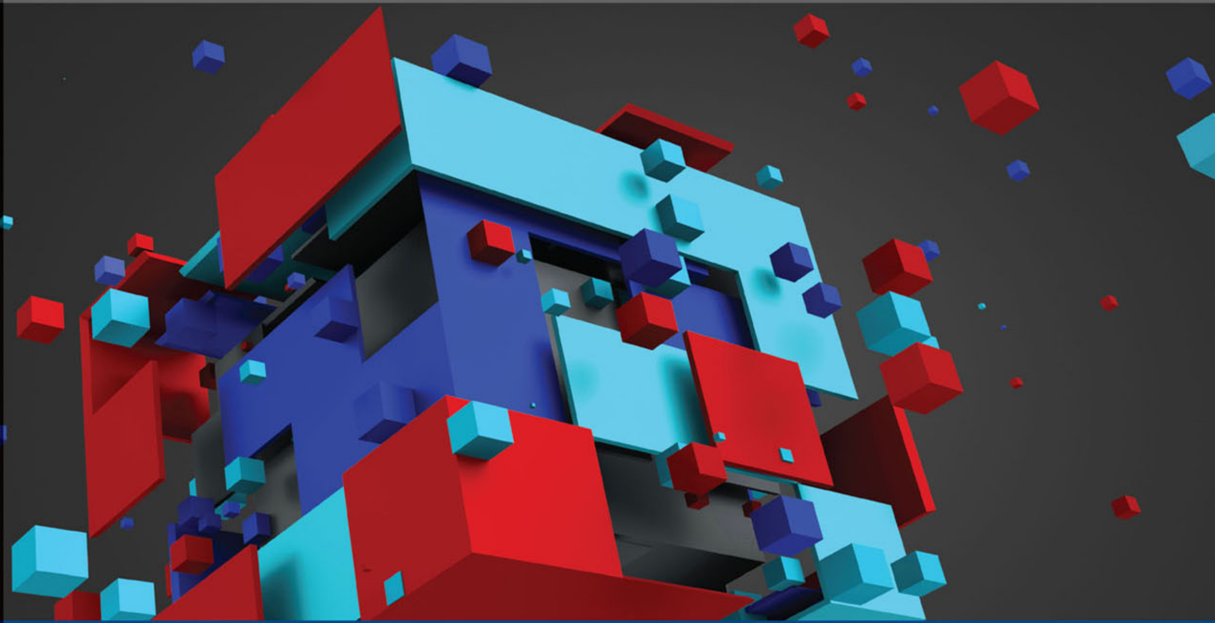
"This book is an essential guide for every architect and developer deploying secure, business-critical solutions on Azure."

—*from the foreword by* **Scott Guthrie,**
Executive Vice President, Cloud + AI Group, Microsoft

Microsoft

Designing and Developing Secure Azure Solutions

Best practices



Michael Howard • Simone Curzi • Heinrich Gantenbein

FREE SAMPLE CHAPTER

Microsoft

Designing and Developing Secure Azure Solutions

Michael Howard
Heinrich Gantenbein
Simone Curzi

Designing and Developing Secure Azure Solutions

Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

Copyright © 2023 by Pearson Education.

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/permissions.

No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-13-790875-2
ISBN-10: 0-13-790875-X
Library of Congress Control Number: 2022947556

ScoutAutomatedPrintCode

Trademarks

Microsoft and the trademarks listed at http://www.microsoft.com on the "Trademarks" webpage are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author, the publisher, and Microsoft Corporation shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the programs accompanying it.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

CREDITS

Editor-in-Chief:
Brett Bartow

Executive Editor:
Loretta Yates

Sponsoring Editor:
Charvi Aora

Development Editor:
Kate Shoup

Managing Editor:
Sandra Schroeder

Senior Project Editor:
Tracey Croom

Copy Editor:
Kim Wimpsett

Indexer:
Ken Johnson

Proofreader:
Barbara Mack

Technical Editors:
Jonathan Davis, Mike Becker, Rick Alba, Altaz Valani, Hasan Yasar

Editorial Assistant:
Cindy Teeters

Cover Designer:
Twist Creative, Seattle

Compositor:
codeMantra

Art:
Cover illustration:
Valex/Shutterstock

Figure 10-37:
Gordon Lyon

Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where:

- Everyone has an equitable and lifelong opportunity to succeed through learning.
- Our educational products and services are inclusive and represent the rich diversity of learners.
- Our educational content accurately reflects the histories and experiences of the learners we serve.
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview).

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

- Please contact us with concerns about any potential bias at https://www.pearson.com/report-bias.html.

For my family, who had patience with me as I wrote another book.

—MICHAEL

With love to my wife Denyse, thanking her for the support she provides.

—HEINRICH

With gratitude and love to my family, Silvia, Alice, and Viola, who endure and support me tirelessly.

—SIMONE

Contents at a glance

	Acknowledgments	*xxi*
	About the Authors	*xxiii*
	Foreword	*xxiv*
	Introduction	*xxv*
PART I	**SECURITY PRINCIPLES**	
CHAPTER 1	**Secure development lifecycle processes**	**3**
CHAPTER 2	**Secure design**	**23**
CHAPTER 3	**Security patterns**	**51**
CHAPTER 4	**Threat modeling**	**79**
CHAPTER 5	**Identity, authentication, and authorization**	**123**
CHAPTER 6	**Monitoring and auditing**	**171**
CHAPTER 7	**Governance**	**201**
CHAPTER 8	**Compliance and risk programs**	**213**
PART II	**SECURE IMPLEMENTATION**	
CHAPTER 9	**Secure coding**	**237**
CHAPTER 10	**Cryptography in Azure**	**285**
CHAPTER 11	**Confidential computing**	**359**
CHAPTER 12	**Container security**	**373**
CHAPTER 13	**Database security**	**391**
CHAPTER 14	**CI/CD security**	**435**
CHAPTER 15	**Network security**	**443**
	Appendix A: Core cryptographic techniques	*461*
	Index	*469*

Contents

Acknowledgments. . . . .*xxi*

Foreword . . . .*xxiii*

About the Authors. . . . .*xxiv*

Introduction . . . .*xxv*

PART I SECURITY PRINCIPLES

Chapter 1 Secure development lifecycle processes 3

Developers are the number-one source of compromises 3

Introducing the Microsoft Security Development Lifecycle 4

Quality ≠ security 4

Securing features vs. security features 5

SDL components 5

Security training 6

Defining your bug bar 7

Attack surface analysis 10

Threat modeling. 10

Defining your toolchain 11

Avoiding banned functionality 12

Using static analysis tools 13

Using dynamic analysis tools 16

Security code review. 17

Having an incident response plan 18

Performing penetration tests. 19

SDL tasks by sprint. 20

The human element 21

Summary 21

Chapter 2 Secure design 23

The cloud, DevOps, and security 23

IaaS vs. PaaS vs. SaaS, and the shared responsibility .24

Zero trust for developers. 27

Thinking about secure design 31

Security design principles applied to Azure 32

Attack surface reduction 32

Complete mediation 33

Defense in depth 34

Economy of mechanisms 37

Fail-safe defaults 38

Fail safe and fail secure 39

Least common mechanism 40

Least privilege 41

Leveraging existing components 42

Open design 43

Psychological acceptability 45

Separation of duties 45

Single point of failure 46

Weakest link 47

Summary 48

Chapter 3 Security patterns 51

What is a pattern? 51

Our take on Azure security patterns 51

Authentication pattern 53

Use a centralized identity provider for authentication 54

Authorization patterns 56

Adopt just-in-time administration 56

Assign roles to groups 58

Isolate from the internet 59

Isolate with an identity perimeter 60

Use role-based access control (RBAC) 61

Secrets management patterns 64

Use managed identities 64

Protect secrets with Azure Key Vault 66

Sensitive information management patterns 69

Create secure channels 69

Encrypt data client-side 71

Use bring your own key (BYOK). . . . 73

Availability pattern 74

Design for denial of service 75

Summary 77

Chapter 4 Threat modeling 79

TL;DR 79

What is threat modeling? 80

The four main phases of threat modeling 81

STRIDE's threat-classification approach 85

The trouble with threat modeling 86

Searching for a better threat modeling process 88

A better way to perform threat modeling: The five factors 90

Threat-modeling tools 91

Assessing the five factors. 92

CAIRIS 93

Microsoft Threat Modeling Tool 94

OWASP Threat Dragon. 96

pytm. 97

Threagile. 98

Threats Manager Studio 99

How to threat model: A real-life example 101

Analyzing the solution: The first meeting 102

Analyzing the solution: The second meeting 104

Identifying specific threats and mitigations 108

Automatically identifying additional threats and mitigations 113

Creating the roadmap 115

Using the dashboard. 118

Pushing selected mitigations into the backlog. 119

Summary 122

Chapter 5 Identity, authentication, and authorization 123

Identity, authentication, and authorization through a security lens 123

Authentication vs. authorization vs. identity 124

Modern identity and access management 125

Identity: OpenID Connect and OAuth2 fundamentals 125

- OpenID Connect and OAuth2 128
- Application registration 129
- Microsoft Authentication Library 130
- OAuth2 roles 134
- Flows 134
- Client types 137
- Tokens 138
- Scopes, permissions, and consent 138
- Anatomy of a JWT 142
- Using OAuth2 in your Azure applications 146

Authentication 150

- Something you know 151
- Something you have 151
- Something you are 152
- Multifactor authentication 153
- Who is authenticating whom? 154
- Creating your own authentication solution. 155
- The role of single sign-on 156
- Getting access without authenticating 157
- Authenticating applications 159

Authorization 161

- Azure AD roles and scopes 162
- Azure control plane built-in RBAC roles 163
- Azure data plane built-in RBAC roles. 164
- Managing role assignments 164
- Custom role definitions 165
- Denying assignments 167
- Role assignment best practices 167
- Azure AD Privileged Identity Management 168
- Azure attribute-based access control 168

Summary 170

Chapter 6 Monitoring and auditing 171

Monitoring, auditing, logging, oh my! 171

Leveraging the Azure platform 172

Diagnostic settings 173

Log categories and category groups 175

Log Analytics 176

Kusto queries 176

Raising alerts 181

Protecting audit logs 186

Using policy to add logs 189

Taming costs 189

The need for intentional security monitoring and auditing 190

The role of threat modeling 190

Custom events 193

Alerts from custom events on Azure Sentinel 197

Summary 199

Chapter 7 Governance 201

Governance and the developer 201

Azure Security Benchmark version 3 202

Network security 202

Identity management 203

Privileged access 203

Data protection 203

Asset management 204

Logging and threat detection 204

Incident response 204

Posture and vulnerability management 204

Endpoint security 205

Backup and recovery 205

DevOps security 205

Governance and strategy 205

Governance enforcement 206
Enforcement through processes 206
Governance documentation and security education 206
Role-based access control 206
Automated enforcement during deployment 206

Microsoft Defender for Cloud 207
Secure Score 207
Reviewing compliance state for solution 208

Azure Policy 209
Azure Initiatives and compliance frameworks 210
Azure Policy effects 210
Enforcement (effects) levels and RBAC by environment 210
Policy Assignments 211
Policy as code 212

Summary 212

Chapter 8 Compliance and risk programs 213

Something important to get out of way 213

What is compliance? 213
HIPAA 215
HITRUST 216
GDPR 216
PCI DSS 217
FedRAMP 218
NIST SP 800-53 219
NIST Cybersecurity Framework 220
FIPS 140 221
SOC 224
ISO/IEC 27001 225
ISO/IEC 27034 226
Center for Internet Security Benchmarks 226
Azure Security Benchmark 227
OWASP 229
MITRE 229
Compliance synopsis 232

Using threat models to drive compliance artifacts 233

Summary 234

PART II SECURE IMPLEMENTATION

Chapter 9 Secure coding 237

Insecure code 237

Rule #1: All input is evil 238

Verify explicitly 242

Determine correctness 243

Reject known bad data 253

Encode data 256

Common vulnerabilities 256

A01: Broken access control 257

A02: Cryptographic failures 257

A03: Injection 258

A04: Insecure design 259

A05: Security misconfiguration 259

A06: Vulnerable and outdated components .264

A07: Identification and authentication failures 264

A08: Software and data integrity failures 266

A09: Security logging and monitoring failures 269

A10: Server-side request forgery (SSRF) 269

Comments about using C++ 270

Don't write glorified C 270

Use compiler and linker defenses 270

Use analysis tools 271

Security code review 273

Keeping developers honest with fuzz testing 274

Generating totally random data 276

Mutating existing data 277

Intelligently manipulating data knowing its format 280

Fuzzing APIs 280

Summary 283

Chapter 10 Cryptography in Azure 285

A truth about security 286

Securing keys 287

- Access control and Azure Key Vault 288
- Use Key Vault Premium in production 299
- Enable logging and auditing 301
- Network isolation 304
- Use Microsoft Defender for Key Vault 306
- Back up your Key Vault assets 306
- Managed HSM and Azure Key Vault 308
- Secure keys with Key Vault summary 312

Cryptographic agility 312

- How to achieve crypto agility 314
- Implement crypto agility 315

The Microsoft Data Encryption SDK 324

- Optional parameters 326
- Managing SDK keys in Key Vault 327

Azure services and cryptography 329

- Server-side encryption with platform-managed keys 329
- Server-side encryption with customer-managed keys 330
- Client-side encryption 331
- Azure Storage cryptography 331
- Azure VM cryptography 335
- Azure SQL Database and Cosmos DB cryptography 337

Key rotation 337

- Azure Key Vault key rotation 339

Protecting data in transit 341

- TLS and crypto agility 343
- Ciphersuites 343
- TLS in Azure PaaS 345
- Setting ciphersuites 346
- TLS in Azure IaaS 350
- A common TLS mistake in .NET code 354
- Testing TLS 354

Debugging TLS errors. 354
Unsecure use of SSH 357

Summary 357

Chapter 11 Confidential computing 359

What is confidential computing? 359

Confidential computing processors. 361
Intel Software Guard Extensions 361
AMD Secure Encrypted Virtualization-Secure Nested Paging 362
Arm TrustZone 363
DCsv3-series VMs, SGX, Intel Total Memory Encryption, and Intel Total Memory Encryption Multi-Key 363
Attestation 364
Trusted launch for Azure VMs 366

Azure Services that use confidential computing 367

Summary 371

Chapter 12 Container security 373

What are containers? 373
You do not need containers for that!. 374
How to proceed from here 375

Container-related services on Azure. 375
Using containers on IaaS offerings. 376
Comparing Azure container services. 376

Problems with containers 381
Complexity. 381
Immaturity 383
Fragmentation 383

Securing container services 383
Development and deployment 384
The container registry 385
The cluster 386
The nodes. 387

The pods and containers . . . 387

The application . . . 388

Summary . . . 389

Chapter 13 Database security 391

Why database security? . . . 391

Which databases? . . . 392

Thinking about database security . . . 392

The SQL Server Family . . . 394

SQL Server . . . 394

Azure SQL Database . . . 394

Azure SQL Managed Instance . . . 395

Security in the SQL Server family . . . 395

Control plane authentication . . . 396

Control plane authorization . . . 398

Control plane auditing . . . 399

Control plane crypto on the wire . . . 400

Control plane network isolation . . . 401

Data plane authentication . . . 402

Data plane authorization. . . . 403

Data plane auditing. . . . 404

Data plane crypto on the wire . . . 405

Data plane network isolation . . . 405

Cryptographic controls for data at rest. . . . 406

Miscellaneous . . . 408

Cosmos DB security. . . . 411

Control plane authentication . . . 412

Control plane authorization . . . 413

Control plane auditing . . . 414

Control plane network isolation . . . 414

Data plane authentication . . . 414

Data plane authorization. . . . 415

Data plane auditing. . . . 419

Data plane crypto on the wire . . . 420

Data plane network isolation 420
Cryptographic controls for data at rest. 421
Miscellaneous 421

Encryption of data in use: Always Encrypted 422
Always Encrypted. 423
Always Encrypted with secure enclaves 429
Cosmos DB and Always Encrypted 431

SQL injection 433

Summary 434

Chapter 14 CI/CD security 435

What is CI/CD? 435

CI/CD tools. 435

Source control systems and supply chain attacks 436
Security tooling 436
Protecting your developers 436
Pull request approvals and PR hygiene. 437
Separation of duties, least privilege overview .437

Secrets and service connections 438

Protecting the main branch in Azure DevOps and GitHub. 438

Protecting the PROD deployment in Azure DevOps and GitHub. 439

Securing deployment agents. 440
Securing Azure DevOps agents. 440
Securing GitHub agents. 441

Summary 441

Chapter 15 Network security 443

Azure networking primer 443
IPv4, IPv6 in Azure. 445
IPv4 concepts 445
IPv4 addresses in Azure and CIDR 445
Routing and user-defined routes 446
Network security groups 446
Application security groups 447

Landing zones, hubs, and spokes 447

Hub and spoke and segmentation 447

Environment segregation, VNets, and allowed communications. 448

Ingress and egress controls. 449

NVAs and gateways. 449

Azure Firewall 449

Azure Firewall Premium SKU 450

Azure web application firewalls. 450

API Management Gateway 451

Azure Application Proxy 451

PaaS and private networking 451

Private shared PaaS 452

Dedicated PaaS instances 456

Managed VNets 456

Agent-based network participation. 456

Azure Kubernetes Service networking 457

Ingress controls 457

Egress controls with UDR. 457

Private endpoints for Kubernetes API server. .457

Cluster network policies. 457

The dangling DNS problem 459

An example 459

Fixing dangling DNS 460

Summary 460

Appendix A: Core cryptographic techniques 461

Index 469

Acknowledgments

This book covers a diverse and complex set of security-related topics. When writing a book like this, we as authors must make sure our facts are straight and our guidance is correct. We can do this only by asking questions of and obtaining assistance from people in the Azure product groups and people who are experts in their respective fields. To this end, we'd like to graciously acknowledge the help and assistance from the following people at Microsoft:

Amar Gowda, Amaury Chamayou, Anthony Nevico, Antoine Delignat-Lavaud, Barry Dorrans, Ben Co, Ben Hanson, Ben Oberhaus, Bhuvaneshwari Krishnamurthi, Dan Simon, David Nunez Tejerina, Dhruv Iyer, Eric Beauchesne, Eustace Asang-hanwa, Hannah Hayward, Jack Richins, Jakub Szymaszek, Jenny Hunter, Joachim Hammer, Jon Lange, John Lambert, Josh Brown-White, Ken St. Cyr, Kozeta Garrett, Luciano Raso, Mark Simos, Michael McReynolds, Michael Withrow, Mirek Sztajno, Nicholas Kondamudi, Niels Ferguson, Panagiotis Antonopoulos, Pieter Vanhove, Prasad Nelabhotla, Rafael Pazos Rodriguez, Robert Jarret, Rohit Nayak, Run Cai, Sameer Verkhedkar, Shubhra Sinha, Srđan Božović, Steven Gott, Sylvan Clebsch, Taylor Bianchi, Thomas Weiss, Vikas Bhatia, and Yuri Diogenes

Other Microsoft colleagues acted in a consulting capacity, spending hours helping with some of the more complex parts of the book. They are:

Kyle Marsh, Mark Morowczynski, and Bailey Bercik (for identity); Dave Thaler, Graham Berry, and Vikas Bhatia (for confidential computing); and Hervey Wilson (for Key Vault)

We also got feedback from people outside of Microsoft for their specific expertise:

Arun Prabhakar (Boston Consulting Group), Avi Douglen (Bounce Security), Brook S. E. Schoenfield (True Positives, LLC), Dave Kaplan (AMD), David Litchfield (Apple), Donna McCally (HITRUST), Izar Tarandach (Squarespace), Lotfi Ben Othmane (Iowa State University), Mark Bode (AMD), Mark Cox (RedHat), Mark Curphey (Crash Override), Matthew Coles (Dell Technologies), Michael F. Angelo (Micro Focus), Mike Dietz and Robert Seacord (Woven Planet), and Shane Gashette and Steve Christey Coley (MITRE)

We'd like to thank our technical reviewers, who scrutinized every aspect of our drafts. The technical reviewers were:

Altaz Valani (Security Compass), Hasan Yasar (Software Engineering Institute, Carnegie Mellon), Jonathan Davis (Microsoft), Mike Becker (Microsoft), and Rick Alba (Microsoft)

This book would not have been possible without the folks at Pearson/Microsoft Press: Loretta Yates for saying "yes"; Charvi Arora, who kept us marching forward to hit our dates; and finally, Kate Shoup, who did a magnificent job editing our text and yet maintaining our tone and technical intent.

