issues, 369, 405
operating parameters, 370
playbooks for, 373
processes and procedures, 376
severity/priority models, 369
training, 375
incident, crisis versus, 388-391
incremental development and design, 261

indicators of compromise (IOCs), 30
inevitability, culture of, 480
Infer, 300
information warfare, 19
initial velocity, sustained velocity versus, 56-59
injection sinks, 115
insider risk
  defined, 24
  designing for, 29
  MPA protection against, 83
  resilience and, 10
  threat modeling, 28
insider threat, 24
insiders
  attacks by, 24-30
  designing for insider risk, 29
  determining intent of, 27
  first-party, 25
  related, 27
  third-party, 26
  threat modeling insider risk, 28
instant messaging, communicating when system is compromised, 372
intake, 464
integration testing, 276
intelligence gathering, 19
interactive sessions, 67
interactive talks, 476
interfaces
  common object model and, 99
  idempotent operations and, 99
  narrow, 98
  understandable, 98-100
Internet Relay Chat (IRC), 213
intersection of security and reliability, 3-13
  assessment, 8
  budgeting for long-term log storage, 358
  commonalities between reliability and security, 7-13
  communicating when email or instant messaging system is compromised, 372
  confidentiality, integrity, availability, 5-7
  crisis response, 11
  deployment of design, 10
  design considerations, 4
  effects of insiders, 25
  emergency access, 210
  evolution, 9
  Google password manager incident, 3
  invisibility, 7
  logging, 11
  permissions, 64
  recognizing a compromise, 350
  recovery, 12
  relationship between security and reliability, 4
  resilience, 9
  simplicity, 8
intrusion prevention systems (IPSs), 356
invariants, 91
  (see also system invariants)
  common security invariants, 244
  defined, 90
  system invariants, 91-93
investigating systems, 333-360
  debugging techniques, 336-344
  debugging versus, 350
  from debugging to investigation, 334-351
  intersection of security and reliability, 11
  logs, appropriate/useful, 351-358
  robust, secure debugging access, 359
  temporary files, 334-336
invisibility, of reliability and security, 7
IP addresses, microservices and, 101
IPsec, 57
IR team (see incident response team)
Iran, 20
isolation of assets, 429-430
isolation of trust, 164
Israel, 22

J

Java Cryptography Architecture (JCA), 276
JavaScript, 53
jitter, 228
just-in-time notifications, 478

K

key isolation, 165
Key Management Service (KMS), 118
Key Revocation List (KRL), 202, 203
key rotation
  cycle measurement, 178
  Heartbleed bug and, 141
  preventing rollback with, 198
  SSL keys, 434
  time separation and, 166
KMS (Key Management Service), 118

"known safe" functions, 283
known_hosts file, 202
Knusperli, 286-289
Kralevich, Nick, 263
KRL (Key Revocation List), 202, 203
ksplice, 384
Kubernetes, 325

L

lame-duck mode, 153
law enforcement agencies
  as attackers, 19-21
  cyber attacks as domestic activity monitoring, 20
law of diminishing returns, 403
leadership
  building a case for change, 491
  buy-in for security/reliability changes, 490-494
  escalations and problem resolution, 494
  picking your battles, 493
  understanding the decision-making process, 490
least privilege, 61-88
  advanced authorization controls, 79, 81-86
  auditing, 68-70
  best practices, 65-74
  breakglass, 67, 74
  business justifications, 84
  classifying access based on risk, 63
  complexity of security posture, 86
  concepts and terminology, 62
  configuration distribution, 74-78
  defined, 62
  designing for, 61-88
  developer complexity and, 87
  diagnosing access denials, 73
  graceful failure and breakglass mechanisms, 74
  impact on collaboration and company culture, 86
  investing in a widely used authorization framework, 80
  multi-party authorization, 81
  policy framework for authentication/authorization decisions, 78-81
  proxies, 85
  quality of security-impacting data, 87
  reliability/security tradeoffs, 85
  small functional APIs, 65-67
  temporary access and, 85
  testing and, 71-73
  three-factor authorization (3FA), 82-84
  tradeoffs/tensions, 86
  user productivity and, 87
  zero touch, 63
  zero trust networking, 62
legacy systems, 347
Liberia, 23
libFuzzer, 285
linters, 291
LLVM Clang, 282
load balancing, 221
load shedding, 156
location separation, 162-166
  aligning physical and logical architecture, 163
  isolation of confidentiality, 165
  isolation of trust, 164
location-based trust, 164
Lockheed Martin, 19
logging
  attackers' bypassing of, 67
  budget for, 357
  collecting appropriate/useful logs, 351-358
  designing for immutability, 352
  determining which security logs to retain, 354-357
  intersection of security and reliability, 11
  logs as attack target, 5
  privacy issues, 352
  reliability issues, 359
Lonestar, 23
lost causes, value of, 493
low-dependency service, 170-172

