# contents

preface

acknowledgments

about this book

about the author

about the cover illustration

Part 1. Foundations

*1* What is API security?

1.1 An analogy: Taking your driving test

1.2 What is an API?

API styles

1.3 API security in context

A typical API deployment

1.4 Elements of API security

Assets

Security goals

Environments and threat models

1.5 Security mechanisms

Encryption

Identification and authentication

Access control and authorization

Audit logging

Rate-limiting

*2* Secure API development

2.1 The Natter API

Overview of the Natter API

Implementation overview

Setting up the project

Initializing the database

2.2 Developing the REST API

Creating a new space

2.3 Wiring up the REST endpoints

Trying it out

2.4 Injection attacks

Preventing injection attacks

Mitigating SQL injection with permissions

2.5 Input validation

2.6 Producing safe output

Exploiting XSS Attacks

Preventing XSS

Implementing the protections

*3* Securing the Natter API

3.1 Addressing threats with security controls

3.2 Rate-limiting for availability

Rate-limiting with Guava

3.3 Authentication to prevent spoofing

HTTP Basic authentication

Secure password storage with Scrypt

Creating the password database

Registering users in the Natter API

Authenticating users

3.4 Using encryption to keep data private

Enabling HTTPS

Strict transport security

3.5 Audit logging for accountability

3.6 Access control

Enforcing authentication

Access control lists

Enforcing access control in Natter

Adding new members to a Natter space

Avoiding privilege escalation attacks

Part 2. Token-based authentication

*4* Session cookie authentication

4.1 Authentication in web browsers

Calling the Natter API from JavaScript

Intercepting form submission

Serving the HTML from the same origin

Drawbacks of HTTP authentication

4.2 Token-based authentication

A token store abstraction

Implementing token-based login

4.2 Session cookies

Avoiding session fixation attacks

Cookie security attributes

Validating session cookies

4.3 Preventing Cross-Site Request Forgery attacks

SameSite cookies

Hash-based double-submit cookies

Double-submit cookies for the Natter API

4.4 Building the Natter login UI

Calling the login API from JavaScript

4.5 Implementing logout

*5* Modern token-based authentication

5.1 Allowing cross-domain requests with CORS

Preflight requests

CORS headers

Adding CORS headers to the Natter API

5.2 Tokens without cookies

Storing token state in a database

The Bearer authentication scheme

Deleting expired tokens

Storing tokens in Web Storage

Updating the CORS filter

XSS attacks on Web Storage

5.3 Hardening database token storage

Hashing database tokens

Authenticating tokens with HMAC

Protecting sensitive attributes

*6* Self-contained tokens and JWTs

6.1 Storing token state on the client

Protecting JSON tokens with HMAC

6.2 JSON Web Tokens

The standard JWT claims

The JOSE header

Generating standard JWTs

Validating a signed JWT

6.3 Encrypting sensitive attributes

Authenticated encryption

Authenticated encryption with NaCl

Encrypted JWTs

Using a JWT library

6.4 Using types for secure API design

6.5 Handling token revocation

Implementing hybrid tokens

Part 3. Authorization

7 OAuth2 and OpenID Connect

7.1 Scoped tokens

Adding scoped tokens to Natter

The difference between scopes and permissions

7.2 Introducing OAuth

Types of clients

Authorization grants

Discovering OAuth2 endpoints

7.3 The Authorization Code grant

Redirect URIs for different types of clients

Hardening code exchange with PKCE

Refresh tokens

7.4 Validating an access token

Token introspection

Securing the HTTPS client configuration

Token revocation

JWT access tokens

Encrypted JWT access tokens

Letting the AS decrypt the tokens

7.5 Single sign-on

7.6 OpenID Connect

ID tokens

Hardening OIDC

Passing an ID token to an API

*8* Identity-based access control

8.1 Users and groups

LDAP groups

8.2 Role-based access control

Mapping roles to permissions

Static roles

Determining user roles

Dynamic roles

8.3 Attribute-based access control

Combining decisions

Implementing ABAC decisions

Policy agents and API gateways

Distributed policy enforcement and XACML

Best practices for ABAC

*9* Capability-based security and macaroons

9.1 Capability-based security

9.2 Capabilities and REST

Capabilities as URIs

Using capability URIs in the Natter API

HATEOAS

Capability URIs for browser-based clients

Combining capabilities with identity

Hardening capability URIs

9.3 Macaroons: Tokens with caveats

Contextual caveats

A macaroon token store

First-party caveats

Third-party caveats

Part 4. Microservice APIs in Kubernetes

*10* Microservice APIs in Kubernetes

10.1 Microservice APIs on Kubernetes

10.2 Deploying Natter on Kubernetes

Building H2 database as a Docker container

Deploying the database to Kubernetes

Building the Natter API as a Docker container

The link-preview microservice 353Deploying the new microservice

Calling the link-preview microservice

Preventing SSRF attacks

DNS rebinding attacks

10.3 Securing microservice communications

Securing communications with TLS

Using a service mesh for TLS

Locking down network connections

10.4 Securing incoming requests

*11* Securing service-to-service APIs

11.1 API keys and JWT bearer authentication

11.2 The OAuth2 client credentials grant

Service accounts

11.3 The JWT bearer grant for OAuth

Client authentication

Generating the JWT

Service account authentication

11.4 Mutual TLS authentication

How TLS certificate authentication works

Client certificate authentication

Verifying client identity

Using a service mesh

Mutual TLS with OAuth

Certificate-bound access tokens

11.5 Managing service credentials

Kubernetes secrets

Key and secret management services

Avoiding long-lived secrets on disk

Key derivation

11.6 Service API calls in response to user requests

The phantom token pattern

OAuth2 token exchange

Part 5. APIs for the Internet of Things

*12* Securing IoT communications

12.1 Transport layer security

Datagram TLS

Cipher suites for constrained devices

12.2 Pre-shared keys

Implementing a PSK server

The PSK client

Supporting raw PSK cipher suites

PSK with forward secrecy

12.3 End-to-end security

COSE

Alternatives to COSE

Misuse-resistant authenticated encryption

12.4 Key distribution and management

One-off key provisioning

Key distribution servers

Ratcheting for forward secrecy

Post-compromise security

*13* Securing IoT APIs

13.1 Authenticating devices

Identifying devices

Device certificates

Authenticating at the transport layer

13.2 End-to-end authentication

OSCORE

Avoiding replay in REST APIs

13.3 OAuth2 for constrained environments

The device authorization grant

ACE-OAuth

13.4 Offline access control

Offline user authentication

Offline authorization

*appendix A.* Setting up Java and Maven

*appendix B.* Setting up Kubernetes

index