Finally, we'd like to thank Scott Guthrie for writing the foreword and for leading the magnificent team that is Microsoft Azure.

Michael Howard
Austin, Texas

Heinrich Gantenbein
St. Paul, Minnesota

Simone Curzi
Perugia, Italy

About the Authors

Michael Howard

Michael Howard is a 30-year Microsoft veteran and is currently a Principal Security Program Manager in the Azure Data Platform team, working on security engineering. He is one of the original architects of the Microsoft Security Development Lifecycle and has helped diverse customers such as government, military, education, finance, and healthcare secure their Azure workloads. He was the application security lead for the Rio 2016 Olympic games, which were hosted on Azure.

Heinrich Gantenbein

Heinrich Gantenbein is a Senior Principal Consultant on Cybersecurity in Microsoft's Industry Solutions Delivery. With 30+ years of experience in software engineering and more than 30 years of experience in consulting, he brings a wealth of practical know-how to his role. Heinrich specializes in Azure security, threat modeling, and DevSecOps.

Simone Curzi

Simone Curzi is a Principal Consultant from Microsoft's Industry Solutions Delivery. He has 20+ years of experience covering various technical roles in Microsoft and has fully devoted himself to security for more than 10 years. A renowned threat modeling and Microsoft Security Development Lifecycle expert, Simone is a regular speaker at international conferences such as Microsoft Ready, Microsoft Spark, (ISC)^2 Security Congress, Carnegie Mellon's SEI DevOps Days, and Security Compass Equilibrium. Simone is also author of an open source threat modeling tool, Threats Manager Studio.

Foreword

In the last decade we have witnessed a dramatic shift in the way organizations have harnessed technology to completely reinvent and transform how they do business. Recent global challenges and unpredictable far-reaching events have only accelerated that change, and organizations have had to pivot and adapt to meet their customer and employee needs and ensure business resilience.

This digital transformation has been made possible in part by technology advancements and hyperscale cloud providers like Microsoft Azure that provide organizations with the agility to realize new efficiencies and capabilities. However, as we continue through this era of unprecedented transformation, including migration to the cloud, we are also experiencing new threats and requirements to ensure security and privacy.

When people think of security, they often think of endpoint protection, firewalls, and anti-malware tools, which are critically important, but architects and developers can't ignore application security during design and development. This book — *Designing and Developing Secure Azure Solutions* — is a necessary resource to understanding the essential elements of end-to-end secure software design and development on Azure. It addresses two areas I care about deeply – the security of Azure and software development.

The Microsoft Cloud has many reliability and security benefits compared to on-premises solutions, but architects and developers cannot ignore fundamental security practices when they deploy on Azure. Cloud-based solutions have a shared responsibility model, and some of the security onus is on the tenant as well as the cloud provider. *Designing and Developing Secure Azure Solutions* provides a holistic and approachable resource for anyone building secure workloads running on Azure. Readers working on Azure solutions will gain a contemporary understanding of secure development, design, and implementation.

The authors Michael, Heinrich, and Simone have decades of application security experience between them. They have worked with governments and companies — large and small — enabling each to design, develop, deploy, and manage secured solutions on Azure. I know the authors to be dedicated to helping anyone designing and developing on Azure achieve the reliability, scalability, and security demanded by their organizations and end users.

This book is an essential guide for every architect and developer deploying secure, business-critical solutions on Azure.

Scott Guthrie
Executive Vice President
Cloud + AI Group, Microsoft

Introduction

In mid-2021, during a recording of the *Azure Security Podcast*, Azure security expert and author Yuri Diogenes asked Michael if he planned to write an update to his book, *The Security Development Lifecycle*. Without hesitation, Michael responded, "No!"

But that wasn't the end of the matter.

The question Yuri asked planted a seed. Over the next few weeks, the three of us—Michael, Heinrich, and Simone—assembled a plan to write this book. Between us, we have worked with hundreds of customers to help them deploy business-critical solutions on Azure with confidence. This book is the culmination of that real-world experience.

The reason we wrote this book was not only to help you understand how to design and develop secure solutions running on Azure but to offer you pragmatic advice. The Venn diagram shown in Figure I-1 reflects how we see this book.

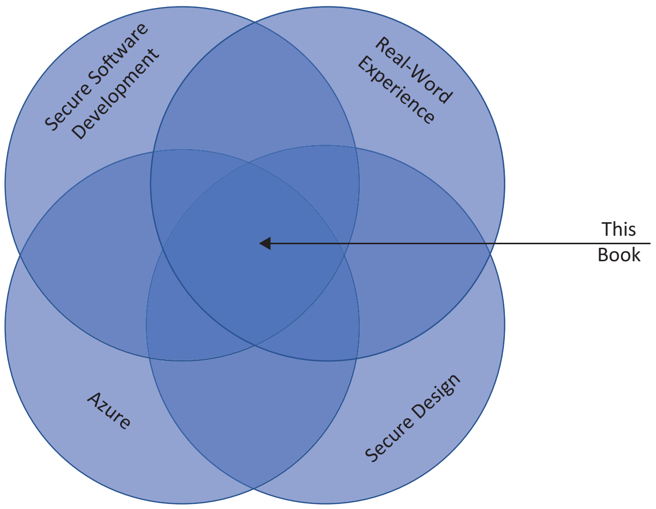


FIGURE I-1 The Intersection of this book's areas of coverage.

We do not cover some areas within Azure; otherwise, this book would be quite a tome. Most notably, we do not cover topics such as the following:

- **Privileged access workstations (PAWs)** A PAW is a workstation designed for administrative tasks only. It does not have access to email, general web browsing, and other productivity tasks. PAWs are used by elevated accounts to perform actions in high-risk environments, such as production, account administration, and more. You can learn more about PAWs here: *https://azsec.tech/irb*.

- **Conditional access and multifactor authentication (MFA)** These are often handled by an identity team, and the infrastructure should already be in place. With that said, conditional access and MFA are critical to securing an Azure-based solution. Learn more here: *https://azsec.tech/59d*.
- **Privacy** This is a book on security. Although security and privacy do overlap, security is mainly about fortifying a system and its data against unauthorized use, while privacy is about handling personal data. You can have security without privacy, but you cannot have privacy without security.

We've kept things relatively brief by including lots of links to outside information rather than covering some topics in depth in this book.

Organization of this book

This book is not designed to be read from cover to cover. You can do that, of course, but we have tried to make the chapters as independent as possible so they can be read individually. With that said, there are cross-references between chapters, and you might sometimes need to read a section of a different chapter to get the big picture.

The book also covers multiple ways to achieve a task, such as the following:

- Using the Azure Portal (although it's not common to use the Azure Portal in production systems because deploying in the real world usually uses a pipeline to push resources)
- Using the Azure command-line Interface (CLI)
- Using PowerShell code
- Using more complete code examples in different languages such as C#, Python, JavaScript, and more

Tip We have uploaded code samples and snippets to our GitHub repository at *https://github.com/AzureDevSecurityBook/*, so please make a point of visiting regularly.

Who should read this book

Just who is this book for? It's for anyone deploying solutions on Azure—whether they're architects, developers, or testers—who might not know a great deal about security but who want to make sure their design and code are as secure as possible. We cover a lot of ground in the book, but we also cover many complex topics in depth.

One final point: if you use the NIST Cybersecurity Framework (NIST CSF), then you're familiar with its core components: identify, protect, detect, respond, and recover. The material in this book focuses primarily on the protect component and some aspects of the detect component. Rolling out industry-grade solutions on Azure requires your organization to cover the other four components of the NIST CSF. You can read more about the NIST CSF in Chapter 8, "Compliance and risk programs," and on the NIST website, at *https://azsec.tech/81t*.

Thanks for reading!

Conventions and features in this book

This book presents information using conventions designed to make the information readable and easy to follow:

- Boxed elements with labels such as "Note" provide additional information
- Text that you type (apart from code blocks) appears in bold
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, "Press Alt+Tab" means that you hold down the Alt key while you press the Tab key
- A vertical bar between two or more menu items (e.g. File | Close), means that you should select the first menu or menu item, then the next, and so on

System requirements

Examples and scenarios in the book require access to a Microsoft Azure subscription and a computer that can connect to Azure. You can learn more about a trial subscription at this site:

azure.microsoft.com/en-us/free

GitHub Repo

The book's GitHub repository includes sample code and code snippets, and the authors will update this over time. The repo is *github.com/AzureDevSecurityBook/*.

The download content will also be available on the book's product page: *MicrosoftPressStore.com/SecureAzureSolutions/downloads*

Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

MicrosoftPressStore.com/SecureAzureSolutions/errata

If you discover an error that is not already listed, please submit it to us at the same page.

For additional book support and information, please visit *MicrosoftPressStore.com/Support*.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to *support.microsoft.com*.

Stay in touch

Let's keep the conversation going! We're on Twitter: *twitter.com/MicrosoftPress*

CHAPTER 3

Security patterns

After completing this chapter, you will be able to:

- Adopt the proposed patterns to improve how you securely design your solutions.
- Identify even more Azure security patterns, further improving your understanding of Azure.

What is a pattern?

Design patterns are not new to information technology, but they still play a fundamental role. Design patterns were conceived by a British-American architect of Austrian origins named Christopher Alexander. In 1977, Alexander wrote a book about recurring solutions to common problems related to building physical structures. However, this book became influential beyond its original field. Indeed, Alexander's work inspired four computer scientists and researchers—Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides—to apply the same concepts to software design. The result was a book titled *Design Patterns: Elements of Reusable Object-Oriented Software*, which is still widely used today.

In his book, Christopher Alexander defines patterns as follows:

> "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

The point here is that patterns represent a structured approach to address common problems. They are a way to collect and share know-how that has consistently provided value to many disciplines, including software design. Given that this book relates to the development of secure solutions on Azure, we focus here on design patterns in that context.

Our take on Azure security patterns

Azure uses design patterns extensively. You can find them almost everywhere such as when dealing with data protection at rest, implementing user authentication, and too many other scenarios to include here. Instead of reinventing the wheel, Azure adopts the same patterns and sometimes even the same services to provide critical capabilities and address common problems.

It is essential to know Azure's security patterns because they represent the best way to address common security problems. They may even enable you to design secure solutions without adopting more sophisticated approaches, like threat modeling, discussed in Chapter 4, "Threat modeling." This chapter introduces some of these patterns and includes the following information about each one:

- The name of the pattern
- The intent of and motivation behind the pattern
- A description of the pattern
- Examples of the pattern's implementation in Azure
- Related security principles (discussed in Chapter 2, "Secure design")
- Related patterns

Furthermore, the patterns are split into categories to simplify their identification:

- **Authentication** These patterns deal with the authentication of the counterparts of an interaction.
- **Authorization** These patterns focus on controlling access to resources.
- **Secrets management** These patterns deal with how the solution manages the secrets.
- **Sensitive information management** These patterns focus on how to manage sensitive information.
- **Availability** These patterns deal with ensuring that resources are accessible by legitimate users.

In the following pages, we describe a few patterns, sorting them by alphabetical order within each category. This is not an exhaustive list by any means. It simply includes some of the most common patterns we have seen in our practice that focus solely on solution design. This chapter does not cover implementation and deployment patterns, like those related to the supply chain. Considerations that relate to those patterns are covered elsewhere in the book, in Chapter 9, "Secure coding."
By discussing some of the most important patterns here and clearly stating why you need to adopt them, we aim to provide you with a consistent view.

Note For a more complete list of patterns, see https://azsec.tech/kph.

Once you know about patterns and their importance in Azure, you might be able to identify other patterns in the services you use. This understanding of design decisions empowers you to design better solutions by adopting the same concepts and by using Azure Services correctly.

The Azure Well-Architected Framework

Microsoft has published a set of guidelines on implementing sound architectures on Azure. These guidelines are available as part of the Azure Well-Architected Framework, which is available here: https://azsec.tech/dm8. These guidelines cover multiple aspects of implementing architectures, including reliability, security, cost optimization, operational excellence, performance efficiency, workloads, and services. For our purposes, the section on security is the most interesting. It covers topics like governance, landing zones, identity and access management, and much more. Many concepts within the Azure Well-Architected Framework can be mapped to the design patterns discussed in this chapter, but it goes well beyond our scope. Therefore, our recommendation is to take a look at it.

Authentication pattern

Authentication is an essential property of any solution. It pertains to identifying your counterpart in a conversation with some certainty.

We typically refer to the degree of certainty as the *authentication strength*. For example, suppose someone declares who they are without providing any proof. This would not be an authentication, but rather an identification. If they do a little better and provide a password, this would be weak authentication. Of course, you can impose restrictions to ensure an attacker cannot easily guess user passwords, but this has a limited effect on security.

Authentication typically involves the use of one, two, or all of the following parameters:

- **Something you know** This might be a password.
- **Something you have** This could be your phone or a physical token.
- **Something you are** This might be biometric information.

If you base authentication only on the password—that is, something you know—you have single-factor authorization, which can easily be compromised. To increase security, you should pair it with a second factor—for example, your username and password (things you know) and your phone (something you have). The idea here is that although an attacker might easily compromise any one of them, compromising them both at the same time would be much more challenging. Finally, you could add the third factor—something you are—to improve security even more. This approach is typically called *multifactor authentication* (MFA) because it relies on more than one factor.

Note Chapter 5, "Identity, authentication, and authorization," contains more information about this topic.

Authentication does not represent a value per se, but it is instrumental in securing your application and data.

This section focuses on one common pattern for securing your solution through authentication: using a centralized identify provider for authentication.

Use a centralized identity provider for authentication

Intent and motivation

The cloud is a convenient platform for hosting applications—so much so that you might host many applications on it, all of which require authentication. It is only natural for organizations to seek a centralized approach for managing identity to access all different services with a single set of credentials. This requirement is not only for simplicity but also to retain control and visibility.

Centralizing identity allows for the adoption of tools like user and entity behavior analytics (UEBA). These tools enable you to determine whether any account or system represents a potential risk for the organization by analyzing its behavior, often adopting artificial intelligence (AI) algorithms able to identify changes in usage patterns.

Another advantage of centralized identity systems is they provide a single location for managing identities and grants. They also allow you to integrate identity management with HR processes—for example, to remove or disable a user's account as soon as that user ceases their relationship with the organization. Finally, these identity systems enable you to review assigned rights and remove them when necessary.

Description

Azure AD represents a complete and unified approach to identity management on Azure. It provides fundamental capabilities, like managed identities and access reviews, and can be extended with additional services to increase security. These services include:

- **Azure AD Identity Protection** This service determines which identities are at risk by analyzing signals from many sources, such as threat intelligence, leaked credentials, and Microsoft Defender for Cloud Apps (Microsoft's Cloud Access Service Broker, or CASB).
- **Microsoft Defender for Cloud Apps** You can use this service to control and limit the adoption of applications and to detect the presence of shadow IT in your organization.
- **Microsoft Defender for Endpoints** This is a user and entity behavior analytics (UEBA) solution that can be integrated with Windows 10, Windows 11, and various devices.
- **Azure AD Privileged Identity Management (PIM)** This service enables you to assign access rights when required, eventually requiring approval from a third party before executing the assignment.

Azure AD also provides zero-trust security for the implementation of identities through the use of conditional access. This defines policies to prevent access to services by users who are not trusted enough. For example, suppose Azure AD Identity Protection has identified a potential security risk related to a particular user. It can then force that user to authenticate with MFA when accessing sensitive resources.

Note Azure AD is not your only option for identity management. You could use a third-party tool instead. Indeed, there are various reasons to use a third-party identity provider. For example, your organization might have already adopted one for your on-premises environment or to support different clouds, like AWS or GCP.

Examples

- Adopt conditional access to require MFA for privileged users such as the solution administrators.
- Use Azure AD to define custom application roles to control how the solution is used. You can find out more about these here: https://azsec.tech/0g6.
- Although some services—like Azure SQL Database, SQL Managed Instances, Cosmos DB, and Azure Storage—provide different ways to authenticate, including using Azure AD credentials, you should use Azure AD credentials whenever possible, because they can be better controlled from a central location. Using Azure AD credentials also enables the adoption of the whole set of capabilities offered to secure these identities.