M

malicious actions, recovery from, 185
malware reports, 31
MASVN (minimum acceptable security version numbers), 195-198
mean time to detection (MTTD), 223
mean time to repair (MTTR), 223
meetings, in crisis management situations, 408
Mehta, Neel, 140
memory corruption, checksums and, 338
memory-safe languages, 263
mental models

idempotency and, 99
understandability and, 93
microservices
designing for change with, 125-127
Google's frontend design, 126
Google-internal framework, 54
rate-limiting mechanism as, 188
role separation, 162
military, cyber warfare and, 19
Miller, Matt, 263
minimum acceptable security version numbers (MASVN), 195-198
Mission Control program, 490
mission, of IR team, 369
mistakes, threat modeling and, 29
MIT (Massachusetts Institute of Technology), 18
mitigation doc, 422
mitigation strategies, advanced (see advanced mitigation strategies)
MITRE, 32
morale issues
IC's responsibility for, 405
on incident response teams, 369
motivations, of attacker, 16
MTTD (mean time to detection), 223
MTTR (mean time to repair), 223
multi-party authorization (MPA), 81
code review as, 307
reliability and, 80
resilience and, 10
unilateral insider risk protection, 83
multicomponent failure testing, 380
multilevel nesting, 260
mutation testing, 273

N
NASA, 481
nation-state actors, protecting systems from, 20 (see also governments)
Netflix, 281
network intrusion detection systems (NIDSs), 356
nonfunctional requirements, 45
nontechnical risks, 51
North Korea, 17
notes, keeping during recovery, 420, 436
NotPetya ransomware, 33
NSA, 33
NSO Group, 20

O
observability, improving, 345
OIDC (OpenID Connect), 102, 104
OL (operations lead), 394
one-time passwords (OTPs), 133-135
one-time programmable (OTP) devices, 199
OODA (observe, orient, decide, act) loop, 402
open source components
for Google custom CA, 238
third-party insider threats, 26
OpenID Connect (OIDC), 102, 104
OpenSSH
configuration distribution via, 75
custom OpenSSH ForceCommand, 76
OpenSSL library, 9, 140
operating parameters, IR team, 370
operating system logs, 354
Operation Aurora, 20, 436
operational overload, 339, 341
operational security (OpSec)
crisis management, 394-397
hypothetical crisis management example, 411
trading good OpSec for the greater good, 397
operations lead (OL), 394
OSS-Fuzz, 290
OTP (one-time programmable) devices, 199
OTPs (one-time passwords), 133-135
overcommunication, 137, 407, 488
overprovisioning, 219

