

Authentifizierung und Autorisierung

Das Handbuch für die Webentwicklung

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhalt

Geleitwort des Fachgutachters	17
Danksagung	19
 1 Einführung in die moderne Sicherheit von Webanwendungen	 21
1.1 Welche Bedeutung hat die Absicherung von Webanwendungen?	21
1.2 Die Entwicklung von Sicherheitspraktiken für Webanwendungen	22
1.3 OAuth2 und OpenID Connect – die heutigen Sicherheits herausforderungen bewältigen	23
1.4 Die Rolle von OAuth2 und OpenID Connect bei der Abwehr neuer Sicherheitsbedrohungen	24
 2 Das Grundprinzip der Authentifizierung	 27
2.1 Passwortbasierte Authentifizierung	28
2.2 Cookiebasierte Authentifizierung	29
2.3 Tokenbasierte Authentifizierung	30
2.3.1 Wie funktioniert die tokenbasierte Authentifizierung?	31
2.4 Zertifikatsbasierte Authentifizierung	33
2.4.1 Wie funktioniert die zertifikatsbasierte Authentifizierung?	33
2.4.2 Komponenten der zertifikatsbasierten Authentifizierung	34
2.5 Biometrische Authentifizierung	35
2.6 Multi-Faktor-Authentifizierung	37
2.6.1 Authentifizierungsfaktoren	38
2.7 Vergleich der Authentifizierungsmethoden	38
2.8 Single Sign-on: Zentralisierte Authentifizierung für mehr Komfort und Sicherheit	40
2.8.1 Wie funktioniert Single Sign-on?	40
2.8.2 Die Risiken von Single Sign-on	42
2.9 FIDO: Sicherheitsstandards für starke Authentifizierung	44
2.9.1 Die Mission der FIDO-Allianz	45
2.9.2 Public-Key-Kryptografie	46
2.9.3 Die FIDO-Spezifikationen im Detail	47

- 3 Autorisierung: Wer darf was? 65**
 - 3.1 Die Grundlagen der Autorisierung 66**
 - 3.1.1 Subjekte, Ressourcen und Aktionen 66
 - 3.1.2 Wichtige Sicherheitsprinzipien 67
 - 3.2 Klassische Autorisierungsmodelle 68**
 - 3.2.1 Zugriffskontrolllisten (Access Control Lists, ACL) 69
 - 3.2.2 Rollenbasierte Zugriffskontrolle (RBAC) 69
 - 3.2.3 Attributbasierte Zugriffskontrolle (ABAC) 71
 - 3.2.4 Policy-basierte Autorisierung (PBAC) 72
 - 3.2.5 Beziehungsbasierte Zugriffskontrolle (ReBAC) 73
 - 3.2.6 Discretionary Access Control (DAC) –
 eigentümerbasierte Kontrolle 74
 - 3.2.7 Mandatory Access Control (MAC) –
 zentral kontrollierte Sicherheit 75
 - 3.2.8 Autorisierungsmodelle im Vergleich – ein Fazit 76
 - 3.3 Autorisierung in modernen Webanwendungen 78**
 - 3.3.1 Die Herausforderungen verteilter Systeme 78
 - 3.3.2 Zentrale vs. dezentrale Autorisierung 79
 - 3.3.3 Multi-Tenancy: Mandantenfähige Anwendungen 81
 - 3.3.4 Performance und Skalierung 82
 - 3.4 Moderne Ansätze und Standards 83**
 - 3.4.1 OAuth 2.0 und OpenID Connect 83
 - 3.4.2 Zero Trust und kontinuierliche Autorisierung 85
 - 3.4.3 Fine-Grained Authorization mit OpenFGA 86
 - 3.5 Praktische Umsetzung 87**
 - 3.5.1 Wo wird autorisiert? Durchsetzungspunkte 87
 - 3.5.2 Umgang mit Fehlern und Grenzfällen 88
 - 3.5.3 Logging und Auditierung 89
 - 3.6 Herausforderungen und Zukunftsausblick 91**
 - 3.6.1 Die Balance zwischen Sicherheit und Benutzerfreundlichkeit 91
 - 3.6.2 Die Zukunft der Autorisierung 92
 - 3.6.3 Was bleibt: Grundprinzipien der Autorisierung 93
 - 3.7 Fazit 94**