Related security principles

- Zero trust
- Complete mediation
- Defense in depth
- Economy of mechanisms
- Least privilege
- Leveraging existing components

Related patterns

- Adopt just-in-time administration
- Use role-based access control (RBAC)
- Use managed identities

Authorization patterns

Authorization is another fundamental property of any secure solution. It comes after authentication. Authorization focuses on allowing users to perform actions they are entitled to perform and on preventing them from performing actions they are not entitled to perform. This section focuses on some common patterns for securing your solution through authorization.

We mentioned that authentication does not represent a value per se, but it is instrumental in securing your application and data. The same applies to authorization.

Adopt just-in-time administration

Intent and motivation

Accounts are not all equal. Some have considerable privileges assigned to them, making them a juicy target for malicious actors. For example, highly privileged administrators, like those assigned the roles of Global Administrator or User Administrator, are powerful, and it would cause significant damage to the organization if their user accounts were abused. For this reason, the most common recommendation is to apply the principle of least privilege by assigning the various roles only to those users who require them for a legitimate business reason and only when there is no other alternative that would allow for the assignment of fewer rights. For example, suppose one user needs to read security reports because she works on the security team. In that case, you should assign her the Security Reader role instead of Secure Administrator or Global Administrator.

This approach is a critical best practice, so there is a good chance your organization already applies it. Still, too many organizations struggle to adopt this approach due to the many roles defined in Azure AD and Azure Services. That is to say, sometimes the "best" role for a user—the one that most closely meets that user's needs—is not the most secure one. This pattern ensures that no unnecessary rights are assigned that could be exploited by an attacker or a malicious insider. Still, the rights that are assigned could be enough to cause significant losses.

A study by IBM Security and ObserveIT, published in 2020, found that the average cost of a security breach caused by an insider was $11.45 million. The study also found that, on average, the companies interviewed during the course of the study experienced 3.2 such incidents per year. Finally, the study determined that it's possible to reduce these losses by about $ 3 million by adopting a privileged access management (PAM) tool.

Note You can download the study mentioned here from https://azsec.tech/56t. For a discussion of the study, see https://azsec.tech/au8.

The idea behind PAM tools is that users do not need privileges 24×7. Rather, they need them for only a limited time. So, a privileged account represents a potential risk only when someone uses it to perform necessary actions. Outside that period, a PAM tool can revoke the rights, such that the user in question has no assigned privileges. PAM tools also deter malicious use because they require users to specify a business reason to obtain a particular privilege. They can also be used to require the approval for a third party to assign a particular right, in accordance with the separation of duties principle.

Description

Azure AD PIM is the primary tool within Azure for just-in-time administration. This tool extends how you assign roles to users or groups. With plain Azure AD, you assign roles globally or within a specific context, depending on the scope of the role. The role is then assigned forever. With Azure AD PIM, you can mark a role as eligible and then assign policies to that role. An eligible role remains dormant until it is activated. Activation can be subject to approval and could require additional authentication with MFA as well as the specification of a business reason. Activation can also be temporary and automatically removed after a set time.

Tip Azure AD PIM is not something you implement for the purposes of application development. It is part of a broader initiative that is typically the responsibility of those who manage your Azure AD tenant. If you decide you need Azure AD PIM to limit the exposure of privileged accounts for your application, you might want to ask the owners of the tenant to adopt it. If your organization has already adopted it, you can ask to define the rules you require for your application.

Example

Sometimes developers need access to production data to troubleshoot problems. While this is understandable, you should try to avoid it. That being said, in an emergency, you might not be able to avoid this, as developers will need to access this data to quickly identify a solution. The best way to handle these types of situations is to plan for them. In this effort, Azure AD PIM may play a significant role by enabling you to define an authorization process that requires a valid reason and approval by a third party.

Related security principles

- Attack surface reduction
- Defense in depth
- Least privilege
- Leveraging existing components
- Separation of duties

Related patterns

- Use a centralized identity provider for authentication
- Use role-based access control (RBAC)

Assign roles to groups

Intent and motivation

Let's face it: determining what access rights are required is not always a trivial matter. In many cases, identifying the right blend of rights is a matter of trial and error. It's very tempting to assign full rights and call it a day!

This behavior is a consequence of many factors, not least of which is the number of built-in roles provided with Azure. There are currently more than 80 built-in roles for Azure AD and more than 240 built-in roles for Azure services! No wonder it is so difficult to find the right roles to assign.

Still, it is imperative to identify a feasible approach to guarantee that users are assigned the most limited rights possible. Focusing on feasibility might be the difference between having an academic requirement and effectively making a difference.

Description

The critical point here is to apply least privilege without sacrificing manageability. One way to achieve this is by using groups. The idea is to define whatever usage scenarios you need and then create groups to support them. Once that's done, you can assign the required roles to each group and then assign users to groups as needed.

This approach is helpful for two reasons:

- It minimizes management because it aligns groups according to how the organization works.
- It enables you to minimize the number of role assignments. This is essential because Azure restricts you to 2,000 role assignments per subscription. For more on this, see https://azsec.tech/ad8.

Common exceptions to this rule are service principals and managed identities, which are typically assigned the required roles directly.

Tip Designing a suitable authorization model is a matter of business requirements, which must be agreed upon in advance among business, architecture, and operations stakeholders, per the organization's current policies and with requirements from security and operation. Many aspects are common to all projects, but the details will depend on your solution.

Example

If you have a data lake, you might want to guarantee different levels of visibility, depending on your area. You can address this requirement by assigning the required permissions. The easiest way to do this is to define a group for each category of users—like HR, research, and marketing—and then assign the required rights to various branches in the data lake, based on the desired visibility. You can even reuse the same groups for many applications simply by assigning additional rights to them when required.

Related security principles

- Least privilege

Related patterns

- Use a centralized identity provider for authentication
- Use role-based access control (RBAC)

Isolate from the internet

Intent and motivation

Chapter 2 showed that zero trust has been designed to address the many shortcomings of walled-in defenses. For example, suppose that for your security, you rely only on controls blocking external users from accessing the internals of your solution. In that case, you could be compromised as soon as someone figures out how to circumvent your layer of protection.

So, the answer would be to *not* rely on firewalls and similar protections, right? Wrong! The defense-in-depth principle states that every single control you can put in place can be circumvented. This principle doesn't imply, however, that you should give up your hopes of protecting your solution or that you should ditch controls like firewalls because they can't fully protect you on their own. It simply means that you need to integrate them with other controls to make your solution more secure. The bottom line is that network isolation still plays a significant role in Azure.

Description

So, how can you protect your solution using network defenses? The first step is to identify the parts of your solution that you must expose to the internet, as well as the parts of your solution that must remain internal. Then, you need to focus on these parts that should not be exposed to ensure that no unauthorized entities can reach them from the internet.

There are a couple of ways to achieve that:

- **Define firewall rules** Most services can define firewall rules. Once these rules are defined, you can use them to block access from the internet. This approach is simple and requires minimal configuration. But it has a downside: your resource or service is exposed. In other words, the firewall rules blocking unwanted traffic are your only protection against external malicious actors.

- **Define private endpoints** As with firewall rules, most services can define private endpoints. With private endpoints, there is no exposure over the internet, because the endpoint gets a private IP address. You can then connect to the private endpoint using private links. Configuring private endpoints and private links requires more work than configuring firewall rules, but it is more secure.

Note Chapter 15, "Network security," includes additional information and details on implementing this pattern and on networking in general.

Examples

- If you have a web application or a web API that is internal to your organization and you do not want to expose it on the internet, you can host it with App Service Environment (ASE). An ASE is a high-performance isolated environment to host your web applications and APIs. ASE is the only way to deploy web applications and APIs based on App Service to a VNet.
- You can configure Azure SQL with firewall rules to prevent direct access from the internet as a whole but still allow access from specific IP addresses. If you instead want to prevent all access from the internet, you might want to use a private endpoint.

Related security principles

- Attack surface reduction
- Defense in depth

Related patterns

- Create secure channels
- Isolate with an identity perimeter

Isolate with an identity perimeter

Intent and motivation

You learned from the preceding pattern, isolate from the internet, that network isolation is a key mitigation that you should not disregard. The same goes for identity. Both should be applied simultaneously, per the defense-in-depth principle.

Note This pattern is specular and complementary to the previous one.

Description

Identity can represent a perimeter that a user must cross to access protected resources. In this way, it is similar to networking. With networking, you can use rules to allow specific IP addresses to access your solution but not others. Similarly, you can use identity authentication and authorization to control who can access your resources and what they can do with them.

These approaches are both different and complementary, providing different capabilities due to their respective limitations. For example, with IP filtering, you can allow requests from specific geographies or buildings and deny everything else. With authentication and authorization, you can define rules for a specific user or account and establish its rights with regard to your solution or its components.

Example

Azure Front Door, App Gateway, and API Management Gateway are three network virtual appliances (NVAs) that you can configure to provide network perimeter defenses via private networking. The private networking approach is often considered enough for many solutions, but is that really so? As usual, the defense-in-depth principle says no. Rather, the recommendation is to add an identity perimeter defense as an additional layered defense. Adding an identity perimeter is typically achieved with TLS mutual authentication. With this approach, the NVA presents its client certificate to the back end. In turn, the back end verifies the certificate and its validity to ensure that the connection comes from the expected NVA and not from a malicious or accidental source.

Related security principles

- Zero trust
- Defense in depth

Related patterns

- Use a centralized identity provider for authentication
- Isolate from the internet

Use role-based access control (RBAC)

Intent and motivation

Managing authorization can be complicated. There are many resources and applications, and each of them defines multiple actions that must be authorized. With thousands of actions that can be authorized, you need some way to group and manage them. Moreover, you need a mechanism for use with custom solutions. The idea is that by taking a widely used approach, you can have a more reliable and secure method for authorization than what you could have with a custom authorization process.

Description

Azure RBAC is a common approach that you can use for Azure AD and Azure Services. It provides a structured way to assign users prebuilt sets of rights designed to address everyday needs. Azure AD currently provides more than 120 built-in roles you can choose from. For example, the Reader role allows the user to read the configuration of a service but typically not its data. In contrast, the Contributor role allows the user to change the configuration of a service or create new resources, depending on where the role is assigned. You can also create custom roles, but these are rarely necessary and often create management issues.

A fundamental advantage of using Azure RBAC is that it allows for a centralized view of all the rights assigned to users. This is essential in the event of a compromise because it enables you to browse for affected user accounts to grasp the security implications of their compromise. If you instead use credentials local to the resources and assign rights without using RBAC, you must inspect each resource to obtain the same information. Using RBAC also enables the execution of access reviews to periodically verify that the granted access is still required. And adopting Azure AD credentials and RBAC allows you to leverage a growing set of tools to analyze identities, like Azure AD Identity Protection and Sentinel.

Tip Azure RBAC is not the only approach to authorization. Sometimes you might need something different, like a custom database representing your authorization matrix. Before searching for an alternative solution, however, you should determine whether you can leverage Azure RBAC to achieve the desired result.

Roles for applications

You might want to apply something similar to RBAC for your applications. For example, if you are creating a web portal and you want some users to provide content, it would be great if you could define a role called Editor and then have that role sent back to the application as part of the authentication token.

Of course, you could use an Azure group to do that, but this might not provide you with the right granularity. For example, maybe you want to have multiple distinct groups of users with the Editor role, one for each structure in your company. This is where defining a role would come in handy because it allows you to create as many groups as you want and then assign the role to all of them. Your application could then check if the role is present, not whether the user belongs to one of the groups.

Azure provides this capability through the app roles. App roles can be assigned to users and groups and are sent to the application when a user accesses it. App roles can also be assigned to client applications to access your application. App roles are a great way to structure access control for applications, using a secure approach that is integrated with the platform. You can find additional information about app roles here: https://azsec.tech/0g6.

Example

Suppose you need to create a multilevel authorization hierarchy. Your organization has multiple departments, including one called the Commercial department. The top level of this department is Global Commercial, which controls various regions. For example, as shown in Figure 3-1, there's a region called ATZ, another called Europe, and others. Several countries comprise each region. So, for instance, ATZ includes the USA, Canada, Mexico, and other countries. Each user in Global Commercial has complete visibility; each user associated with a particular region can see everything within that region; and users associated with a particular country can see only the information pertaining to that country.

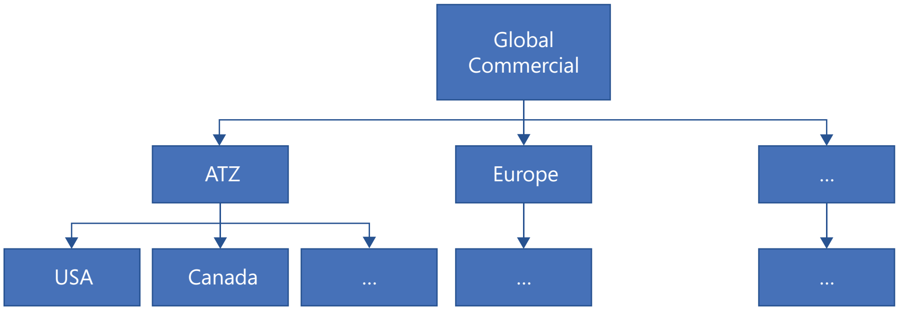


FIGURE 3-1 The hierarchy of the Commercial department.

How might you handle this scenario with Azure RBAC? At a minimum, you would need to rely on a database to represent the hierarchy and on custom code to enforce it. But you can do better. For example, you could do the following (see Figure 3-2):

1. Create a group for each country, a group for each region, and a group for the "global" level.
2. Make the group for a particular region a member of each country group within that region.
3. Make the global group a member of each region group.
4. Use RBAC to assign the required rights to each country and region group, as well as to the global group. This way, each region group will inherit the rights of the country groups within it, and the global group will inherit the rights of each region group.

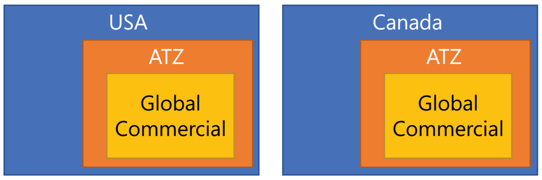


FIGURE 3-2 How the groups must be nested: Global Commercial is a member of ATZ, which is a member of both USA and Canada.

Related security principles

- Leveraging existing components

Related patterns

- Use a centralized identity provider for authentication
- Assign roles to groups

Secrets management patterns

Secrets are important because they are the keys for accessing services and data. Protecting secrets is therefore essential. You could have the most secure authentication mechanism, but if you do not protect the credentials, then you could be compromised.

Unfortunately, protecting your secrets can be complicated. At a certain point, you need them, and that is exactly when a malicious actor could attack you. Therefore, this is one of those situations where sticking with known secure patterns is most important.

Use managed identities

Intent and motivation

Most applications contain multiple components and require some type of interaction among them. This communication typically involves some form of credentials or sensitive information.

For example, suppose you connect to a database. In that case, you need a connection string specifying the name of the server or of the cluster exposing the database and the credentials to access it. Here is the problem: how can you store these credentials securely?

One obvious answer is to store them in configuration files. Unfortunately, this is not secure, because someone could steal them. Of course, you could encrypt the files, but then you have the problem of protecting the encryption key, and so on.

Alternatively, with an IaaS, you could use Data Protection API to encrypt the configuration file. Unfortunately, this approach has a couple of drawbacks, too. First, DPAPI provides only partial security because users with enough rights access secrets protected by DPAPI. Second, it does not protect the secrets in memory.

If you base your solution on App Services, you could use the application settings. These provide some protection and eliminate the need to store secrets in configuration files. Still, they do not protect the secrets in memory. And again, users who have enough rights can read the secrets—for example from the Azure Portal.

Description

A better way to store secrets is to use managed identities. These are service accounts that are entirely managed by Azure AD.

With managed identities, you have no access to the password, and your application doesn't either. The platform automatically injects it when you send out a request using the managed identity.

Not all possible callers on Azure support managed identities, nor do all possible callees. For a list of callers and callees that do, see https://azsec.tech/hia. This list is updated continuously, so you'll want to check it often. For example, Azure Cosmos DB didn't support managed identities until recently.

Note See https://azsec.tech/laf and https://azsec.tech/akg for information on using RBAC and managed identities with Cosmos DB.

Managed identities can be system-assigned or user-assigned. The main difference is that system-assigned managed identities are dedicated to the service, while user-assigned managed identities can be shared with various services. Therefore, if you want to minimize management and have many instances of the same application, you can simply assign the same user-assigned managed identity to all of them.

Tip There is one scenario in which system-assigned managed identities are preferable to user-assigned managed identities: for root cause analysis after an incident. That's because system-assigned managed identities are specific to an instance of a service, which makes it easier to identify the instance affected by the attack.

You can assign managed identities to virtual machines (VMs). This doesn't mean that all applications hosted by the VM can automatically access resources using the assigned managed identity, however. You must write code to leverage this possibility. This typically involves sending a request to the Azure Instance Metadata Service to obtain a token for the required resource. See https://azsec.tech/dff for details on using managed identities with Azure VMs.

Note The URL to call to obtain a token from the Azure Instance Metadata Service is http://169.254.169.254/metadata/identity/oauth2/token. The Azure infrastructure automatically injects the assigned managed identity, without ever exposing them, even in the message metadata.

Examples

- Suppose your code can use managed identities and needs to access a resource that supports managed identities. In this case, you should not store the credentials anywhere. Instead, it would be best to directly use the managed identity to access said resource.

- If your code supports managed identities but not the resource, check whether the resource at least supports managed identities for administrative purposes. If so, you might still be able to use managed identities to retrieve some credentials that grant access to the data from the resource. Therefore, it might be possible to have an initialization phase to gather the credentials from the service with your managed identity via the control plane and then use them for any ensuing calls. For example, Microsoft recommended this approach for accessing Cosmos DB data before it introduced support for managed identities.

Note You might prefer this option to the alternative, which is to store the credentials in Azure Key Vault, because it reduces the exposure of the credentials and simplifies management.

Related security principles

- Attack surface reduction
- Economy of mechanisms
- Leveraging existing components

Related patterns

- Use a centralized identity provider for authentication
- Protect secrets with Azure Key Vault

Protect secrets with Azure Key Vault

Intent and motivation

As discussed with the previous pattern, you need some place to store your secrets. For example, you might need to store a private key associated with a certificate instead of a username and password. A typical example is a bring-your-own-key (BYOK) scenario, where you provide a key to be used in some way, like for encrypting an SQL Database using Transparent Data Encryption (TDE).

