Table of Contents

Preface. xi

1. Resilience in Software and Systems. 1

What Is a Complex System? 2

Variety Defines Complex Systems 3

Complex Systems Are Adaptive 4

The Holistic Nature of Complex Systems 5

What Is Failure? 6

Acute and Chronic Stressors in Complex Systems 7

Surprises in Complex Systems 9

What Is Resilience? 11

Critical Functionality 12

Safety Boundaries (Thresholds) 13

Interactions Across Space-Time 15

Feedback Loops and Learning Culture 16

Flexibility and Openness to Change 18

Resilience Is a Verb 20

Resilience: Myth Versus Reality 21

Myth: Robustness = Resilience 21

Myth: We Can and Should Prevent Failure 22

Myth: The Security of Each Component Adds Up to Resilience 24

Myth: Creating a "Security Culture" Fixes Human Error 25

Chapter Takeaways 28

2. Systems-Oriented Security. 31

Mental Models of System Behavior 32

How Attackers Exploit Our Mental Models 34

Refining Our Mental Models 35
Resilience Stress Testing 35
The E&E Resilience Assessment Approach 39
Evaluation: Tier 1 Assessment 39
Mapping Flows to Critical Functionality 40
Document Assumptions About Safety Boundaries 41
Making Attacker Math Work for You 42
Starting the Feedback Flywheel with Decision Trees 59
Moving Toward Tier 2: Experimentation 59
Experimentation: Tier 2 Assessment 60
The Value of Experimental Evidence 60
Sustaining Resilience Assessments 63
Fail-Safe Versus Safe-to-Fail 64
Uncertainty Versus Ambiguity 67
Fail-Safe Neglects the Systems Perspective 68
The Fragmented World of Fail-Safe 69
SCE Versus Security Theater 71
What Is Security Theater? 72
How Does SCE Differ from Security Theater? 72
How to RAVE Your Way to Resilience 75
Repeatability: Handling Complexity 75
Accessibility: Making Security Easier for Engineers 76
Variability: Supporting Evolution 77
Chapter Takeaways 79

3. Architecting and Designing. 83
The Effort Investment Portfolio 84
Allocating Your Effort Investment Portfolio 85
Investing Effort Based on Local Context 86
The Four Failure Modes Resulting from System Design 89
The Two Key Axes of Resilient Design: Coupling and Complexity 91
Designing to Preserve Possibilities 91
Coupling in Complex Systems 93
The Tight Coupling Trade-Off 96
The Dangers of Tight Coupling: Taming the Forest 98
Investing in Loose Coupling in Software Systems 100
Chaos Experiments Expose Coupling 106
Complexity in Complex Systems 108
Understanding Complexity: Essential and Accidental 109
Complexity and Mental Models 111
Introducing Linearity into Our Systems 113

Designing for Interactivity: Identity and Access Management 121
Navigating Flawed Mental Models 123
Chapter Takeaways 126

4. **Building and Delivering.. 129**
Mental Models When Developing Software 130
Who Owns Application Security (and Resilience)? 131
Lessons We Can Learn from Database Administration Going DevOps 132
Decisions on Critical Functionality Before Building 134
Defining System Goals and Guidelines on "What to Throw Out the Airlock" 136
Code Reviews and Mental Models 136
"Boring" Technology Is Resilient Technology 138
Standardization of Raw Materials 140
Developing and Delivering to Expand Safety Boundaries 144
Anticipating Scale and SLOs 145
Automating Security Checks via CI/CD 146
Standardization of Patterns and Tools 151
Dependency Analysis and Prioritizing Vulnerabilities 154
Observe System Interactions Across Space-Time (or Make More Linear) 159
Configuration as Code 160
Fault Injection During Development 162
Integration Tests, Load Tests, and Test Theater 163
Beware Premature and Improper Abstractions 177
Fostering Feedback Loops and Learning During Build and Deliver 180
Test Automation 180
Documenting Why and When 183
Distributed Tracing and Logging 187
Refining How Humans Interact with Build and Delivery Practices 192
Flexibility and Willingness to Change 193
Iteration to Mimic Evolution 193
Modularity: Humanity's Ancient Tool for Resilience 195
Feature Flags and Dark Launches 198
Preserving Possibilities for Refactoring: Typing 199
The Strangler Fig Pattern 201
Chapter Takeaways 205