4	Das JSON Web Token	97
4.1	Einführung und Verwendung eines JSON Web Tokens	97
4.1.1	Authentifizierung	98
4.1.2	Autorisierung	98
4.1.3	Sicherer Informationsaustausch	99
4.2	Anatomie eines JWT	99
4.3	JWT-Claims im Detail	103
4.3.1	Registrierte Claims	103
4.3.2	Öffentliche Claims	105
4.3.3	Private Claims	105
4.4	JSON Web Signature (JWS)	107
4.4.1	Eine JSON Web Signature erstellen	108
4.5	JSON Web Encryption (JWE)	117
4.5.1	Verschlüsseln und Entschlüsseln von Daten	118
4.5.2	JWT-Verschlüsselung mit JWE	120
4.6	Sign-then-Encrypt: Die Kombination von JWS und JWE	126
4.7	Bedrohungen und Schwachstellen bei der Verwendung von JWTs	127
4.7.1	Schwache Signaturen und unzureichende Signaturprüfung	127
4.7.2	Bedrohungen durch schwache oder fehlerhafte kryptografische Algorithmen	128
4.7.3	Informationsleck durch Analyse der Länge des Chiffretexts	128
4.7.4	Mehrdeutige JSON-Codierungen	128
4.7.5	Substitutionsangriffe zwischen Empfängern	129
4.7.6	Verwechslung von Token-Typen (Cross-JWT Confusion)	129
4.8	Bewährte Praktiken im Umgang mit JSON Web Tokens	129
4.8.1	Den verwendeten Algorithmus strikt validieren	129
4.8.2	Alle kryptografischen Operationen validieren	130
4.8.3	Kryptografische Schlüssel mit ausreichender Entropie sicherstellen	130
4.8.4	Korrekte Verwendung und Validierung der Claims	131
4.8.5	Validierungsregeln kontextspezifisch gestalten	131
4.8.6	Weiterführende Praktiken	132
4.9	Post-Quantum-JWTs-Standards	133
4.9.1	Kryptografische Bedrohung durch Quantencomputer	133
4.9.2	Anforderungen an Post-Quantum-JWTs	134
4.9.3	Aktueller Stand der Standardisierung	134
4.9.4	Praktische Empfehlungen	135
4.9.5	Ausblick	135

5	Die Entwicklung von OAuth2 und OpenID Connect: Eine historische Betrachtung	137
5.1	Von zentralisierten zu föderierten Identitäten	137
5.2	Die Ära vor OAuth: SAML und seine Grenzen	138
5.2.1	Enterprise-Fokus und XML-basierte Architektur	140
5.2.2	Stärken und Einschränkungen für das moderne Web	141
5.2.3	Der Übergang zu leichtgewichtigeren Alternativen	142
5.3	Die Geburt von OAuth 1.0	143
5.3.1	Das Twitter-Problem: API-Zugriff ohne Passwort-Weitergabe	143
5.3.2	Entwicklung und Standardisierung	144
5.3.3	Standardisierung durch die IETF	145
5.3.4	Technische Herausforderungen: Signatur-Komplexität und mobile Einschränkungen	146
5.4	OAuth 2.0: Die Revolution – von Signaturen zu Bearer-Tokens	148
5.4.1	Das OAuth-2.0-Framework	149
5.4.2	Die vier Grant Types und ihre Anwendungsfälle	150
5.4.3	Die Kontroverse: Eran Hammer-Lahavs Ausstieg und Kritik	151
5.5	Die Authentifizierungslücke – warum OAuth 2.0 allein nicht reichte	152
5.5.1	Die OpenID-Herausforderung	152
5.5.2	OpenID Connect: Die Neuerfindung	153
5.5.3	OAuth 2.0 als Fundament für die Authentifizierung	154
6	OAuth 2.0: Sichere Autorisierung für moderne Webanwendungen	155
6.1	Die Rollen im OAuth-2.0-Ökosystem	156
6.1.1	Resource-Owner	156
6.1.2	Resource-Server	156
6.1.3	Client	157
6.1.4	Authorization-Server	157
6.1.5	Der abstrakte OAuth-2.0-Protokollfluss	158
6.2	Kernkonzepte von OAuth 2.0	160
6.2.1	Die Idee: Delegierte Autorisierung	160
6.2.2	Scopes: Granulare Berechtigungen definieren	161
6.2.3	Consent: Die Zustimmung des Resource-Owners	164
6.3	Grant-Typen: Wege zur Autorisierung	167
6.3.1	Client-Registrierung und Authentifizierung	167
6.3.2	Authorization-Code-Grant	178