You cannot always use managed identities to achieve this. For example, a compute service or called resource might not support managed identities. Or if you have an off-the-shelf application hosted in IaaS, it might not have been modified to support managed identities, and therefore it cannot use them. For example, it sometimes takes Microsoft some time to implement support for managed identities in open source solutions, like Redis Cache, that are incorporated as Azure Services.

So, what to do?

Description

Whatever your situation, if you cannot use managed identities to access a resource directly, the best approach is to use Azure Key Vault—a secure and centralized storage for secrets and encryption keys. Azure Key Vault is more secure than the alternatives because it can be isolated and made inaccessible from the internet, even if your application needs to be exposed using private links. You can rely on various services to secure Azure Key Vault, like Microsoft Defender for Key Vault, which—among its other capabilities—can help identify anomalous interactions. It is also more secure than many alternatives because the Premium SKU provides hardware security module (HSM) capabilities to prevent attackers from stealing private keys.

Leveraging Azure Key Vault is not enough to make your solution secure. For example, as discussed in our coverage of the prefer managed identity principle, if you host code in an environment supporting managed identities, like a VM or an App Service, you should use a managed identity to call Azure Key Vault. If you need a key to access Key Vault, you have the problem of protecting that key, which becomes very difficult. With VMs, you can use DPAPI, but this is not possible with a PaaS solution, because they might be occasionally moved to other servers by the Azure infrastructure, rendering your secrets inaccessible.

Another critical decision you need to make is how many Azure Key Vault instances to use. Microsoft recommends dedicating multiple instances for your application—one for each environment. Our recommendation is to have even more of them, particularly for production environments. If your application has multiple layers, you might dedicate an Azure Key Vault to each of them to ensure that an attacker can get hold of only a few secrets in the event of a compromise. (See Figure 3-3.)

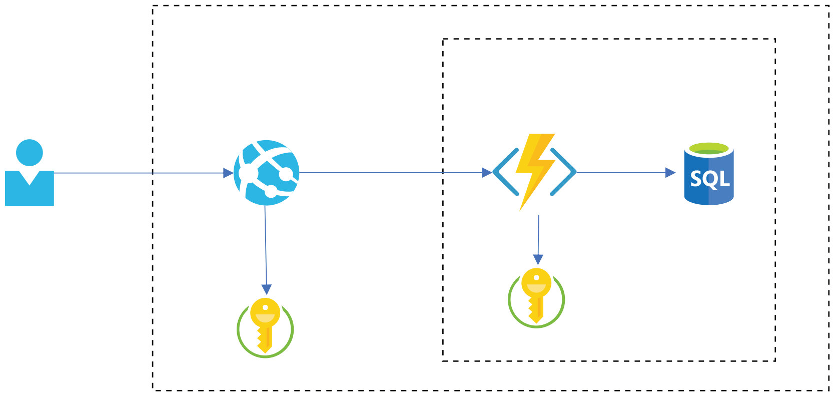


FIGURE 3-3 An example of a system with two Key Vaults: one for the front end and one for the back end.

This approach might seem radical and sometimes unnecessary. After all, having multiple Azure Key Vaults results in a more complex solution, introducing additional management burdens and increasing the possibility of mistakes by expanding the attack surface. So, the approach you decide to adopt depends on the characteristics of your solution.

Azure Key Vault for microservices

Microservices architectures present an interesting scenario. They are characterized by many small applications interacting with each other. If you had to apply a Key Vault for each microservice, the system would quickly become unmanageable. In this case, it might be best to avoid the fragmentation of the AKVs and dedicate just a single instance. Remember, per the economy of mechanisms principle, simpler is typically better.

Be aware that you might achieve significant compartmentalization even without dedicated instances. This is typically achieved by leveraging authorization. Azure Key Vault supports two authorization models: access policies and RBAC. With access policies, you assign access rights to whole categories of objects, like secrets or private keys. This means you cannot discriminate between secrets; you can read all of them or none of them. In contrast, with RBAC, you can assign roles even at a single object level. You can then have a single instance, because the object-level RBAC role assignment provides the granularity needed to determine what anyone can access.

Important The ability to assign access rights so granularly is excellent, but it has one significant downside: Azure supports up to 2,000 role assignments per subscription. So, it might be best to adopt RBAC (because it provides the greatest flexibility and control), but use it sparingly.

As discussed with our coverage of the use RBAC principle, you should use RBAC because it enables you to verify and manage role assignments centrally. Moreover, this comprehensive visibility enables you to adopt tools to identify potential risks and processes to manage role assignments more or less automatically.

Note Chapter 10, “Cryptography in Azure,” has more details on Azure Key Vault.

Examples

Azure Key Vault is great for storing secrets to access resources that don’t support managed identities, like Azure Cache for Redis. Azure Key Vault is also effective for storing secrets that you cannot replace with managed identities, like connection strings or private keys. When you store private keys, you should choose the Premium SKU to leverage its HSM capabilities. The Standard SKU is enough for all other scenarios.

Related security principles

- Fail secure
- Least privilege
- Leveraging existing components

Related patterns

- Use role-based access control (RBAC)
- Use managed identities
- Use bring your own key (BYOK)

Sensitive information management patterns

Secrets are important, but the information they protect is even more critical. There are many options for securing your data, but not all options are born equal, and it can be tricky to choose the right ones. Consider, for example, data protection at rest: all storage options from Azure allow or even enforce encryption of data at rest. For example, Azure encrypts Azure Managed Disks automatically. You have only a few options to determine how they are protected and who provides the encryption key. But do you need this type of encryption? Of course, the answer is yes, but things aren't so simple. You have to satisfy multiple needs for encryption, and this sort of protection addresses just a few of them. Again, relying on patterns is important to correctly address the most significant problems.

Create secure channels

Intent and motivation

Communications are usually the main risk for a solution. If you have a closed box that does not receive any input and does not provide any output, it is fully protected and will not be attacked. Unfortunately, such an isolated system makes sense only as an intellectual exercise.

All practical applications of technology are connected to *something*. It could be the internet, a local network, or even just a power outlet. Even devices that aren't connected to anything, because they use a battery and have their Wi-Fi connectivity disabled, still send out information through screens or simply through the electromagnetic field emitted by their electronics.

In Chapter 2, you learned that the Azure security model assigns customers partial responsibility for the security of the solutions it hosts. But whatever model you choose—IaaS, PaaS, or SaaS—it is Microsoft's responsibility to protect the physical devices. Good news, then: you do not need to be concerned about power connections and electromagnetic fields! Still, all the other concerns are well within your scope.

Description

What does it mean to create secure channels? Or better, how can you determine whether a channel is secure? There are a few requirements that must be satisfied:

- **Confidentiality** The channel must preserve data confidentiality. Transmission over the internet moves through many systems, and each of these systems could potentially read and disclose the contents of the communication. To preserve confidentiality, you must wrap the content such that unauthorized parties cannot read it, typically through encryption.

- **Authentication** The channel must authenticate with certainty all parties that are privy to the communication. In most situations, the main concern is the authentication of the caller, but this should not be the case. You also need to ensure that the called service authenticates itself. Fortunately, the adoption of TLS provides the authentication of the service implicitly. Still, you might need to ensure that your callers perform additional checks on the provided certificate—for example, checking whether the certificate has been issued by the expected certification authority.
- **Integrity** You must confirm the integrity of transmitted information. In other words, those who receive it must check whether it has been modified by some third party while in transit. Transmission protocols usually split big messages into multiple packets. Therefore, some of these packets may be lost in transit or received in a different order than expected. To preserve integrity, you must ensure that what is received matches what has been sent.

Azure addresses some of these requirements by imposing channel encryption for most communications. With channel cryptography, like TLS, you can provide confidentiality, integrity and server authentication. But you still have to address client authentication, which is optional in TLS..

There are various ways to authenticate the client—too many to include them all here. But it might be helpful to talk about one of them in particular: client certificates. These establish a strong connection that prevents man-in-the-middle attacks. If you don't use client certificates, an attacker can intercept a communication, terminate the TLS channel at its end, and create a new, false one directed toward the server. If you do use client certificates with TLS, however, this would not be possible because the man in the middle would not present your certificate to the target server. (Admittedly, we rarely use client certificates because they involve a high cost.)

Another approach is to create secure channels and isolate them from the internet. You can then use these secure channels to prevent the exposure of your resources and services over the internet. You can typically achieve this by using VPNs or ExpressRoute.

The likelihood of interception

How likely is it that an attacker will intercept your traffic? After all, the internet is vast, with billions of messages sent every second. Intercepting *your* traffic isn't like finding a needle in a haystack; it's harder! So, how big of a problem is it, really?

The truth is, there are a couple situations in which the chances of your traffic being intercepted raise significantly:

- The attacker might just be in the right place at the right moment, and you might unknowingly send your traffic their way. This could happen if you use an easily intercepted channel, like Wi-Fi, when the attacker is near you. This type of attack occurs more frequently than you might imagine—particularly in specific situations, like at security conferences.
- You might unknowingly send your traffic through malicious nodes. A typical example of this situation is when you use Tor or some free VPN. Malicious actors host many of these resources, and they use them to tap into your traffic and potentially perform malicious actions.

Examples

- Configure end-to-end TLS with Azure Application Gateway to ensure that TLS encryption is provided internally between Application Gateway and the target service or resource.
- Use a point-to-site VPN to connect a single workstation to resources on Azure to avoid exposing them over the internet.
- Use a site-to-site VPN to connect a site, which could be your office, to cloud resources not exposed over the internet. This typically requires the installation of VPN gateways to collect the traffic on both sides.
- Use ExpressRoute to connect a site similarly to the site-to-site VPN. The main difference is that ExpressRoute is based on a dedicated infrastructure, so the connection is typically faster and more reliable.

Related security principles

- Defense in depth

Related patterns

- Encrypt data client-side
- Use bring your own key (BYOK)

Encrypt data client-side

Intent and motivation

As discussed at the beginning of this section, Azure uses encryption to protect all storage. This is obviously a great feature, but does it address all your needs? To answer this question, you must first identify exactly what needs this sort of encryption *does* address.

The encryption at rest provided by Azure for most storage options falls under the category of transparent encryption. In other words, it ensures that if you access the storage using one of the sanctioned channels, data will be available in unencrypted form. If you access the storage using any other means, however, the data will unreadable. For example, if you try to access the data by stealing the virtual or physical disks, you will wind up with encrypted content that is not readable, even if you have the required rights.

So, transparent encryption increases the isolation of the customer data from Microsoft management environments. More specifically, it makes it more difficult for Microsoft's administrators to get to your data. However, there are other needs that transparent encryption does not address. For example, data in memory remains unencrypted.

Data can assume three states, and you have to protect all three of them:

- **At rest** This is when the data is stored somewhere.
- **In transit** This happens when you transmit the data between two locations.
- **In memory** This is when the data is temporarily stored in computer memory, ready for processing.

Transparent encryption only protects data at rest. When you use transparent encryption, the data is unencrypted in memory for both the database or storage server and the client. So, you typically protect data in transit by adopting TLS, which Azure enables by default.

Still, if a malicious actor manages to get hold of some authentication material, that person could access the data stored in some repository. Of course, there are a few conditions for this to happen. For example, the malicious actor must have access to the repository. But when these conditions are met, the malicious actor can read the data. This is a scenario that, in most cases, could be safely considered "already mitigated." Still, if your solution requires a higher level of security assurances, you need to consider something else, like client-side encryption.

Description

The idea behind client-side encryption is that you encrypt the data on the client before sending it to the storage system. This ensures that the data is encrypted from that point on, including in the repository memory. Of course, the application must decrypt the data to consume it. At that point, the data would be potentially at risk.

If your storage system is a database server, things become interesting. Because the client-side encrypted data is not readable by the logic executed on the database server, it would be impossible to perform typical activities like searching its content. Still, with some implementation of client-side encryption, like Cosmos DB and SQL Server Always Encrypted, it is possible to perform limited comparisons.

The implementation of client-side encryption is possible when the platform supports it, and it might still be achievable as a custom activity. But in that case, it should be treated as a delicate task, requiring thorough testing and in-depth validation by experts, because it is possible to make fatal mistakes. And even under the best conditions, client-side encryption can have a significant impact on the performance of the system, due to computational costs and because it may be impossible to index data to improve search speed.

TIP When you implement client-side encryption, a typical pattern is to consider adopting Azure Key Vault for storing the cryptographic keys.

Examples

- Cosmos DB and SQL Server Always Encrypted provides a way to encrypt data client-side. It includes a mode called *deterministic encryption*, which allows to search for records having some specific value. For example, if you encrypt a tax code with deterministic encryption, you can get all rows in a table where a field has the same tax code.

Note The very characteristic that makes deterministic encryption worthwhile is also its main weakness. For example, an attacker could group all the records associated with the same tax code, which could provide them with enough information to identify a person from the other metadata. For this reason, consider *randomized encryption* as the first option and revert to deterministic encryption if necessary.

- Some libraries that provide programmatic access to Azure Storage implement the necessary logic to perform client-side encryption for Azure Storage. For examples of this with .NET, Java, and Python code, see https://azsec.tech/ci5.

Related security principles

- Defense in depth
- Fail secure
- Leveraging existing components

Related patterns

- Protect secrets with Azure Key Vault

Use bring your own key (BYOK)

Intent and motivation

Data encryption has multiple roles. One of the least commonly considered is crypto-shredding. Crypto-shredding involves deliberately deleting or overwriting encryption keys used to secure sensitive data. In this way, you can ensure that nobody can read the data you no longer need, or you can block data exfiltration in an emergency. With crypto-shredding, you can make it impossible for anyone to access your data.

Description

One of the best ways to do this is to use BYOK. With this approach, Azure Key Vault stores the key, which is under your control. So, to crypto-shred your data, you simply purge your key from its Key Vault. When you do, all that data becomes immediately unrecoverable.

More on BYOK

One of the primary purposes of BYOK is to increase users' control over their keys. The idea is that by introducing their own keys, users make it harder for the cloud service provider to access their data. Unfortunately, that's not why BYOK is useful. If you do not trust Microsoft with your data, why should you trust it to handle your key securely? You are storing it in its systems. In other words, if you assume that a malicious administrator could access your data, you should also assume that person could access your keys, too. Fortunately, Azure includes many other controls to prevent this from happening, including strong physical controls and various isolation layers that require escalation paths under the control of the customer to access the data.

Of course, this is a two-edged sword, because a malicious actor could leverage this approach to cause a denial of service. Some SaaS solutions like Microsoft 365 implement mechanisms to prevent losses due to the destruction of the BYOK. One such feature is called Availability Key and is discussed here: https://azsec.tech/89t.

Example

If you have an Azure SQL or Azure Storage and have used BYOK to encrypt them at rest, you can remove the key from Azure Storage to crypto-shred the content.

Related security principles

- Defense in depth
- Leveraging existing components
- Separation of duties

Related patterns

- Protect secrets with Azure Key Vault

Availability pattern

Availability is often considered a given. You expect your solution to be there when needed and provide its services to every user. Unfortunately, it is not so automatic to achieve this, and you have to work hard to guarantee that your solution is up and running.

In the cloud, the situation is even worse, if possible. When you develop a complex system integrating many services, you rely on all of them to be available. The unavailability of any service may cause your solution to be partially or entirely unavailable. When each of these systems is managed at least in part by a third party, you lack control over the maintenance activities. Therefore, you have to design your solution to be more resilient than you used to do with on-prem solutions.

Important It would be a mistake to think that this problem is specific to the cloud or to Azure. On the contrary, the pervasive adoption of automation you have with these platforms has dramatically reduced the prevalence of the incidents compared to on-prem. Still, incidents are a possibility and maintenance a necessity. The new approach gives you this awareness and the tools to design and implement your solutions in a resilient way to meet your availability requirements.

Let’s see what you can do to improve the availability of your solution.

Design for denial of service

Intent and motivation

Denial-of-service (DoS) attacks are common occurrences. They happen when someone creates the conditions for your solution to fail by bombarding it with more requests than it can handle or by sending artfully crafted messages causing your solution to crash. In any case, these attacks cause the unavailability of your service. A variant of DoS attacks is called *distributed DoS* (DDoS). DDoS attacks are characterized by the generation of the attack from multiple points, sometimes in the order of the thousands or tens of thousands.

DoS and DDoS are some of the easiest attacks to execute. Some organizations even provide DDoS attacks as a service. You tell them who the target is, you pay them, and they do the rest! Very convenient and powerful.

You can address this problem in various ways, but the easiest is to simply add more resources to your application. One of the characteristics of the cloud is elasticity, which means you can allocate resources dynamically, as you need them. However, although this is easy and fast, it can be expensive and unfeasible in the long run. Here is where this pattern becomes useful.

Description

All public cloud platforms, including Azure, offer a base level of protection from DoS attacks. Azure also offers Azure DDoS Protection Standard (https://azsec.tech/s1h), designed to protect public IP addresses from potentially massive DDoS attacks.

The documentation for this service uses some specific wording that you should be aware of:

> “Azure DDoS Protection Standard, combined with application design best practices, provides enhanced DDoS mitigation.”

Note that it says *enhanced*, not *complete*. In other words, as with any other anti-DDoS system, Azure cannot represent your only line of defense against DDoS attacks. These systems complement a more comprehensive strategy, which starts with the design of your solution. For this reason, you should not design a system without thinking about how your architecture might respond to a DDoS attack.

Consider this real-life example: a customer with nearly 100 public IP addresses protected by the Azure DDoS Protection Standard service suffered severe service degradation. This was due to a DDoS attack, but it went undetected because the attack was "low and slow." That is, it fell below the triggering threshold for Azure DDoS Protection Standard, so none of the service's mitigation policies were triggered. As a result, the combined traffic flow across all IP addresses overloaded the back-end NVAs, which forced them to drop part of the traffic. The security anti-pattern in this design takes data from multiple untrusted sources and concentrates that traffic on a single, internal endpoint. In this case, all that data was concentrated on a single VM running the NVA, which is where the impact of the attack became evident. The customer quickly applied the NVA's built-in auto-scaling capabilities to bring more network bandwidth and compute online. In this scenario, a range of one to three NVAs mitigated the attack.

Note To see Microsoft documentation on including DoS and DDoS protection support in your designs, see https://azsec.tech/xin.

One final note: many Azure services can throttle requests. For example, Azure Key Vault allows 4,000 secret transactions (for example, reading an SQL connection string) per 10 seconds. If your code goes beyond this threshold, further requests are throttled and return a 429 ("Too Many Requests") response. To remedy this, cache data if possible. In the Key Vault example, you could cache the connection string in memory for 15 minutes and read from Key Vault only four times per hour, which is way under the threshold. You can find a list of Azure subscription and service limits, quotas, and constraints at https://azsec.tech/9it.