P
panic rooms, 74
parallelizing an incident, 401
Park Jin Hyok, 17
partial restores, 209
passwords, 3
patch, defined, 129
payment processing system design (case study), 50-53
security/reliability considerations, 50
third-party service provider for sensitive data, 51-53
Peach Fuzzer, 282
penetration testers, 19, 467
permissions, 64

persistent data, 208
personally identifiable information (PII), 50
Petya ransomware, 33
phishing attack
  credential rotation and, 433
  recovery from, 439-440
  two-factor authentication to address risk of, 133
phone bridges, 212
physical location, 162-166
PII (personally identifiable information), 50
pivot points, 400
playbooks, IR team, 373
poisoned regions, 279
police (see law enforcement agencies)
policies
  avoiding automated unsupervised changes, 159
  creating unambiguous, 329
POSIX API, 66, 75
postmortems, 436-437, 472, 480
post_install command, 205
prestaging (disaster planning), 373-376
  configuring systems, 374
  processes and procedures, 376
  training, 375
pre_rm command, 205
priority models, IR teams and, 370
privacy, logging and, 352
production
  intersection of security and reliability, 10
  safe proxies in production environments, 37-40
  single system testing/fault injection, 379
  testing response in production environments, 379
productivity
  increasing, 486-488
  least privilege and, 87
profile, attacker, 18
program analysis
  dynamic, 277-280
  static, 290-301
    (see also static analysis)
Project Shield, 221
provenance
  binary, 314-317, 432
  data sanitization and, 432
  ensuring unambiguous provenance, 328
provenance-based deployment policies, 317-319
proxies, 85
  benefits of, 38
  downsides of, 39
  safe (see safe proxies (case study))
pseudonymization, 353
publicly trusted certificate authority (CA), 233-241
Purple Team, 382, 466
Pwn2Own, 450
Pwnium, 450

Q
quality-of-service (QoS) controls, 223
quarantine (isolation)
  assets, 429-430
  compartments and, 160

R
random errors, recovery from, 184
ransomware attacks, 23
  Petya, 33
  responses based on culture, 389
Rapid (software release tool at Google), 431
rate-limiting mechanism, 188
readability, 258
recovery, 417-435, 437-442
  aftermath of, 435-437
  attacker's response to, 423
  checklists, 428
  compromised cloud instances, 438
  compromised infrastructure/tooling, 424
  credential/secret rotation, 433-435
  data sanitization, 431
  examples, 437-442
  initiating, 429-435
  intersection of security and reliability, 12
  isolation of assets, 429-430
  large-scale phishing attack, 439-440
  logistics, 418-420
  mitigation option questions, 426-428
  open-ended questions/considerations, 423-428
  planning, 421-428
  postmortems, 436-437
  recovery data, 432
  reintroduction of attack vectors, 426
  repeating risks, 425

roles and responsibilities in, 419
scoping, 421
speed of mitigation, 154
system rebuilds and software upgrades, 430
targeted attack requiring complex recovery, 441
timeline, 420
variants of attack, 425
recovery checklists, 418, 420, 428, 441
recovery data, 432
recovery speed, 154
recovery, designing for, 183-215
accidental errors, 185
design for reasonable speed, 186-190
design for testing/continuous validation, 209
design principles, 186-210
emergency access, 210-214
explicit revocation mechanism, 200-204
isolating the rate-limiting mechanism, 188
knowing intended state, 204-209
limiting dependencies on external notions of time, 190-192
malicious actions, 185
random errors, 184
rollbacks as tradeoff between security and reliability, 192-200
scenarios requiring recovery, 184-186
software errors, 185
unexpected benefits, 214
Red Teams, 19, 381, 465-468
defined, 466
location-based trust and, 165
redundancy
controlling, 172-174
failover strategies, 172
payment processing system example, 52
reliability/security tradeoffs, 5
refactoring, 262, 274
reference documentation, 477
(see also documentation)
related insiders, 27
reliability
as emergent property of system design, 46
security and (see intersection of security and reliability; tradeoffs, reliability/security)
reliability risks, 4, 52
remediation lead (RL), 402, 413, 420
remote procedure calls (RPCs)
framework for backends, 247-249
safe proxies and, 38
three-factor authorization and, 83
tool proxies and, 40
reproducible builds, 322
researchers, 468-470
resilience, designing for, 143-181
continuous validation, 174-179
controlling blast radius, 159-166
controlling degradation, 150-159
controlling redundancies, 172-174
defense in depth, 145-150
design principles, 144
failure domains, 167-172
intersection of security and reliability, 9
overcoming resistance to, 180
response mechanism deployment, 154-159
solutions in order of costs, 179
starting points for, 179-181
resistance to change, 483
response mechanism deployment, 154-158
automated response, 157
load shedding, 156
reliability/security tradeoffs, 159
throttling, 156
response plans
auditing automated systems, 377
communicating when email/instant messaging system is compromised, 372
conducting nonintrusive tabletops, 378-379
developing, 371-373
DiRT exercise testing emergency access, 384
earthquake response test, 383
evaluating responses, 382
human resource testing, 380
industry-wide vulnerabilities in Linux kernel, 384
multicomponent testing, 380
in production environments, 379
real-world examples from Google, 383-385
Red Team exercises, 381
single system testing/fault injection, 379
system-wide failures/failovers, 381
testing, 376-382
reverse engineering, 31, 298
review, culture of, 474-475
revocation list, 201