6.3.3	Implicit-Grant	189
6.3.4	Resource-Owner-Password-Credentials-Grant	195
6.3.5	Client-Credentials-Grant	201
6.4	Access-Tokens: Das Herzstück der OAuth-Autorisierung	205
6.4.1	Struktur und Format: JWT vs. Opaque Tokens	205
6.4.2	Standard-Claims im Access-Token	209
6.4.3	Token-Expiration: Sicherheitsabwägungen	213
6.5	Refresh-Tokens: Langlebige Sitzungen ohne Passwörter	214
6.5.1	Grundlagen und Eigenschaften von Refresh-Tokens	214
6.5.2	Der Refresh-Token-Grant-Flow	216
6.5.3	Die Refresh-Token-Rotation	218
6.5.4	Sicherheitsaspekte von Refresh-Tokens	220
6.6	Management der Tokens	221
6.6.1	Token-Introspection	222
6.6.2	Token-Revocation	224
6.7	Referenzen und weiterführende Standards	226
7	OpenID Connect: Authentifizierung auf Basis von OAuth 2.0	229
7.1	Einführung in OpenID Connect	230
7.1.1	Der openid-Scope als Authentifizierungssignal	231
7.1.2	Das ID-Token: Verifizierbare Identitätsaussagen	231
7.1.3	Der UserInfo-Endpoint: Erweiterbare Profilinformationen	232
7.1.4	Standardisierung als Schlüssel zur Interoperabilität	232
7.1.5	Die drei Hauptakteure in OIDC	232
7.1.6	Der abstrakte OIDC-Protokollfluss	235
7.2	Discovery und Metadaten: Wie Clients OpenID-Provider finden	237
7.2.1	OpenID-Provider Issuer Discovery	237
7.2.2	Das OpenID-Provider-Metadaten-Dokument	238
7.2.3	Das JWKS-Dokument: Öffentliche Schlüssel des Providers	242
7.2.4	Der Discovery-Prozess in der Praxis	244
7.2.5	Das Verfügbarkeitsrisiko des Discovery-Prozesses	245
7.2.6	Dynamic Client Registration	245
7.3	Das ID-Token: Identitätsnachweis in OpenID Connect	247
7.3.1	Struktur und Format des ID-Tokens	247
7.3.2	Standard-Claims im ID-Token	248
7.3.3	ID-Token vs. Access-Token: Der entscheidende Unterschied	251

7.4	OpenID-Connect-Flows	254
7.4.1	OpenID-Connect-Standard-Scopes	254
7.4.2	Response-Types und Response-Modes	256
7.4.3	Authorization-Code-Flow mit OIDC	260
7.4.4	Implicit Flow mit OIDC	263
7.4.5	Hybrid Flow	264
7.5	Nutzerdaten abrufen: Der UserInfo-Endpoint	267
7.5.1	Funktionsweise und Zugriff	267
7.5.2	Claims im ID-Token vs. Claims über UserInfo	268
7.5.3	Sicherheit des UserInfo-Endpoints	269
7.5.4	Aggregated und Distributed Claims	271
7.5.5	Validierung der UserInfo-Response	272
7.6	Social Login und Identity-Federation	273
7.6.1	Social Login als bequeme Login-Methode?	274
7.6.2	Integration externer Identity-Provider	279
7.6.3	Account-Linking und Merge-Strategien	281
7.6.4	Identity-Brokering und zentrale Authentifizierung	289
7.7	Sessions und Authentifizierungsstatus verwalten	294
7.7.1	Session-Lebensdauer vs. Token-Lebensdauer	295
7.7.2	Token-Erneuerung ohne Nutzerinteraktion (Silent Authentication)	297
7.7.3	Session-Monitoring mit dem Check-Session-iFrame	299
7.7.4	Aktualität der Authentifizierung: max_age und auth_time	306
7.8	Session-Beendigung und koordinierter Logout	309
7.8.1	Local Logout vs. Global Logout	310
7.8.2	Relying-Party-Initiated Logout (End-Session-Endpoint)	312
7.8.3	Front-Channel-Logout	315
7.8.4	Back-Channel-Logout	317
7.8.5	Single Logout in Multi-App-Umgebungen	321
7.9	Fazit	323
8	OAuth in Microservice-Architekturen	325
8.1	Das Problem: Delegation-Chains in Microservices	326
8.1.1	Verletzung des Principle of Least Privilege	326
8.1.2	Fehlende Audience-Restriction	326
8.1.3	Kein aussagekräftiger Audit-Trail	327