Examples

- Many services can be configured with networking rules or private endpoint connections. One such service is Azure Storage. If you opt to adopt the networking rules, the service itself is still exposed and can receive requests; you just have a high-speed mechanism to reject the requests because they are not from an acceptable IP address. While fast, this mechanism can still be overwhelmed or bypassed—for example, by an IP spoofing attack. Therefore, it is typically better to use private endpoints.
- Consider the availability requirements of your solution. If you have strict availability requirements that do not allow for partial unavailability, it's best to design the solution accordingly. For example, you could use a content delivery network (CDN) to serve the static content and a distributed and redundant architecture to provide the service to the relevant geographies.

Related security principles

- Attack surface reduction
- Defense in depth

- Single point of failure
- Weakest link

Related patterns

- Isolate from the internet
- Isolate with an identity perimeter

Summary

This chapter introduced Azure security patterns. It started with a discussion of why the concept of patterns is so essential. It then introduced a few of the most common patterns to secure your solution on Azure.

This list is limited, and you might be able to identify one or two significant ones that are missing. The intent here is not to be comprehensive but to introduce you to an approach for building secure software on Azure based on the adoption of renowned patterns.

The next chapter goes beyond this to introduce a different way of thinking about security and designing secure solutions: threat modeling.

Index

Numbers

429 ("Too Many Requests") messages, 76

A

AAD (Azure Active Directory)
- AAD Data Plane RBAC, 416–419
- identity management, 203

ABAC (Attribute-Based Access Control), 168–170
acceptability, psychological, 45
accepting risk, PCI DSS, 218
access
- ABAC, 168–170
- AD Access Reviews, 42
- Azure Key Vault, 288–290, 301
- broken access control, 257
- CAE, 131
- Entra, 130
- managing, modern identity, 125
- PAM tool, 56–57
- PAW, 42
- privileged access, 203
- RBAC, 61–62, 206, 210–211, 291–299
- rights, limiting duration of, 28–29
- SAS tokens, 158
- tokens, 138
- without authentication, 151–159

accounts, break-glass accounts, 298
ACI (Azure Container Instances), 377–378
ACR (Azure Container Registry), 385–386
action groups, 185–186
Active Directory Authentication Library (ADAL), 131
AD (Active Directory)
- AAD, identity management, 203
- Access Reviews, 42
- ADAL, 131
- PIM, 46, 168
- roles, 162–164
- scopes, 162
- SecOps, 125

ADAL (Active Directory Authentication Library), 131
adding audit logs with Azure Policy, 189
Addiscott, Richard, threat modeling, 89
administration, just-in-time, 56–58
ADO SHA (Azure DevOps Self-Hosted Agents), 456
Agile development, characteristics of, 23
Agile SDL (Security Development Lifecycle)
- attack surface analysis, 10
- banned functionality, avoiding, 12–13
- bug bars, defining, 7–10
- CVSS, 7
- DTD bomb attacks, 13
- dynamic analysis tools, 16–17
- flow analysis, 17
  - control-flow analysis, 17–18
  - data-flow analysis, 18
- incident response plans, 18
- overview, 20–21
- penetration testing (pentests), 19
- security training, 6
- static analysis tools, 13–16, 17
- tasks by sprint, 20–21
- technical security debt, 19
- threat modeling, 10
- toolchains, defining, 11

agility, cryptographic, 312–313
  CredScan, 313–314
  delimiters, 323
  implementing, 315–323
  metadata, 316–317
  nonclass factory cryptographic algorithms, 317–320
  Office (MS) documents, 317
  process of, 314–315
  real-world experience, 322–323
  summary, 323–324
  TLS, 343
  version numbers, 320–322
AKS (Azure Kubernetes Services), 107, 378–380, 457
alerts, 197–199, 303
Alexander, Christopher, design patterns, 51
all input is evil, 238–239
allLogs, 175
allowed communications, VNets, 448
Always Encrypted, 422–426, 429–433
AMD, SEV-SNP, 362–363
analysis phase, threat modeling, 82
analyzing
  attack surfaces, 10
  dynamic analysis tools, 16–17
  flow analysis, 17
    control-flow analysis, 17–18
    data-flow analysis, 18
  SCA tools, CI/CD, 436
  static analysis tools, 13–16, 17
  use analysis tools, 271–273
API (Application Programming Interface)
  fuzz testing, 280–283
  Management Gateways, 451
  REST API security, 282–283
App Services, 47, 377
Application Programming Interface (API)
  fuzz testing, 280–283
  REST API security, 282–283
Application Security Groups (ASG), 447
applications
  AppServiceAppLogs, 173
  ASG, 447
  authentication, 159–161
  Azure Application Proxy, 451
  Azure Container Apps, 378
  DAST, 16–17
  hosting, 133
  OAuth2
    flows, 136–137
    registration, 129
  permissions, 138
  roles, 62–64
  shared app service plans, 455
AppServiceAppLogs, 173
AppServiceAuditLogs, 173
AppServiceConsoleLogs, 173
AppServiceHTTPLogs, 173
AppServiceIPSecAuditLogs, 173–174
AppServicePlatformLogs, 174
Arm TrustZone, 363
ASB (Azure Security Benchmark), 202, 227–228
ASG (Application Security Groups), 447
assessing risk, bug bars, 7–10
asset management, 204
assigning
  Azure Policies, 211–212
  roles
    best practices, 167–168
    Blueprints, 167
    denying assignments, 167
    managing assignments, 164–165
    to groups, 58–59
assume-breach, 29
assumptions, mitigations, 113
ATT&CK, 231
attack surfaces
  analysis, 10
  reduction, 32–33
attacks. *See also* threat modeling
  brute-force attacks, 350–351
  DoS attacks, 75–77, 391
  DTD bomb attacks, 13

elevation of privileges attacks, 391
honeypots, 171
reconnaissance attacks, 171
repudiation attacks, 391
spoofing attacks, 391
supply chain attacks, 436
tampering attacks, 391
attestation, remote, 360, 364–366, 431
Attribute-Based Access Control (ABAC), 168–170
audit logs, 175, 186–188
auditing
allLogs, 175
AppServiceAppLogs, 173
AppServiceAuditLogs, 173
AppServiceConsoleLogs, 173
AppServiceHTTPLogs, 173
AppServiceIPSecAuditLogs, 173–174
AppServicePlatformLogs, 174
audit logs, 175, 186–188
Azure Key Vault, 301–303
Azure Monitor
Azure Storage, 174, 187–188
diagnostic settings, 172–175
Event Hub, 174
Log Analytics workspaces, 174, 176
Log Analytics workspaces, action groups, 185–186
Log Analytics workspaces, KQL queries, 176–180
Log Analytics workspaces, protecting audit logs, 186–188
Log Analytics workspaces, raising alerts, 181–185
Partner Solution, 174
Azure Policy, adding audit logs, 189
Azure Sentinel, 186
category groups, 175
CMK, 186–187
control (management) plane, 399–400
Cosmos DB Security, 414, 419–420
costs, 189–190
crypto-shredding, 186
data plane, 404
intentional security monitoring/auditing, 190
KQL queries, 176–180
logging for auditing, defined, 172
SQL Server and database security, 395
threat modeling events, 190–199
authentication (authN), 123–124, 150–151, 154
access without authentication, 151–159
ADAL, 131
applications, 159–161
control (management) plane, 396–398
Cosmos DB Security, 411–412
data plane, 402–403, 414–415
defined, 124
failures, 264–266
importance to developers, 155
keys, 338
MFA, 152–153
MSAL, 130–132
ADAL, 131
CAE, 131
debugging tools, 132–133
passwordless authentication, 152
personal solutions, creating, 155–156
SAML, 129
SAS tokens, 158
security patterns, 53–55
Azure AD, 54–55
centralized identity providers, 54–55
server authentication, 154
SMS-based authentication, 151–152
SQL Server, 395
SSO authentication, 156–157
threat modeling, 107
zero trust, 28
ZKPP, 151
authorization (AuthZ), 123–124, 161–162, 395, 403–404
code flows, 134–135
control (management) plane, 398–399
Cosmos DB Security, 413, 415–418
defined, 124
FedRAMP, 218–219, 221
policy-based authorization, 28

security patterns, 53–55
servers, OAuth2, 134
automation
container development/deployment, 384
deployments, governance, 206
threat identification, 113–115
threat modeling, 91
availability patterns, 74–75
Azure
AD, 54–55
Access Reviews, 42
ADAL, 131
PIM, 46, 57, 168
roles, 162–168
scopes, 162
SecOps, 125
AKS, 107, 378–380
App Services, 47
App Services Web App Containers, 377
ASB, 202
Blueprints, 167
confidential containers, 380
Confidential ledger, 368–370
Container Apps, 378
container services, 375–376
cryptographic services (overview), 329
Customer Lockbox, 25–26, 46
Dedicated Hosts, 41
dedicated hosts, 27
Disk Encryption, 336
Function Cointainers, 377
Initiatives, 210
IPv4 addresses, 445–446
Key Vault, 66–69, 76
FIPS 140 and managed HSM, 224
PaaS, TLS, 345–346
Policy, 209
assigning policies, 211–212
effects, 210
effects, enforcing by environment, 210–211
policy as code, 212
Portal, compliance state, governance, 208–209
RBAC, 61–62, 291–299
rules of engagement, 275
secure design, 32
Security Benchmark, 80, 81
security patterns, 51–52
Sentinel, 186
alerts for custom events, 197–199
Shared Responsibility Model, 24–25
Storage, cryptography, 331–335
storage, redundancy levels, 47
Storage, Azure Monitor, 174
protecting audit logs, 187–188
Storage Keys, 107
VM
cryptography, 335–337
Intel SGX, 361–362, 363–364
Intel SGX, keys, 364
SEV-SNP, 362–363
trusted launches, 366–367
Well-Architected Framework, 53
Azure Active Directory (AAD), AAD Data Plane RBAC, 416–419
Azure Application Proxy, 451
Azure Container Instances (ACI), 377–378
Azure Container Registry (ACR), 385–386
Azure DevOps Self-Hosted Agents (ADO SHA), 456
Azure Firewall, 449–450
Azure Firewall Premium SKU, 450
Azure Key Vault, 66–69, 76, 288
access control, 288–290, 301
auditing, 301–303
backups, 306–307
break-glass accounts, 298
“bring your own key” strategy, 301
certificates, 288
compound identities, 299
contributor permissions, 297
customer-managed keys, 306
elliptic-curve keys, 301
encrypting operations, 290
FIPS 140 and managed HSM, 224
get operations, 291–299
honey keys, 303

HSM
Dedicated HSM, 309
Managed HSM, 308–311
Payment HSM, 309
keys, 288, 301
logging, 301–303
Microsoft Defender, 306
network isolation, 304–305
permission models
selecting, 295
switching, 292
Premium edition, 299–301
RBAC, 291–299
restoring key versions, 307–308
rotating keys, 339–341
RSA, 301
secrets, 288
secure keys, 312
Standard edition, 299–300
templates, 300
verifying operations, 290
wrapping operations, 290
Azure Kubernetes Services (AKS), 378–380, 457
Azure Monitor
Azure Storage, 174, 187–188
diagnostic settings, 172–175
Event Hub, 174
Log Analytics workspaces, 174, 176
action groups, 185–186
KQL queries, 176–180
protecting audit logs, 186–188
raising alerts, 181–185
Partner Solution, 174
Azure Policy, adding audit logs, 189
Azure Security Baseline, 411, 421–422
Azure Security Benchmark (ASB), 227–228
Azure SQL Database, 394–395
Azure SQL Ledger, 409–410
Azure SQL Managed Instance, 395
Azure WAF (Web Application Firewalls), 450–451

B

backlogging, mitigations, 119–122
backups
Azure Key Vault assets, 306–307
governance, 205
bad data (known), rejecting, 253–255
banned functionality
avoiding, 12–13
examples of, 12
benchmarks
ASB, 202, 227–228
Azure Security Benchmark, 80, 81
CIS Benchmarks, 80–81, 226–227
binaries, signing, 268–269
Blueprints, 167
bomb attacks, DTD, 13
bounty programs, Hyper-V, 40–41
break-glass accounts, 298
breaking keys/passwords, cost of, 350–351
Britt, Jim, Azure Policy, 189
broken access control, 257
brute-force attacks, 350–351
bug bars, defining, 7–10
BYOK (Bring Your Own Key), 73–74

C

C programming language, 270–271
C++ programming language, secure coding, 270–271
CAE (Continuous Access Elevation), 131
CAIRIS, threat modeling, 93–94
CAPEC (Common Attack Pattern Enumeration and Classification), 231
Carielli, Sandy, threat modeling, 89
Carmack, John, static analysis tools, 14
Center of Internet Security (CIS), benchmarks, 80–81
centralized identity providers, authentication, 54–55
certificates, keys, 288

Chambers, John T., hacks, 172
channels, secure, 69–70
CI/CD (Continuous Integration/Continuous Deployment)
  defined, 435
  deployment agents, 440–441
  developers, specialized security, 436–437
  main branch (trunk) security, 438–439
  PR approvals, 437
  PROD deployments, 439–440
  SCA tools, 436
  secrets and service connections, 438
  separation of duties, 437–438
  source control systems, 436
  supply chain attacks, 436
  tools (overview), 435–436
  trunk (main branch) security, 438–439
CIDR, IPv4 addresses, 445–446
ciphersuites, 286, 343–344, 345, 346–349, 356
CIS (Center of Internet Security), benchmarks, 80–81, 226–227
classification, data discovery, 409
clients
  confidential clients, 135
  credential flows, 135
  cryptography, 331
  data encryption, 71–73
  OAuth2, 134, 137
    confidential clients, 137–138
    public clients, 137
cloud computing
  Microsoft DFC, 207, 208
  secure design, 23–24, 28–29
clusters
  containers, 386
  network policies, 457–458
CMK (Customer-Managed Keys), 186–187, 204, 330
CNAME (Canonical Names), 459–460
code security
  all input is evil, 238–239
  attackers
    identity of, 240–241
    what attackers control, 241–242
  C programming language, 270–271
  C++ programming language, 270–271
  fuzz testing, 274, 275–276
    API, 280–283
    Azure rules of engagement, 275
    generating random data, 276–277
    manipulating data by format, 280
    mutating existing data, 277–280
  hashes in coding, 341
  insecure coding, 237
  reviewing, 273–274
  revocation checking, 266
  threat modeling, 239–242
  use analysis tools, 271–273
  verifying data, 242
    determining correctness, 243–253
    encoding data, 256
    rejecting known bad data, 253–255
  vulnerabilities, 256–257
    authentication failures, 264–266
    broken access control, 257
    checking signatures, 268
    cryptography failures, 257–258
    data integrity failures, 266–269
    debugging security headers, 264
    identification failures, 264–266
    injection, 258
    inline scripts/styles, 260–263
    insecure design, 259
    logging failures, 269
    misconfigured security, 259–260
    monitoring failures, 269
    signing binaries, 268–269
    software failures, 266–269
    SSRF, 269–270
    vulnerable/outdated components, 264
code verifiers, 135
Coles, Matthew J., threat modeling, 81
column encryption, 408
comment fields, validating, 249–250
Common Attack Pattern Enumeration and Classification (CAPEC), 231
common vulnerabilities, 256–257

authentication failures, 264–266
binaries, signing, 268–269
broken access control, 257
cryptography failures, 257–258
data integrity failures, 266–269
debugging security headers, 264
identification failures, 264–266
injection, 258
inline scripts/styles, 260–263
insecure design, 259
logging failures, 269
misconfigured security, 259–260
monitoring failures, 269
signatures, checking, 268
software failures, 266–269
SSRF, 269–270
vulnerable/outdated components, 264
Common Vulnerabilities and Exposures (CVE), 230
Common Vulnerability Scoring System (CVSS), 7, 230
Common Weakness Enumeration (CWE), 231
communications (allowed), VNets, 448
compensating controls, 286
complete mediation, 33–34
complexity, containers, 381–382
compliance, 232–233
ASB, 227–228
ATT&CK, 231
CAPEC, 231
CIS benchmarks, 226–227
CVE, 230
CWE, 231
defined, 213–215
FedRAMP, 218–219, 221
FIPS 140, 221–222
Azure Key Vault and managed HSM, 224
SHA-2 in .NET, 222–223
GDPR, 216–217
HIPPA, 215–216
HITRUST, 216
ISO/IEC 27001, 225–226
ISO/IEC 27034, 226
MITRE, 229–230
NIST Cybersecurity Framework, 220–221
NIST SP 800–53, 219–220, 221
NVD, 230
OWASP, 229
PCI DSS, 217–218
SOC, 224–225
threat modeling, 233–234
compliance state, reviewing, 208–209
components
leveraging existing components, 42–43
vulnerable/outdated, 264
compound identities, 299
compromises
developers, 3–4
reasons for, 171–172
sources of, 3–4
confidential clients, 135, 137–138, 359
confidential computing
adopting, 360
Arm TrustZone, 363
Confidential containers, 370–371
Confidential ledger, 368–370
DCsv3 VM, 363–364
defined, 359–360
Intel SGX, 361–362, 363–364
Intel TME-MK, 363–364
memory isolation/encryption, 360
processors, 361
remote attestation, 360, 364–366
resources, 360
secure key management, 360
SEV-SNP, 362–363
TEE code, 361–362
trusted launches, 360, 366–367
VBS, 368
confidential containers, 370–371, 380
Confidential ledger, 368–370
connection strings, SQL, 402
consent
incremental consent, 140
OAuth2, 139–140