revocation mechanism (see explicit revocation mechanism)
risk
  classifying access based on, 63
  reliability versus security design considerations, 4
risk analysis, 366, 499
risk assessment
  disaster risk assessment matrix, 499
  understanding adversaries, 32-33
risk ratings, 366
risk reduction, reducing fear with, 484-486
risk taking, balancing accountability with, 479
RL (remediation lead), 402, 413, 420
role separation, 162
roles and responsibilities, 455-470
  Blue and Red Teams, 465-468
  embedding security at Google, 463-465
  embedding security specialists and security teams, 462
  external researchers, 468-470
  integrating security into the organization, 460-470
  recovery and, 419
  security/reliability as everyone's responsibility, 456-460
  security/reliability risk evaluation, 457
  specialists' roles, 456
  understanding security expertise, 458
rollbacks
  deny lists, 194
  firmware/hardware-centric constraints, 199
  minimum acceptable security version numbers, 195-198
  rotating signing keys, 198
  security/reliability tradeoffs, 192-200
rollouts
  incremental, 258
  legacy conversions, 259
  staged, 484
  strategy for, 258
rotation of keys (see key rotation)
RPCs (see remote procedure calls)
RSA, 19, 151

S

safe proxies (case study), 37-42
  Google Tool Proxy, 40-42
  in production environments, 37-40
SafeHtml, 115, 254
SafeSql, 115
safety nets, 486
same-origin policy, 110
sandboxing, 147, 149, 451
sanitizing code, 267
sanitizing data, 431
scalable revocation, 203
scope, of IR team charter, 369
secrets
  dangers of including in code, 311
  rotation of, 433-435
    (see also key rotation)
security (generally)
  as emergent property of system design, 46
  integrating into the organization, 460-470
  reliability and (see intersection of security and reliability; tradeoffs, reliability/security)
security boundaries
  defined, 106
  small TCBs and strong security boundaries, 108
  threat models and, 109
  understandability and, 105-111
security by design, xxv
security champions, 463
security changes (see change, designing for)
security experts (see security specialists)
security investigations (see investigating systems)
security logs
  application logs, 355
  cloud logs, 355
  determining which security logs to retain, 354-357
  host agents, 354
  network-based logging and detection, 356
  operating system logs, 354
security posture, changes to, 132-135
security risks, 4, 53
security specialists
  certifications and academia, 459
  embedding into the organization, 462
  hiring, 458
  role of, 456
security teams, 462
  (see also Blue Teams; Chrome security team; Red Teams)