5. **Operating and Observing.. 209**
What Does Operating and Observing Involve? 210
Operational Goals in SCE 211
The Overlap of SRE and Security 212

Measuring Operational Success 214
Crafting Success Metrics like Attackers 215
The DORA Metrics 217
SLOs, SLAs, and Principled Performance Analytics 220
Embracing Confidence-Based Security 222
Observability for Resilience and Security 223
Thresholding to Uncover Safety Boundaries 227
Attack Observability 228
Scalable Is Safer 231
Navigating Scalability 233
Automating Away Toil 234
Chapter Takeaways 235

6. Responding and Recovering.. 239
Responding to Surprises in Complex Systems 240
Incident Response and the Effort Investment Portfolio 241
Action Bias in Incident Response 242
Practicing Response Activities 244
Recovering from Surprises 247
Blameless Culture 250
Blaming Human Error 253
Hindsight Bias and Outcome Bias 257
The Just-World Hypothesis 259
Neutral Practitioner Questions 261
Chapter Takeaways 263

7. Platform Resilience Engineering... 267
Production Pressures and How They Influence System Behavior 267
What Is Platform Engineering? 272
Defining a Vision 276
Defining a User Problem 277
Local Context Is Critical 278
User Personas, Stories, and Journeys 279
Understanding How Humans Make Trade-Offs Under Pressure 282
Designing a Solution 286
The Ice Cream Cone Hierarchy of Security Solutions 286
System Design and Redesign to Eliminate Hazards 289
Substitute Less Hazardous Methods or Materials 291
Incorporate Safety Devices and Guards 294
Provide Warning and Awareness Systems 295
Apply Administrative Controls Including Guidelines and Training 299

Two Paths: The Control Strategy or the Resilience Strategy 305
Experimentation and Feedback Loops for Solution Design 310
Implementing a Solution 311
Fostering Consensus 311
Planning for Migration 313
Success Metrics 313
Chapter Takeaways 317

8. Security Chaos Experiments. 321
Lessons Learned from Early Adopters 323
Lesson #1. Start in Nonproduction Environments; You Can Still Learn a Lot 323
Lesson #2. Use Past Incidents as a Source of Experiments 324
Lesson #3. Publish and Evangelize Experimental Findings 324
Setting Experiments Up for Success 324
Designing a Hypothesis 326
Designing an Experiment 328
Experiment Design Specifications 330
Conducting Experiments 335
Collecting Evidence 335
Analyzing and Documenting Evidence 337
Capturing Knowledge for Feedback Loops 338
Document Experiment Release Notes 339
Automating Experiments 340
Easing into Chaos: Game Days 341
Example Security Chaos Experiments 341
Security Chaos Experiments for Production Infrastructure 342
Security Chaos Experiments for Build Pipelines 344
Security Chaos Experiments in Cloud Native Environments 346
Security Chaos Experiments in Windows Environments 348
Chapter Takeaways 350

9. Security Chaos Engineering in the Wild. 353
Experience Report: The Existence of Order Through Chaos (UnitedHealth Group) 353
The Story of ChaoSlingr 355
Step-by-Step Example: PortSlingr 357
Experience Report: A Quest for Stronger Reliability (Verizon) 359
The Bigger They Are... 361
All Hands on Deck Means No Hands on the Helm 362
Assert Your Hypothesis 363
Reliability Experiments 363

Cost Experiments 366
Performance Experiments 367
Risk Experiments 368
More Traditionally Known Experiments 369
Changing the Paradigm to Continuous 370
Lessons Learned 371
Experience Report: Security Monitoring (OpenDoor) 371
Experience Report: Applied Security (Cardinal Health) 372
Building the SCE Culture 373
The Mission of Applied Security 373
The Method: Continuous Verification and Validation (CVV) 374
The CVV Process Includes Four Steps 374
Experience Report: Balancing Reliability and Security via SCE (Accenture Global) 376
Our Roadmap to SCE Enterprise Capability 377
Our Process for Adoption 378
Experience Report: Cyber Chaos Engineering (Capital One) 381
What Does All This Have to Do with SCE? 381
What Is Secure Today May Not Be Secure Tomorrow 382
How We Started 382
How We Did This in Ye Olden Days 383
Things I've Learned Along the Way 386
A Reduction of Guesswork 387
Driving Value 387
Conclusion 387
Chapter Takeaways 388

Index. 389