containers, 375, 388
  ACI, 377–378
  ACR, 385–386
  AKS, 378–380
  Azure App Services Web App Containers, 377
  Azure confidential containers, 380
  Azure Container Apps, 378
  Azure container services, 375–376
  Azure Function Containers, 377
  clusters, 386
  complexity, 381–382
confidential containers, 370–371, 380
  defined, 373–374
  Docker, 373–374
  fragmentation, 383
  IaaS, 376
  images, 385
  immaturity, 383
  K8s, 373–375
  Managed OpenShift, 381
  Microsoft Defender for Containers, 386
  pods, 387–388
  problems with, 381–383
  resources, 375
  service security
    deployments, 384–385
    development, 384–385
    overview, 383–384
Content-Security-Policy (CSP), 260
Continuous Access Elevation (CAE), 131
Continuous Integration/Continuous Deployment (CI/CD)
  defined, 435
  deployment agents, 440–441
  developers, specialized security, 436–437
  main branch (trunk) security, 438–439
  PR approvals, 437
  PROD deployments, 439–440
  SCA tools, 436
  secrets and service connections, 438
  separation of duties, 437–438
  source control systems, 436
  supply chain attacks, 436
  tools (overview), 435–436
  trunk (main branch) security, 438–439
contributor permissions, 297
control (management) plane, 392, 393
  auditing, 399–400, 414
  authentication, 396–398, 412–413
  authorization, 398–399, 413
  cryptography, 400–401
  network isolation, 401–402, 414
  permissions, 161
control-flow analysis, 17–18
correctness (verifying data), determining, 243
  error handling, 253
  Greek question mark, 251–252
  HTML, 251
  names, security decisions based on, 253
  namespace std, 245–246
  real-world experience, 243–245
  validating
    comment fields, 249–250
    date and time, 247
    high-level validation tools, 252
    open spots, 248–249
    vaccination center ID, 246
    vaccination type, 248
Cosmos DB Security, 411–412
  Always Encrypted, 431–433
  auditing, 414
  authentication, 412–413
  authorization, 413
  Azure Security Baseline, 422
  cryptography, 414, 421
  data plane
    auditing, 419–420
    cryptography, 420
    network isolation, 420–421
  data plane authentication, 414–415
  Microsoft Defender for Cosmos DB, 421–422
  network isolation, 414
CredScan, 313–314
Cross-Site Scripting (XSS), 30, 258
crypto shredding, 407

cryptography, 391
agility, 312–313
CredScan, 313–314
delimiters, 323
implementing, 315–323
metadata, 316–317
nonclass factory cryptographic algorithms, 317–320
Office (MS) documents, 317
process of, 314–315
real-world experience, 322–323
summary, 323–324
TLS, 343
version numbers, 320–322
Azure Disk Encryption, 336
Azure Key Vault, 288
access control, 288–290, 301
auditing, 301–303
backups, 306–307
break-glass accounts, 298
"bring your own key" strategy, 301
certificates, 288
compound identities, 299
contributor permissions, 297
customer-managed keys, 306
Dedicated HSM, 309
elliptic-curve keys, 301
encrypting operations, 290
get operations, 291–299
honey keys, 303
logging, 301–303
Managed HSM, 308–311
Microsoft Defender, 306
network isolation, 304–305
Payment HSM, 309
Premium edition, 299–301
RBAC, 291–299
restoring key versions, 307–308
RSA, 301
secrets, 288
secure keys, 312
selecting permission models, 295
Standard edition, 299–300
switching permission models, 292
templates, 300
verifying operations, 290
wrapping operations, 290
Azure services (overview), 329
Azure Storage, 331–335
Azure VM, 335–337
ciphersuites, 286, 343–344, 345, 346–349, 356
client-side cryptography, 331
compensating controls, 286
complexity, 286–287
control (management) plane, 400–401
Cosmos DB Security, 414, 421
CredScan, 313–314
data at rest, 406–407
data in transit, 341–343
data plane, 405, 420
database security, 393, 395–396
defined, 285–286
DEK, 329
delimiters, 323
E@H, 336–337
ECDSA, 353
failures, 257–258
hashes in coding, 341
KEK, 329
keys
agreements, 344
certificates, 288
defined, 287
exchanges, 344
secrets, 288
security, 287
metadata, 316–317
Microsoft Data Encryption SDK, 324–329
nonclass factory cryptographic algorithms, 317–320
Office (MS) documents, 317
PFS, 344–345
rotating keys, 337–341
RSA, 353
SQL Server and database security, 395–396

SSE, 335–336
CMK, 330
PMK, 329–330
SSL, 342
TLS, 342, 345
.NET code, common mistakes, 354
cryptographic agility, 343
debugging, 354–356
IaaS, 350
Linux VM, 350–351
PaaS, 345–346
testing, 354
versions of, 342–343
Windows VM, 352–353
unsecured SSH, 357
version numbers, 320–322
crypto-shredding, 186
CSP (Content-Security-Policy), 260
custom events
alerts, 197–199
threat modeling, 193–197
custom role definitions, 165–166
Customer Lockbox, 25–26, 46
Customer-Managed Keys (CMK), 186–187, 204, 330
CVE (Coimmon Vulnerabilities and Exposures), 230
CVSS (Common Vulnerability Scoring System), 7, 230
CWE (Common Weakness Enumeration), 231
Cybersecurity Framework, NIST, 220–221

D

dashboard, Threats Manager Studio, 118–119
DAST (Dynamic Application Security Testing), 16–17
data at rest, cryptography, 406–407
data discovery/classification, 409
data encryption
Azure Disk Encryption, 336
client-side data encryption, 71–73
data in transit, 341–343
E@H, 336–337
Microsoft Data Encryption SDK, 324–329
SSE, 335–336
Data Encryption SDK (MS), 324–329
Data Flow Diagrams (DFD), threat modeling, 82–85
data in transit, encryption, 29, 341–343
data integrity failures, 266–269
data plane, 392, 393
auditing, 404, 419–420
authentication, 402–403, 414
authorization, 403–404, 415–418
cryptography, 405, 420
network isolation, 405–406, 420–421
permissions, 161
data protection
GDPR, 216–217
governance, 203–204
data recovery, governance, 205
data verification, 242
determining correctness, 243
error handling, 253
Greek question mark, 251–252
high-level validation tools, 252
HTML, 251
namespace std, 245–246
real-world experience, 243–245
security decisions based on names, 253
validating comment fields, 249–250
validating date and time, 247
validating open spots, 248–249
validating vaccination center ID, 246
validating vaccination type, 248
encoding data, 256
rejecting known bad data, determining correctness, 253–255
database security
Always Encrypted, 429–433
auditing, Cosmos DB Security, 419–420
Azure Security Baseline, 411
Azure SQL Database, 394–395
Azure SQL Ledger, 409–410
Azure SQL Managed Instance, 395
control (management) plane, 392, 393

auditing, 399–400
authentication, 396–398
authorization, 398–399
cryptography, 400–401
network isolation, 401–402
Cosmos DB Security, 411–412
auditing, 414
authentication, 412–413, 414–415
authorization, 413, 415–418
Azure Security Baseline, 422
cryptography, 414, 421
data plane authentication, 414–415
Microsoft Defender for Cosmos DB, 421–422
network isolation, 414, 420–421
crypto shredding, 407
cryptographic controls, 393
data at rest, cryptography, 406–407
data discovery/classification, 409
data plane, 392, 393
auditing, 404
authentication, 402–403
authorization, 403–404
cryptography, 405, 421
network isolation, 405–406
dynamic data masking, 408
encryption, Always Encrypted, 422–426
golden rules, 394
immutable storage, 410–411
importance of, 391
managed identities, 407–408
Microsoft Defender for Cosmos DB, 421–422
NIN, 426–429
remote attestation, 431
secure enclaves, 431
security services, 393–394
SQL Injection, 433–434
SQL Server, 394, 395
auditing, 395
authentication, 395
authorization, 395
cryptography, 395–396
network isolation, 396
SQL Server family (overview), 394
SQL Server IaaS Agent, 411
SSN, 426–429
supported products, 392
TDE, 406–407
techniques (overview), 392
data-flow analysis, 18
date and time validation, data verification, 247
DCsv3 VM (Virtual Machines), 363–364
debt, technical security, 19
debugging
OAuth2, 132–133
OpenID Connect, 132–133
security headers, 264
TLS, 354–356
dedicated hosts, 27, 41
Dedicated HSM, Azure Key Vault, 309
dedicated PaaS instances, 456
Defender, Microsoft, Azure Key Vault, 306
Defender for Cloud (DFC), Microsoft, 207, 208
Defender for Containers, Microsoft, 386
Defender for Cosmos DB, Microsoft, 421–422
Defender for SQL, Microsoft, 410–411
defense in depth, 34–37
defining
bug bars, 7–10
firewall rules, 59
mitigations, 110–112
private endpoints, 60
roles, custom definitions, 165–166
toolchains, 11
DEK, 329, 338
delegated permissions, 138
delimiters, cryptographic agility, 323
Demilitarized Zones (DMZ), 34–35
Denial-of-Service (DOS) attacks, 75–77, 391
denying role assignments, 167
deployments
agents, 440–441
CI/CD
defined, 435
deployment agents, 440–441
developers, specialized security, 436–437
main branch (trunk) security, 438–439

PR approvals, 437
PROD deployments, 439–440
SCA tools, 436
secrets and service connections, 438
separation of duties, 437–438
source control systems, 436
supply chain attacks, 436
tools (overview), 435–436
trunk (main branch) security, 438–439
PROD, 439–440
design patterns, 51. *See also* security patterns
design security
access rights, 28–29
AD Access Reviews, 42
AD PIM, 46
assume-breach, 29
attack surface reduction, 32–33
Azure security design principles, 32
cloud computing, 23–24, 28–29
complete mediation, 33–34
Customer Lockbox, 25–26
Dedicated Hosts, 41
dedicated hosts, 27
defense in depth, 34–37
DevOps, 23–24
DMZ, 34–35
economy of mechanisms, 37–38
encrypting data in transit, 29
fail-safe defaults, 38–40
FAIR, 31–32
firewalls, 27
IaaS, 24–27
insecure design, 259
least common mechanism, 40–41
leveraging existing components, 42–43
micro-segmenting networks, 29
monitoring, 29
open design, 43–45
open source, 44–45
PaaS, 24–27
PAW, 42
PIM, 28
policy-based authorization, 28
principle of least privilege, 28, 41–42
prioritizing mitigations, 48
psychological acceptability, 45
residual risk, 37
SaaS, 24–27
scanning, 29
separation of duties, 45–46
Shared Responsibility Model, 24–25
shifting left, 31
single point of failure, 46–47
SQLMap, 30
thoughts on, 31–32
WAF, 30
weakest link, 47–48
zero trust, 27–31
determining correctness, verifying data, 243
error handling, 253
Greek question mark, 251–252
HTML, 251
names, security decisions based on, 253
namespace std, 245–246
real-world experience, 243–245
validating
comment fields, 249–250
date and time, 247
high-level validation tools, 252
open spots, 248–249
vaccination center ID, 246
vaccination type, 248
deterministic encryption, 73
DEV environments, 448
developers
authentication, importance of, 155
CI/CD, 436–437
compromises, 3–4
fuzz testing, 274
governance, 201
specialized security, 436–437
zero trust, 27–31
development, logging for, 172
device code flows, 136
DEVINT environments, 448
DevOps

ADO SHA, 456
deployment agents, 440–441
governance, 205
identity, 438
intentional security monitoring/auditing, 190
main branch (trunk) security, 438–439
PROD deployments, 439–440
secure design, 23–24
service connections, 438
trunk (main branch) security, 438–439
DFC (Defender for Cloud), Microsoft, 207, 208
DFD (Data Flow Diagrams), threat modeling, 82–85
disclosure attacks, information, 391
discovery/classification, data, 409
Disk Encryption, Azure, 336
DMZ (Demilitarized Zones), 34–35
DNS (Domain Name Systems), 454–455
CNAME, 459–460
dangling DNS, 459–460
Docker, containers, 373–374
documentation
governance, 206
Office (MS) documents, defined, 317
Domain Name Systems (DNS), 454–455
CNAME, 459–460
dangling DNS, 459–460
domain security, 309–311
DoS (Denial-of-Service) attacks, 75–77, 391
DTD bomb attacks, 13
duties, separation of, 45–46, 437–438
dynamic analysis tools, 16–17
dynamic data masking, 408

E

E@H (Encryption at Host), 336–337
ECDSA (Elliptic Curve Digital Signature Algorithm), 353
economy of mechanisms, 37–38
education (security), governance, 206
egress/ingress controls, 449, 457
elevation of privileges attacks, 391
elliptic-curve keys, 301
enclaves, secure, 431
encoding data, verifying data, 256
encryption
Always Encrypted, 422–426, 429–433
client-side data encryption, 71–73
column encryption, 408
data encryption
Azure Disk Encryption, 336
data in transit, 29, 341–343
E@H, 336–337
Microsoft Data Encryption SDK, 324–329
SSE, 335–336
database security, Always Encrypted, 422–426
deterministic encryption, 73
E@H, 336–337
Intel TME-MK, 363–364
KEK, 329, 338
memory, 360
operations, Azure Key Vault, 290–299
SQL Server Always Encrypted, 73
SSE
CMK, 330
PMK, 329–330
TDE, 406–407
Encryption at Host (E@H), 336–337
endpoints, 205
defining, 60
private endpoints, 454–455, 457
Entra, 130
EPAC (Enterprise Policy as Code), 212
error handling, determining correctness by verifying data, 253
Event Hub, Azure Monitor, 174
events
custom events, 193–199
threat modeling, m 190–199
existing data, mutating, 277–280

F

Fahmy, Sonia, firewalls, 27
fail-safe defaults, 38–40

failure, single point of failure, 46–47
FAIR, 31–32
Fair Institute, 32
FedRAMP (Federal Risk and Authorization
  Management Program), 218–219, 221
FIPS 140 (Federal Information Processing
  Standard 140), 221–222
  Azure Key Vault and managed HSM, 224
  SHA-2 in .NET, 222–223
firewalls, 27
  Azure Firewall, 449–450
  Azure Firewall Premium SKU, 450
  rules, defining, 59
  WAF, 30
flexibility, threat modeling, 91
flows
  analysis, 17
    control-flow analysis, 17–18
    data-flow analysis, 18
  OAuth2, 127, 134
    authorization code flows, 134–135
    client credential flows, 135
    device code flows, 136
    implicit flows, 136
    OBO flows, 135
    ROPC flows, 136
    supported applications, 136–137
formatting data, manipulating by
  format, 280
Forrester and Gartner, threat
  modeling, 89
fragmentation, containers, 383
frameworks, Azure Well-Architected
  Framework, 53
Frantzen, Michael, firewalls, 27
Function Cointainers, Azure, 377
functionality, banned, 12–13
fuzz testing, 274, 275–276
  API, 280–283
  Azure rules of engagement, 275
  generating random data, 276–277
  manipulating data by format, 280
  mutating existing data, 277–280

G

Gamma, Erich, design patterns, 51
gate tools, quality, 20
GDPR (General Data Protection Regulation),
  216–217
Geo-Redundant Storage (GRS), 47
Geo-Zone-Redundant Storage (GZRS), 47
get operations, 291–299
GitHub
  Azure Policy, 189
  deployment agents, 441
  main branch (trunk) security, 438–439
  PROD deployments, 439–440
  public repos, 441
  service principal secrets, 438
  trunk (main branch) security, 438–439
golden rules, 394
governance
  ASB, 202
  asset management, 204
  automating deployments, 206
  Azure Initiatives, 210
  Azure Policy, 209
    assigning policies, 211–212
    effects, 210
    effects, enforcing by environment, 210–211
    policy as code, 212
  Azure Portal, 208–209
  backups, 205
  compliance state, reviewing, 208–209
  data protection, 203–204
  data recovery, 205
  developers, 201
  DevOps, 205
  documentation, 206
  endpoint security, 205
  enforcement, 206
  identity management, 203
  incident response plans, 204
  logging, 204
  Microsoft DFC, 207, 208
  network security, 202

posture management, 204–205
privileged access, 203
RBAC, 206, 210–211
Secure Score, 207–208
security education, 206
strategies, 205
vulnerability management, 204–205
grant flows. *See* flows
Greek question mark, determining correctness by verifying data, 251–252
grouping actions, Log Analytics workspaces, 185–186
groups, role assignments, 58–59
GRS (Geo-Redundant Storage), 47
GUID, version numbers, 324
guidance, threat modeling, 91
GZRS (Geo-Zone-Redundant Storage), 47

H

hacks, reasons for, 171–172
hardware root of trust, 359
hashes in coding, 341
Health Information Trust (HITRUST), 216
Health Insurance Portability and Accountability Act (HIPPA), 215–216
Helm, Richard, design patterns, 51
HIPPA (Health Insurance Portability and Accountability Act), 215–216
HITRUST (Health Information Trust), 216
honey keys, 303
HoneyPi, 171
honeypots, 171
hosting, applications, 133
hosts
dedicated hosts, 27
E@H, 336–337
HSM (Hierarchical Storage Management)
Azure Key Vault
Dedicated HSM, 309
FIPS 140, 224
Managed HSM, 308–311
Payment HSM, 309
security domains, 309–311
HTML (Hypertext Markup Language), determining correctness, verifying data, 251
HTTP (Hypertext Transfer Protocol), AppServiceHTTPLogs, 173
hubs, 447
Hyper-V, bounty programs, 40–41

I

IaaS (Infrastructure as a Service), 24–27
containers, 376
SQL Server IaaS Agent, 411
TLS, 350
IaC (Infrastructure as Code), 443
ID tokens, 138
identification failures, 264–266
identifying threats, 108–109, 113–115
identity, 123–124
of attackers, secure coding, 240–241
centralized identity providers, authentication, 54–55
compound identities, 299
defined, 124
DevOps, 438
Entra, 130
isolating identity perimeters, 60–61
managing
governance, 203
identities, 64–66, 403, 407–408
modern identity, access management, 125
NIN, 426–429
OAuth2, 125–129, 134, 146–150
application registration, 129
authorization servers, 134
clients, 134, 137–138
consent, 139–140
debugging, 132–133
flows, 127, 134–137