seed corpus, 282
self-inflicted DoS attacks, 226-228
  client retry behavior, 228
  user behavior, 226
self-registration, 487
self-resolution, 487
severity model, 369
Shellshock, 130
Shipshape, 297
sidecar daemon, 77
signing keys (see key rotation)
Simian Army, 281
simplicity, 8
single sign-on (SSO) services, 434
single system testing, fault injection and, 379
Site Isolation project, 451
Site Reliability Engineer/Engineering (SRE)
  crisis management, 392
  problem space's similarities to security problems, xix
  risk evaluation by, 457
  Security Engineering's similarities to, xxiii
  SRE Security Exchange Program, 489
software development (see testing (code); writing code)
software errors, recovery from, 185
software supply chain, 304-305
software upgrades, recovery and, 430
Sony Pictures, 17
Space Shuttle Columbia incident, 481
Spanner, 235, 334-336
SQL injection (SQLI), 245, 252-254
SRE (see Site Reliability Engineer/Engineering)
SRE Security Exchange Program, 489
SSL key rotation, 434
SSO (single sign-on) services, 434
staged rollouts, 484
stalkerware, 23
state
  defined, 204
  device firmware, 206
  global services, 207
  host management, 204-206
  knowing intended state, 204-209
  persistent data and, 208
state exhaustion attack, 221
static analysis, 290-301
  abstract interpretation, 299
  automated code inspection tools, 291-296
  formal methods, 301
  integration into developer workflow, 296-299
  reverse engineering and test input generation, 298
static type checking, 264
Stoll, Clifford, 15
stored XSS bugs, 115
Strava, 21
strongly typed language, 264, 265-267
structured justification, 70
supply chain, 214
  (see also software supply chain)
  code deployment issues, 328
  security benefits of designing for recovery, 214
sustainability, culture of, 481-483
sustained velocity, initial velocity versus, 56-59
Syrian Electronic Army, 21
system invariants, 91-93, 92
system logs, as attack target, 5
system rebuilds, 430
systems (generally)
  in context of systems engineering, xxv
  investigating (see investigating systems)

T

tabletop exercises, 378-379
tactics, techniques, and procedures (TTPs), 32, 422
talks, interactive, 476
TCB (see trusted computing base)
technical debt
  compromised assets as, 430
  during recovery, 427
  repaying when writing code, 261
technical program managers (TPMs), 450
temporary access, least privilege and, 85
test, defined, 304
testing (automated), 124
testing (code), 271-302
  correctness validation, 275
  dynamic program analysis, 277-280
  fuzz testing, 280-290
  integration testing, 276
  static program analysis, 290-301
  unit testing, 272-276
testing (general)
  designing for recovery, 209

Google's certificate authority, 239
least privilege and, 71-73
test environment security/reliability tradeoffs, 72
testing (response plans)
auditing automated systems, 377
conducting nonintrusive tabletops, 378-379
DiRT exercise testing emergency access, 384
earthquake response test, 383
evaluating responses, 382
human resource testing, 380
industry-wide vulnerabilities in Linux kernel, 384
multicomponent testing, 380
in production environments, 379
real-world examples from Google, 383-385
Red Team exercises, 381
single system testing/fault injection, 379
system-wide failures/failovers, 381
third-party code, 313
third-party components, 238
third-party insiders, 26
third-party service providers
benefits of, 51
costs and nontechnical risks, 51
reliability risks, 52
security risks, 53
sensitive data and, 51-53
threat intelligence, 30
threat mitigation (see advanced mitigation strategies)
threat modeling
deploying code, 306
insider risk, 28
mistakes and, 29
securing code against threat model, 311-314, 330
security boundaries and, 109
Trojan Horse attack, 146
three-factor authorization (3FA), 82-84
ticket queue, 464
time dependence, limiting, 190-192
Tink library, 117-119
Tool Proxy (Google binary), 40-42
TPMs (technical program managers), 450
tradeoffs, design (see design tradeoffs)
tradeoffs, reliability/security
controlling degradation, 152
deny lists, 194
imminent risk, 397
incident management, 5
redundancy, 5
response mechanisms, 159
rollbacks, 192-200
system usability in least-privilege environment, 85
test environments, 72
transparency
Chrome security team case study, 452
when advocating for change, 488
transport security, 104
triage, 388-390, 411
Tricorder, 297
Trojan Horse attack, 145-147
compromise, 147
deployment, 146
execution, 147
threat modeling and vulnerability discovery, 146
trust, isolation of, 164
trusted computing base (TCB), 105-111
small TCBs and strong security boundaries, 108
understandability and, 111
TrustedSqlString, 252-254
TTPs (tactics, techniques, and procedures), 32, 422
two-factor authentication (2FA)
3FA vs., 84
benefits during recovery, 435
using FIDO security keys, 133-135
type checking, 264
type conversions, implicit, 266