JWT, 142–146
permissions, 138–140
resource owners, 127
resource servers, 134
roles, 134
scopes, 141
tokens, 138
tokens as credentials, 128
users vs. clients, 131
validation, 143–146
OpenID Connect, 125–129, 132–133
PIM, 28, 168
providers, 134
SSN, 426–429
images, containers, 385
immaturity, containers, 383
immutable storage, 410–411
implicit flows, 136
incident response plans, 18, 204
incremental consent, 140
information
disclosure attacks, 391
sensitive information, management patterns, 69
Infrastructure as a Service (IaaS)
containers, 376
SQL Server IaaS Agent, 411
TLS, 350
Infrastructure as Code (IaC), 443
ingress/egress controls, 449, 457
Initiatives, Azure, 210
injection, 258
inline scripts/styles, 260–263
input, *all input is evil*, 238–239
insecure coding, 237
insecure design, 259
integration
CI/CD
defined, 435
deployment agents, 440–441
developers, specialized security, 436–437
main branch (trunk) security, 438–439
PR approvals, 437
PROD deployments, 439–440
SCA tools, 436
secrets and service connections, 438
separation of duties, 437–438
source control systems, 436
supply chain attacks, 436
tools (overview), 435–436
trunk (main branch) security, 438–439
threat modeling, 91
Intel
SGX, 361–362, 363–364
TME-MK, 363–364
intentional security monitoring/auditing, 190
interception, likelihood of, 70–71
International Organization for Standardization/International Electrotechnical Commission (ISO/IEC)
ISO/IEC 27001, 225–226
ISO/IEC 27034, 226
Internet
CIS Benchmarks, 80–81
isolating networks, 59–60
Internet Protocol version 6 (IPv6), 445
Investment (ROI), Return on, threat modeling, 88
IPv4 (Internet Protocol version 4), 445
addresses, 445–446
concepts, 445
routing, 446
UDR, 446
IPv6 (Internet Protocol version 6), 445
ISO/IEC (International Organization for Standardization/International Electrotechnical Commission)
ISO/IEC 27001, 225–226
ISO/IEC 27034, 226
isolating
identity perimeters, 60–61
memory, 360
networks, 59–60, 420–421
networks
Azure Key Vault, 304–305
control (management) plane, 401–402, 414
data plane, 405–406

J

Johnson, Ralph, design patterns, 51
just-in-time administration, 56–58
JWT (JSON Web Tokens), 142–146

K

K8s (Kubernetes), 373–375
Kamara, Seny, firewalls, 27
KEK (Key Encryption Keys), 329, 338
Kerckhoff's Principle, 44
Kerschbaum, Florian, firewalls, 27
Key Vault, Azure, 66–69, 76, 288
  access control, 288–290, 301
  auditing, 301–303
  backups, 306–307
  break-glass accounts, 298
  "bring your own key" strategy, 301
  certificates, 288
  compound identities, 299
  contributor permissions, 297
  customer-managed keys, 306
  elliptic-curve keys, 301
  encrypting operations, 290
  FIPS 140 and managed HSM, 224
  get operations, 291–299
  honey keys, 303
  HSM
    Dedicated HSM, 309
    Managed HSM, 308–311
    Payment HSM, 309
  keys, 288, 301
  logging, 301–303
  Microsoft Defender, 306
  network isolation, 304–305
  permission models
    selecting, 295
    switching, 292
  Premium edition, 299–301
  RBAC, 291–299
  restoring key versions, 307–308
  rotating keys, 339–341
  RSA, 301
  secrets, 288
  secure keys, 312
  Standard edition, 299–300
  templates, 300
  verifying operations, 290
  wrapping operations, 290
keys
  agreements, 344
  authentication keys, 338
  Azure Key Vault. *See* separate entry
  Azure Storage Keys, 107
  breaking, cost of, 350–351
  "bring your own key" strategy, 301
  BYOK, 73–74
  certificates, 288
  CMK, 186–187, 204, 330
  cryptography, defined, 287
  customer-managed keys, 306
  DEK, 329, 338
  elliptic-curve keys, 301
  exchanges, 344
  honey keys, 303
  Intel SGX, 364
  Intel TME-MK, 363–364
  KEK, 329, 338
  PKCE, 135
  PMK, SSE, 329–330
  restoring key versions, 307–308
  rotating, 337–341
  secrets, 288
  secure keys, 287, 312, 360
  signing keys, 338
  storage account keys, misuse of, 158–159
known bad data, rejecting, 253–255
KQL (Kusto Query Language) queries, 176–180
Kubernetes
  AKS, 107, 378–380, 457
  K8s, 373–375
Kuriel, Maor, Kubernetes (AKS), 107
Kusto Query Language (KQL) queries, 176–180

L

landing zones, 447
launches, trusted, 360, 366–367
least common mechanism, 40–41
least privilege, principle of, 28, 41–42
LeBlanc, David, *Writing Secure Code*, 6
leveraging existing components, 42–43
likelihood of interception, 70–71
limiting access rights, 28–29
links, private, 454–455
Linux VM (Virtual Machines), TLS, 350–351
Litchfield, David, determining correctness, 243
Local Redundant Storage (LRS), 47
Log Analytics workspaces, Azure Monitor, 176
    action groups, 185–186
    alerts, raising, 181–185
    audit logs, protecting, 186–188
    KQL queries, 176–180
logging
    allLogs, 175
    AppServiceAppLogs, 173
    AppServiceAuditLogs, 173
    AppServiceConsoleLogs, 173
    AppServiceHTTPLogs, 173
    AppServiceIPSecAuditLogs, 173–174
    AppServicePlatformLogs, 174
    audit logs, 175, 186–188
    for auditing, defined, 172
    Azure Key Vault, 301–303
    Azure Monitor
        Azure Storage, 174, 187–188
        diagnostic settings, 172–175
        Event Hub, 174
        Log Analytics workspaces, 174, 176
        Log Analytics workspaces, action groups, 185–186
        Log Analytics workspaces, KQL queries, 176–180
        Log Analytics workspaces, protecting audit logs, 186–188
        Log Analytics workspaces, raising alerts, 181–185
        Partner Solution, 174
    Azure Policy, adding audit logs, 189
    Azure Sentinel, 186
    category groups, 175
    CMK, 186–187
    costs, 189–190
    crypto-shredding, 186
    defined, 172
    for development, defined, 172
    failures, 269
    governance, 204
    intentional security monitoring/auditing, 190
    KQL queries, 176–180
    monitoring, defined, 172
    threat detection, 204
    threat modeling events, 190–199
login credentials, storage, SQL Server, 398
LRS (Local Redundant Storage), 47

M

main branch (trunk) security, 438–439
malware, TEE code, 362
Managed HSM
    Azure Key Vault, 308–311
    security domains, 309–311
managed identities, 64–66, 403, 407–408
Managed OpenShift, 381
management (control) plane, 392, 393
    auditing, 399–400, 414
    authentication, 396–398, 412–413
    authorization, 398–399, 413
    cryptography, 400–401
    network isolation, 401–402, 414
managing
    access, modern identity, 125
    assets, 204
    identities, 64–66
        governance, 203
        PIM, 28
    PIM, 28, 168
    posture, 204–205
    resource management private links, 401–402
    risk, threat modeling, 101
    role assignments, 164–165

secrets management patterns, 64
secure key management, 360
sensitive information, 69
Threats Manager Studio, 99–101
dashboard, 118–119
roadmaps, 115–118
VNets, 456
vulnerabilities, 204–205
manipulating data by format, 280
masking dynamic data, 408
mechanisms
economy of mechanisms, 37–38
least common mechanism, 40–41
mediation, complete, 33–34
memory
Intel TME-MK, 363–364
isolation/encryption, 360
metadata, cryptographic, 316–317
MFA (Multifactor Authentication), 152–153
micro-segmenting networks, 29
microservices, Key Vault, 68–69
Microsoft
Data Encryption SDK, 324–329
Defender, Azure Key Vault, 306
Defender for Containers, 386
Defender for Cosmos DB, 421–422
Defender for SQL, 410–411
DfrC, 207, 208
Entra, 130
MSAL, 130–132
ADAL, 131
CAE, 131
debugging tools, 132–133
Office (MS) documents, defined, 317
SDL
attack surface analysis, 10
banned functionality, avoiding, 12–13
bug bars, defining, 7–10
components, 5–6
CVSS, 7
defined, 4
DTD bomb attacks, 13
dynamic analysis tools, 16–17
features, 5
flow analysis, 17–18
goals, 4
incident response plans, 18
overview, 20–21
penetration testing (pentests), 19
requirements, 4, 5–6
security training, 6
static analysis tools, 13–16, 17
tasks, 5–6
tasks by sprint, 20–21
technical security debt, 19
threat modeling, 10
toolchains, defining, 11
Threat Modeling Tool, 94–95
misconfigured security, 259–260
mitigation
assumptions, 113
backlogging, 119–122
creating, 111
defining, 110–112
mitigation identification phase, threat modeling, 82
Mitigations Kanbnan, 122
prioritizing, 48
threat modeling, 91
MITRE, 229–230
modeling threats, 10, 32, 79. *See also* attacks
analysis phase, 82
authentication, 107
automation, 91
benchmarks
Azure Security Benchmark, 80, 81
CIS Benchmarks, 80–81
CAIRIS, 93–94
compliance, 233–234
defined, 80–81
development vs. security, 90–91
DFD, 82–85
events, 190–199
example of, 101–102
first meeting, 102–104
second meeting, 104–107
factors of, 92–93
flexibility, 91

guidance, 91
identifying threats, 108–109
automation, 113–115
severity of threats, 110
integration, 91
Microsoft Threat Modeling Tool, 94–95
mitigation, 91
assumptions, 113
backlogging, 119–122
defining, 110–112
mitigation identification phase, 82
OWASP Threat Dragon, 96–97
phases of, 81–84
pytm, 97–98
risk management, 101
roadmaps, 115–118
ROI, 88
searching for better processes, 88–89
secure coding, 239–242
security champions, 88–89
STRIDE threat-classification, 85–86
Threagile, 92–93, 98–99
threat identification phase, 82
Threat Modeling Manifesto, 80
Threats Manager Studio, 99–101
dashboard, 118–119
roadmaps, 115–118
tools (overview), 91–92
trouble with, 86–88
validation phase, 82
modern identity, access management, 125
monitoring
allLogs, 175
AppServiceAppLogs, 173
AppServiceAuditLogs, 173
AppServiceConsoleLogs, 173
AppServiceHTTPLogs, 173
AppServiceIPSecAuditLogs, 173–174
AppServicePlatformLogs, 174
audit logs, 175, 186–188
Azure Monitor
Azure Storage, 174, 187–188
diagnostic settings, 172–175
Event Hub, 174
Log Analytics workspaces, 174, 176
Log Analytics workspaces, action groups, 185–186
Log Analytics workspaces, KQL queries, 176–180
Log Analytics workspaces, protecting audit logs, 186–188
Log Analytics workspaces, raising alerts, 181–185
Partner Solution, 174
Azure Policy, adding audit logs, 189
Azure Sentinel, 186
category groups, 175
CMK, 186–187
costs, 189–190
crypto-shredding, 186
defined, 172
failures, 269
intentional security monitoring/auditing, 190
KQL queries, 176–180
secure design, 29
threat modeling events, 190–199
MSAL (Microsoft Authentication Library), 130–132
ADAL, 131
CAE, 131
debugging tools, 132–133
Mueller III, Robert S., hacks, 172
Multifactor Authentication (MFA), 152–153
mutating existing data, 277–280

N

names, security decisions based on, 253
namespace std, 245–246
National Identity Numbers (NIN), 426–429
National Institute of Standards and Technology (NIST)
Cybersecurity Framework, 220–221
NIST SP 800–53, 219–220, 221
National Vulnerability Database (NVD), 230
.NET code, TLS, common mistakes, 354

.NET SHA-2 and FIPS 140, 222–223
Network Security Groups (NSG), 446–447
networks
  ADO SHA, 456
  agents, 456–458
  AKS, 457
  Azure networking primer, 443–445
  cluster network policies, 457–458
  control (management) plane isolation, 401–402, 414
  data plane isolation, 405–406
  firewalls, Azure Firewall, 449–450
  isolation, 59–60, 396
    Azure Key Vault, 304–305
    Cosmos DB Security, 420–421
  micro-segmenting networks, 29
  private networking, PaaS, 451–452
  security, governance, 202
  segmenting
    networks, 29
    VNets, 447–448
  VNets, 443–445
    ADO SHA, 456
    agents, 456–458
    AKS, 457
    allowed communications, 448
    API Management Gateways, 451
    ASG, 447
    Azure Application Proxy, 451
    Azure Firewall, 449–450
    Azure Firewall Premium SKU, 450
    Azure WAF, 450–451
    cluster network policies, 457–458
    CNAME, 459–460
    dedicated PaaS instances, 456
    DEV environments, 448
    DEVINT environments, 448
    DNS, 454–455
    hubs, 447
    ingress/egress controls, 449, 457
    IPv4, 445–446
    IPv6, 445
    landing zones, 447
    managing, 456
    NONPROD environments, 448
    NSG, 446–447
    NVA, 449
    PaaS and private networking, 451–452
    private endpoints, 454–455, 457
    private shared PaaS, 452–455
    PROD environments, 448
    SANDBOX environments, 448
    segmenting, 447–448
    shared app service plans, 455
    spokes, 447
NIN (National Identity Numbers), 426–429
NIST (National Institute of Standards and Technology)
  Cybersecurity Framework, 220–221
  NIST SP 800–53, 219–220, 221
nodes, AKS, 387
nonclass factory cryptographic algorithms, 317–320
NONPROD environments, 448
NSG (Network Security Groups), 446–447
NVA (Network Virtual Appliances), 449
NVD (National Vulnerability Database), 230

O

OAuth2, 125–129, 146–150
  applications, registration, 129
  authorization servers, 134
  clients, 134, 137
    confidential clients, 137–138
    public clients, 137
  consent, 139–140
  debugging, 132–133
  flows, 127, 134
    authorization code flows, 134–135
    client credential flows, 135
    device code flows, 136
    implicit flows, 136
    OBO flows, 135
    ROPC flows, 136
    supported applications, 136–137

permissions, 138–140
application permissions, 138
delegated permissions, 138
resource owners, 127, 134
resource servers, 134
roles, 134
scopes, 141
tokens
access tokens, 138
as credentials, 128
ID tokens, 138
JWT, 142–146
validation, 143–146
users vs. clients, 131
OBO (On-Behalf-Of) flows, 135
Office (MS) documents, defined, 317
On-Behalf-Of (OBO) flows, 135
open design, 43–45
open source, 44–45
Open Web Application Security Project (OWASP), 229
OpenID Connect, 125–129, 132–133
OpenShift, Managed, 381
outdated components, 264
OWASP (Open Web Application Security Project), 229
static analysis tools, 16
Threat Dragon, 96–97
XSS, 30
owners, resource, 127, 134

P

PaaS (Platform as a Service), 24–27
dedicated PaaS instances, 456
private networking, 451–452
private shared PaaS, 452–455
TLS, 345–346
PAM (Privileged Access Management) tool, 56–57
Partner Solution, Azure Monitor, 174
passwordless authentication, 152
passwords
breaking, cost of, 350–351
ZKPP, 151
patterns
availability patterns, 74–75
design patterns, 51
security patterns
application roles, 62–64
authentication patterns, 53–55
authorization patterns, 56
availability patterns, 74–75
Azure, 51–52
Azure AD PIM, 57
Azure Well-Architected Framework, 53
BYOK, 73–74
client-side data encryption, 71–73
DoS attacks, 75–77
isolating identity perimeters, 60–61
isolating networks, 59–60
just-in-time administration, 56–58
Key Vault, 66–69, 76
likelihood of interception, 70–71
list of, 52
managed identities, 64–66
PAM tool, 56–57
RBAC, 61–62
role assignments, 58–59
secrets management patterns, 64
secure channels, 69–70
sensitive information management patterns, 69
Patterns and Practices Initiative, 42–43
PAW (Privileged Access Workstations), 42
Payment Card Industry Data Security Standard (PCI DSS), 217–218
Payment HSM, 309
PCI DSS (Payment Card Industry Data Security Standard), 217–218
penetration testing (pentests), 19
Perfect Forward Secrecy (PFS), 344–345
perimeter defenses, firewalls, 27
permissions
contributor permissions, 297
control plane permissions, 161

data plane permissions, 161
OAuth2, 138–140
application permissions, 138
delegated permissions, 138
permission models
selecting, 295
switching, 292
PFS (Perfect Forward Secrecy), 344–345
PIM (Privileged Identity Management), 28, 168
PKCE (Proof of Key for Code Exchange), 135
planning, incident response plans, 18
Platform as a Service (PaaS), 24–27
dedicated PaaS instances, 456
private networking, 451–452
private shared PaaS, 452–455
TLS, 345–346
PMK (Platform-Managed Keys), SSE, 329–330
pods, containers, 387–388
Policy, Azure, 209
assigning policies, 211–212
audit logs, adding, 189
effects, 210–211
policy as code, 212
policy-based authorization, 28
Ponemon Institute, 27, 31
Portal, Azure, compliance state governance, 208–209
posture management, 204–205
PR (Pull Requests), approvals, 437
primary keys, data plane authorization, 415–416
principle of least privilege, 28, 41–42
prioritizing mitigations, 48
private endpoints, 60, 454–455, 457
private links, 454–455
private networking, PaaS, 451–452
private shared PaaS, 452–455
privileged access, 203
Privileged Access Management (PAM) tool, 56–57
Privileged Access Workstations (PAW), 42
Privileged Identity Management (PIM), 28, 168
privileges
elevation of privileges attacks, 391
principle of least privilege, 28
processors, confidential computing, 361
PROD
deployments, 439–440
environments, 448
Proof of Key for Code Exchange (PKCE), 135
psychological acceptability, 45
public clients, 137
public repos, GitHub, 441
Pull Requests (PR), approvals, 437
pytm, threat modeling, 97–98

Q

quality gate tools, 20
quality versus security, 4–5
queries, KQL, 176–180
question mark (Greek), determining correctness by verifying data, 251–252

R

random data, fuzz testing, 276–277
RBAC (Role-Based Access Control), 61–62, 163–164, 206, 210–211, 291–299, 416–419
Reagan, U.S. President Ronald, verifying data, 242
reconnaissance attacks, 171
recovery, governance, 205
reducing attack surfaces, 32–33
redundancy levels, Azure storage, 47
Reed, Brian, threat modeling, 89
refresh tokens, 138
registration, OAuth2 applications, 129
rejecting known bad data, 253–255
remote attestation, 360, 364–366, 431
repudiation attacks, 391
requests, SSRF, 269–270
residual risk, 37
resource management private links, 401–402
Resource Owner Password Credentials (ROPC) flows, 136
resource owners, 127, 134
resource servers, 134

resource tokens, data plane authorization, Cosmos DB Security, 416
rest, data at, 406–407
REST API security, 282–283
restoring, key versions, 307–308
Return on Investment (ROI), threat modeling, 88
reviewing compliance state, governance, 208–209
revocation checking, 266
Richer, Justin, OAuth2, 129
risk
  acceptance, PCI DSS, 218
  assessments, bug bars, defining, 7–10
  managing, threat modeling, 101
  programs/compliance, 232–233
    ASB, 227–228
    ATT&CK, 231
    CAPEC, 231
    CIS benchmarks, 226–227
    CVE, 230
    CWE, 231
    defined, 213–215
    FedRAMP, 218–219, 221
    FIPS 140, 221–224
    GDPR, 216–217
    HIPPA, 215–216
    HITRUST, 216
    ISO/IEC 27001, 225–226
    ISO/IEC 27034, 226
    MITRE, 229–230
    NIST Cybersecurity Framework, 220–221
    NIST SP 800–53, 219–220, 221
    NVD, 230
    OWASP, 229
    PCI DSS, 217–218
    SOC, 224–225
    threat modeling, 233–234
  residual risk, 37
RiskLens, 32
roadmaps, 115–118
ROI (Return on Investment), threat modeling, 88
Role-Based Access Control (RBAC), 61–62, 163–164, 206, 210–211, 291–299, 416–419
roles
  AD roles, 162–164
  applications, 62–64
  assigning
    best practices, 167–168
    Blueprints, 167
    denying assignments, 167
    managing assignments, 164–165
  Blueprints, 167
  custom definitions, 165–166
  group assignments, 58–59
  OAuth2, 134
root of trust, hardware, 359
ROPC (Resource Owner Password Credentials) flows, 136
rotating keys, 337–341
routing
  IPv4, 446
  UDR, 446
RSA algorithm, 301, 353
rules
  Azure rules of engagement, 275
  firewalls, defining, 59

S

SaaS (Software as a Service), 24–27
SAFECode, threat modeling, 89
Saltzer, Jerome
  Azure security design principles, 32
  complete mediation, 33–34
  economy of mechanisms, 37–38
  fail-safe defaults, 38–40
  leveraging existing components, 42–43
  open design, 43–44
  psychological acceptability, 45
SAML (Security Assertion Markup Language), 129
SANDBOX environments, 448
Sanso, Antonio, OAuth2, 129
SAS (Shared Access Signature) tokens, 158
saving KQL queries, 180

SCA (Software Component Analysis) tools, CI/CD, 436
scanning, secure design, 29
Schoenfeld, Brook S. E., threat modeling, 81
Schroeder, Michael
  Azure security design principles, 32
  complete mediation, 33–34
  economy of mechanisms, 37–38
  fail-safe defaults, 38–40
  leveraging existing components, 42–43
  open design, 43–44
  psychological acceptability, 45
Schultz, Eugene, firewalls, 27
scopes
  AD scopes, 162
  OAuth2, 141
scoring vulnerabilities (CVSS), 7
scripting
  inline scripts/styles, 260–263
  XSS, 30
SDL (Security Development Lifestyle)
  attack surface analysis, 10
  banned functionality, avoiding, 12–13
  bug bars, defining, 7–10
  components, 5–6
  CVSS, 7
  defined, 4
  DTD bomb attacks, 13
  dynamic analysis tools, 16–17
  features, 5
  flow analysis, 17
    control-flow analysis, 17–18
    data-flow analysis, 18
  goals, 4
  incident response plans, 18
  overview, 20–21
  penetration testing (pentests), 19
  requirements, 4, 5–6
  security training, 6
  static analysis tools, 13–16, 17
  tasks, 5–6
  tasks by sprint, 20–21
  technical security debt, 19
  threat modeling, 10
  toolchains, defining, 11
SecOps (Security Ops), 125
secrets, keys, cryptography, 288
secrets and service connections, CI/CD, 438
secrets management patterns, 64
secure channels, 69–70
secure coding
  all input is evil, 238–239
  attackers
    identity of, 240–241
    what attackers control, 241–242
  C programming language, 270–271
  C++ programming language, 270–271
  fuzz testing, 274, 275–276
    API, 280–283
    Azure rules of engagement, 275
    generating random data, 276–277
    manipulating data by format, 280
    mutating existing data, 277–280
  hashes in coding, 341
  insecure coding, 237
  reviewing, 273–274
  revocation checking, 266
  threat modeling, 239–242
  use analysis tools, 271–273
  verifying data, 242
    determining correctness, 243–253
    encoding data, 256
    rejecting known bad data, 253–255
  vulnerabilities, 256–257
    authentication failures, 264–266
    broken access control, 257
    checking signatures, 268
    cryptography failures, 257–258
    data integrity failures, 266–269
    debugging security headers, 264
    identification failures, 264–266
    injection, 258
    inline scripts/styles, 260–263
    insecure design, 259
    logging failures, 269
    misconfigured security, 259–260

monitoring failures, 269
signing binaries, 268–269
software failures, 266–269
SSRF, 269–270
vulnerable/outdated components, 264
secure design
access rights, 28–29
AD Access Reviews, 42
AD PIM, 46
assume-breach, 29
attack surface reduction, 32–33
Azure security design principles, 32
cloud computing, 23–24, 28–29
complete mediation, 33–34
Customer Lockbox, 25–26
Dedicated Hosts, 41
dedicated hosts, 27
defense in depth, 34–37
DevOps, 23–24
DMZ, 34–35
economy of mechanisms, 37–38
encrypting data in transit, 29
fail-safe defaults, 38–40
FAIR, 31–32
firewalls, 27
IaaS, 24–27
least common mechanism, 40–41
leveraging existing components, 42–43
micro-segmenting networks, 29
monitoring, 29
open design, 43–45
open source, 44–45
PaaS, 24–27
PAW, 42
PIM, 28
policy-based authorization, 28
principle of least privilege, 28, 41–42
prioritizing mitigations, 48
psychological acceptability, 45
residual risk, 37
SaaS, 24–27
scanning, 29
separation of duties, 45–46
Shared Responsibility Model, 24–25
shifting left, 31
single point of failure, 46–47
SQLMap, 30
thoughts on, 31–32
WAF, 30
weakest link, 47–48
zero trust, 27–31
secure enclaves, 431
Secure Encrypted Visualization-Secure Nested Paging (SEV-SNP), AMD, 362–363
secure keys, 287, 312, 360
Secure Score, 207–208
Secure Sockets Layer (SSL), 342
security
champions, 21, 88–89
quality versus, 4–5
technical security debt, 19
training, SDL, 6
Security Assertion Markup Language (SAML), 129
Security Baseline, Azure, 411
Security Benchmark, 80, 81
security champions, 21, 88–89
security controls, types of, 36
security domains, 309–311
security headers, debugging, 264
security misconfigurations, 259–260
security patterns. *See also* design patterns
application roles, 62–64
authentication patterns, 53–55
Azure AD, 54–55
centralized identity providers, 54–55
authorization patterns, 56
availability patterns, 74–75
Azure, 51–52
Azure AD PIM, 57
Azure Well-Architected Framework, 53
BYOK, 73–74
client-side data encryption, 71–73
DoS attacks, 75–77
isolating
identity perimeters, 60–61
networks, 59–60
just-in-time administration, 56–58
Key Vault, 66–69, 76

likelihood of interception, 70–71
list of, 52
managed identities, 64–66
PAM tool, 56–57
RBAC, 61–62
role assignments, 58–59
secrets management patterns, 64
secure channels, 69–70
sensitive information management patterns, 69
segmenting networks, 29, 447–448
selecting permission models, 295
sensitive information, management patterns, 69
Sentinel, Azure, 186, 197–199
separation of duties, 45–46, 437–438
servers
authentication, 154
authorization servers, OAuth2, 134
encryption
CMK, 330
PMK, 329–330
resource servers, 134
SQL Server
IaaS Agent, 411
SQL Server Always Encrypted, 73, 368
SSE, 335–336
Server-Side Encryption (SSE), 335–336
Server-Side Request Forgery (SSRF), 269–270
service plans, shared app, 455
services
AKS, 107
App Services, 47
Azure container services, 375–376
container service security
deployments, 384–385
development, 384–385
overview, 383–384
DoS attacks, security patterns, 75–77
IaaS, 24–27
containers, 376
TLS, 350
microservices, Key Vault, 68–69
PaaS, 24–27, 345–346
principal secrets, GitHub, 438
SaaS, 24–27
secrets and service connections, CI/CD, 438
severity of threats, identifying, 110
SEV-SNP (Secure Encrypted Visualization-Secure Nested Paging), AMD, 362–363
SGX (Software Guard Extensions), Intel, 361–362, 363–364
SHA-2, FIPS 140 and SHA-2 in .NET, 222–223
Shared Access Signature (SAS) tokens, 158
shared app service plans, 455
Shared Responsibility Model, 24–25
shifting left, 31
Shostack, Adam, threat modeling, 81
signatures, checking, 268
signing binaries, 268–269
signing keys, 338
single point of failure, 46–47
Single Sign-On (SSO) authentication, 156–157
SMS-based authentication, 151–152
SOC (System and Organization Controls), 224–225
Social Security Numbers (SSN), 426–429
software
failures, 266–269
Intel SGX, 361–362, 363–364
SaaS, 24–27
SCA tools, CI/CD, 436
Software Guard Extensions (SGX), Intel, 361–362
source control systems, 436
specialized security, 436–437
specifying, severity of threats, 110
spokes, 447
spoofing attacks, 391
SQL (Structured Query Language)
connection strings, 402
Microsoft Defender for SQL, 410–411
SQL Database, Azure, 394–395
SQL Injection, 433–434
SQL Ledger, Azure, 409–410
SQL Managed Instance, Azure, 395
SQL Server
database security, 394, 395
auditing, 395
authentication, 395

authorization, 395
Azure SQL Database, 394–395
Azure SQL Managed Instance, 395
cryptography, 395–396
network isolation, 396
IaaS Agent, 411
login credentials, storage, 398
SQL Server Always Encrypted, 73, 368
SQLi (SQL injection), 258
SQLMap, 30
SSE (Server-Side Encryption), 335–336
SSH (Secure Shell), unsecured, 357
SSL (Secure Sockets Layer), 342
SSN (Social Security Numbers), 426–429
SSO (Single Sign-On) authentication, 156–157
SSRF (Server-Side Request Forgery), 269–270
static analysis tools, 13–16, 17
storage
Azure Storage, 174
cryptography, 331–335
protecting audit logs, 187–188
redundancy levels, 47
GRS, 47
GZRS, 47
HSM, Azure Key Vault
Dedicated HSM, 309
FIPS 140, 224
Managed HSM, 308–311
Payment HSM, 309
immutable storage, 410–411
login credentials, SQL Server, 398
LRS, 47
ZRS, 47
storage account keys, misuse of, 158–159
Storage Keys, 107
STRIDE threat-classification, 85–86
Structured Query Language. *See* SQL
styles, inline, 260–263
supply chain attacks, 436
switching, permission models, 292
System and Organization Controls (SOC), 224–225

T

tampering attacks, 391
Tarandach, Izar
pytm, 97
threat modeling, 81
TDE (Transparent Data Encryption), 406–407
technical security debt, 19
TEE code, 361–362
templates, Azure Key Vault, 300
testing
DAST, 16–17
fuzz testing, 274, 275–276
API, 280–283
Azure rules of engagement, 275
generating random data, 276–277
manipulating data by format, 280
mutating existing data, 277–280
penetration testing (pentests), 19
TLS, 354
Threagile, 92–93, 98–99
threat modeling, 10, 32, 79. *See also* threats (separate entry)
analysis phase, 82
authentication, 107
automation, 91
benchmarks
Azure Security Benchmark, 80, 81
CIS Benchmarks, 80–81
CAIRIS, 93–94
compliance, 233–234
defined, 80–81
development vs. security, 90–91
DFD, 82–85
events, 190–199
example of, 101–102
first meeting, 102–104
second meeting, 104–107
factors of, 92–93
flexibility, 91
guidance, 91
identifying threats, 108–109
automation, 113–115
severity of threats, 110

integration, 91
Microsoft Threat Modeling Tool, 94–95
mitigation, 82, 91
assumptions, 113
backlogging, 119–122
defining, 110–112
OWASP Threat Dragon, 96–97
phases of, 81–84
pytm, 97–98
risk management, 101
roadmaps, 115–118
ROI, 88
searching for better processes, 88–89
secure coding, 239–242
security champions, 88–89
STRIDE threat-classification, 85–86
Threagile, 92–93, 98–99
threat identification phase, 82
Threat Modeling Manifesto, 80
Threats Manager Studio, 99–101, 115–119
tools (overview), 91–92
trouble with, 86–88
validation phase, 82
threats
detection, logging, 204
DoS attacks, 391
elevation of privileges attacks, 391
identification, 82, 108–109, 113–115
information disclosure attacks, 391
monitoring. *See* separate threat monitoring entry
repudiation attacks, 391
spoofing attacks, 391
tampering attacks, 391
Threats Manager Studio, 99–101
dashboard, 118–119
roadmaps, 115–118
time and date validation, data verification, 247
TLS (Transport Layer Security), 342
cryptographic agility, 343
debugging, 354–356
IaaS, 350
Linux VM, 350–351
.NET code, common mistakes, 354
PaaS, 345–346
testing, 354
TLS 1.3 ciphersuites, 345
versions of, 342–343
Windows VM, 352–353
TME-MK (Total Memory Encryption-Multi-Key), Intel, 363–364
tokens
OAuth2, 128
access tokens, 138
ID tokens, 138
JWT, 142–146
validation, 143–146
SAS tokens, 158
“Too Many Requests” messages, 76
toolchains, defining, 11
Total Memory Encryption-Multi-Key (TME-MK), Intel, 363–364
training, security, 6
Transparent Data Encryption (TDE), 406–407
Transport Layer Security (TLS), 342
cryptographic agility, 343
debugging, 354–356
IaaS, 350
Linux VM, 350–351
.NET code, common mistakes, 354
PaaS, 345–346
testing, 354
TLS 1.3 ciphersuites, 345
versions of, 342–343
Windows VM, 352–353
trunk (main branch) security, 438–439
trust
boundaries, DFD, 84–85
hardware root of trust, 359
trusted launches, 360, 366–367
TrustZone, Arm, 363

U

UDR (User-Defined Routing), 446
unsecured SSH, 357
use analysis tools, 271–273
UUID, version numbers, 324

V

validation
  determining correctness, verifying data
    high-level validation tools, 252
    security decisions based on names, 253
    validating comment fields, 249–250
    validating date and time, 247
    validating open spots, 248–249
    validating vaccination center ID, 246
    validating vaccination type, 248
  tokens, 143–146
  validation phase, threat modeling, 82
variances, PCI DSS, 218
VBS (Virtualization-Based Security), 368
verifying data, 242
  code verifiers, 135
  determining correctness, 243
    error handling, 253
    Greek question mark, 251–252
    high-level validation tools, 252
    HTML, 251
    namespace std, 245–246
    real-world experience, 243–245
    security decisions based on names, 253
    validating comment fields, 249–250
    validating date and time, 247
    validating open spots, 248–249
    validating vaccination center ID, 246
    validating vaccination type, 248
  encoding data, 256
  operations, Azure Key Vault, 290
  rejecting known bad data, determining
    correctness, 253–255
version numbers
  cryptographic agility, 320–322
  GUID, 324
  UUID, 324
Virtual Machine Scale Set (VMSS) agents, 440
Virtual Nets (VNets), 443–445
  ADO SHA, 456
  agents, 456–458
  AKS, 457
  allowed communications, 448
  API Management Gateways, 451
  ASG, 447
  Azure Application Proxy, 451
  Azure Firewall, 449–450
  Azure Firewall Premium SKU, 450
  Azure WAF, 450–451
  cluster network policies, 457–458
  CNAME, 459–460
  DEV environments, 448
  DEVINT environments, 448
  DNS, 454–455
  hubs, 447
  ingress/egress controls, 449, 457
  IPv4, 445
    addresses, 445–446
    concepts, 445
    routing, 446
    UDR, 445
  IPv6, 445
  landing zones, 447
  managing, 456
  NONPROD environments, 448
  NSG, 446–447
  NVA, 449
  PaaS
    dedicated PaaS instances, 456
    private networking, 451–452
    private shared PaaS, 452–455
  private endpoints, 454–455, 457
  PROD environments, 448
  SANDBOX environments, 448
  segmenting, 447–448
  shared app service plans, 455
  spokes, 447
Virtualization-Based Security (VBS), 368
Vlisside, John, design patterns, 51
VM (Virtual Machines)
  Azure VM
    cryptography, 335–337
    Intel SGX, 361–362, 363–364
    Intel SGX, keys, 364
    SEV-SNP, 362–363
    trusted launches, 366–367
  DCsv3 VM, 363–364

managed identities, 403
TLS
Linux VM, 350–351
Windows VM, 352–353
VMSS (Virtual Machine Scale Set) agents, 440
VNets (Virtual Nets), 443–445
ADO SHA, 456
agents, 456–458
AKS, 457
allowed communications, 448
API Management Gateways, 451
ASG, 447
Azure Application Proxy, 451
Azure Firewall, 449–450
Azure Firewall Premium SKU, 450
Azure WAF, 450–451
cluster network policies, 457–458
CNAME, 459–460
DEV environments, 448
DEVINT environments, 448
DNS, 454–455
hubs, 447
ingress/egress controls, 449, 457
IPv4, 445
addresses, 445–446
concepts, 445
routing, 446
UDR, 446
IPv6, 445
landing zones, 447
managing, 456
NONPROD environments, 448
NSG, 446–447
NVA, 449
PaaS
dedicated PaaS instances, 456
private networking, 451–452
private shared PaaS, 452–455
private endpoints, 454–455, 457
PROD environments, 448
SANDBOX environments, 448
segmenting, 447–448
shared app service plans, 455
spokes, 447
vulnerabilities, 256–257
authentication failures, 264–266
binaries, signing, 268–269
broken access control, 257
components, 264
cryptography failures, 257–258
CVE, 230
CVSS, 230
data integrity failures, 266–269
debugging security headers, 264
identification failures, 264–266
injection, 258
inline scripts/styles, 260–263
insecure design, 259
logging failures, 269
managing, 204–205
misconfigured security, 259–260
monitoring failures, 269
NVD, 230
scoring (CVSS), 7
signatures, checking, 268
software failures, 266–269
SSRF, 269–270
vulnerable/outdated components, 264

W

WAF (Web Application Firewalls), 30, 450–451
weakest link, 47–48
Well-Architected Framework, Azure, 53
Wikipedia, DMZ, 34
Windows VM, TLS, 352–353
Wireshark, debugging TLS, 355–356
workstations, PAW, 42
wrapping operations, Azure Key Vault, 290
Writing Secure Code, 6

X - Y - Z

XSS (Cross-Site Scripting), 30, 258

zero trust, 27–31
ZKPP (Zero-Knowledge Password Proof), 151
ZRS (Zone-Redundant Storage), 47