U

UCLA Medical Center, 26
Ukraine, 33
understandability, 89-119
access control and, 105
analyzing invariants, 92
API usability and, 116-119
application frameworks for service-wide requirements, 112-113
authentication and transport security, 104
benefits of, 90-94
breaking down complexity, 95
centralized responsibility for security/reliability requirements, 96

- complex data flows, 113-116
- complexity versus, 94
- designing understandable systems, 94-119
- Google CA implementation, 238
- identities, 100-104
- interface specifications, 98-100
- mental models and, 93
- security boundaries, 105-111
- software design, 111-119
- system architecture, 97-111
- system invariants and, 91-93

unit testing, 272-276
- effect on code, 274-275
- when to write unit tests, 273
- writing effective unit tests, 272

universal two-factor (U2F) hardware security tokens, 133
Unix, small functional APIs and, 65-67
unmanaged complexity, understandability versus, 94
unzipping, 194
Upvote, 488
URLs, 113
usability
- focus on, 487
- increasing, 486-488

user experience, failures' effect on, 153
user productivity, least privilege and, 87

V

Valgrind, 9, 268
validation, continuous (see continuous validation)
velocity, initial versus sustained, 56-59
verifiable builds, 319-324
- architectures, 320
- implementation, 322-324
- unauthenticated inputs, 324
- untrusted inputs, 324

Verizon, 8
version advancement, 191
virtual LANs (VLANs), 160
vulnerability researchers, as attackers, 18
Vulnerability Reward Programs (bug bounties), 18, 446, 468-470
vulnerability scanning teams, 467

W

wall-clock time, limiting dependencies on, 190-192
WannaCry Ransomware, 17, 147
web application firewall (WAF), 229
web applications, Google-internal framework, 54
web origin, defined, 110
WebKit, 446
"Wheel of Misfortune", 349
Wilkes, Maurice, 334
writing code, 243-269
- avoiding multilevel nesting, 260
- choice of tools for, 263-264
- common security vulnerabilities, 251-255
- eliminating YAGNI smells, 260
- frameworks to enforce security and reliability, 244-249
- lessons for framework evaluation/construction, 256-259
- memory-safe languages for, 263
- refactoring, 262
- repaying technical debt, 261
- rollout strategy, 258
- sanitizing code, 267
- security and reliability by default, 263-268
- simple, safe, reliable libraries for common tasks, 257
- simplicity's importance to, 259-263
- static type checking for, 264
- strong typing for, 264, 265-267
- understandability, 111-119

Wycheproof, 276

X

XSS (cross-site scripting), 254

Y

YAGNI ("You Aren't Gonna Need It"), 260
Yahoo!, 8
yes, culture of, 479
YouTube, 9

Z

Zero Touch interfaces, 63
Zero Touch Prod (ZTP), 38, 40
zero trust networking, 62
- access control and, 212

Google's corporate network, 212
location-based trust and, 165
zero-day vulnerability
responding to, 129-132
Shellshock example, 130
Zipkin, 345
ZTP (Zero Touch Prod), 38, 40