8.1.4	Eine lange Token-Lebensdauer erhöht das Angriffsrisiko	327
8.1.5	Alternative Ansätze und ihre Schwächen	327
8.2	Die Lösung: Token-Exchange	329
8.2.1	Token-Exchange-Patterns	332
8.2.2	Integration mit bestehenden Infrastrukturen	332
8.2.3	Grenzen und Kompromisse	333
8.3	Token-Exchange – der Standard	333
8.3.1	Der Token-Exchange-Grant-Typ	334
8.3.2	Das Request-Format im Detail	336
8.3.3	Das Response-Format	338
8.3.4	Subject-Token vs. Actor-Token	339
8.4	Der On-Behalf-Of-Flow: Impersonation	341
8.4.1	Ablauf des On-Behalf-Of-Flows	341
8.4.2	Sicherheitsaspekte bei der Nutzung des On-Behalf-Of-Flows	346
8.5	Delegation-Flow: Explizite Service-Identifikation	347
8.5.1	Delegation vs. Impersonation	347
8.5.2	Der entscheidende Unterschied: Das Actor-Token	348
8.5.3	Ablauf des Delegation-Flows	351
8.5.4	Delegation-Chains	354
8.5.5	Die Grenzen von Delegation-Chains	354
8.5.6	Anwendung von Delegation	355
8.5.7	Sicherheitsaspekte bei der Verwendung von Delegation	356
8.6	Token-Translation: Format und Audience	358
8.6.1	Formatkonvertierung	359
8.6.2	Audience-Anpassung	361
8.6.3	Scope-Mapping	362
8.6.4	Token-Translation in der Praxis	363
8.6.5	Sicherheitsaspekte bei Token-Translation	363
8.7	Praktische Integration und Architektur-Patterns	365
8.7.1	Das API-Gateway-Pattern	365
8.7.2	Der Security-Token-Service als zentrale Komponente	367
8.7.3	Service-Mesh-Integration	368
8.7.4	Architekturentscheidungen und -kompromisse	370
8.8	Ausblick: Die Zukunft der Service-zu-Service-Authentifizierung	371
8.8.1	Transaction-Tokens: Token-Exchange für Microservice-Ketten	371
8.8.2	Workload-Identities: Identitäten für Non-Human Entities	372

9	Sicherheit und Bedrohungsmodelle in OAuth 2.0	375
9.1	Das Angreifermodell	376
9.1.1	Web-Attacker	377
9.1.2	Network-Attacker	377
9.1.3	Angreifer mit Lesezugriff auf die Authorization-Response	378
9.1.4	Angreifer mit Lesezugriff auf den Authorization-Request	378
9.1.5	Angreifer mit Zugriff auf Access-Tokens	379
9.1.6	Kombinationen und reale Bedrohungen	379
9.2	Bedrohungen im Authorization-Code-Flow	380
9.2.1	Authorization-Code-Injection	380
9.2.2	Authorization-Code-Leakage	383
9.2.3	Mix-Up-Attacks	386
9.2.4	Browser-Swapping-Attack	389
9.2.5	PKCE-Downgrade-Attack	392
9.2.6	Zusammenfassung	395
9.3	Bedrohungen beim Token-Handling	396
9.3.1	Access-Token-Leakage	396
9.3.2	Token-Replay und -Missbrauch	398
9.4	Schutzmaßnahmen für Refresh-Tokens	400
9.4.1	Gegenmaßnahmen im Detail	401
9.4.2	Zusammenfassung	402
9.5	Clientseitige Bedrohungen	402
9.5.1	Client-Impersonation	403
9.5.2	Client-Secret-Leakage	406
9.5.3	Open Redirectors auf der Clientseite	409
9.5.4	Phishing und Social Engineering	412
9.5.5	Zusammenfassung	415
9.6	Security Checklist für OAuth-2.0-Implementierungen	416
9.6.1	Authorization-Server	416
9.6.2	Clients (siehe Kapitel 6)	417
9.6.3	Resource-Server (siehe Kapitel 6)	418
9.7	Ausblick	419

10	OAuth 2.0 für browserbasierte Anwendungen	421
10.1	Browserbasierte Anwendungen und OAuth:	
	Die Sicherheitsherausforderung	423
10.1.1	Single-Page-Applications als Public Clients	423
10.1.2	Die Bedrohungslandschaft: XSS, Token-Theft und Client-Hijacking	424
10.1.3	Token-Speicherung im Browser: Ein unlösbares Problem?	430
10.2	Das Backend-for-Frontend-Pattern: Die Lösung für sichere Browseranwendungen	432
10.2.1	Die Kernidee: Tokens betreten niemals den Browserkontext	433
10.2.2	Die Architektur im Detail: Komponenten und Verantwortlichkeiten	434
10.2.3	Der BFF-Flow: Von der Anmeldung bis zum API-Zugriff	435
10.2.4	Cookie-Sicherheit: Das Fundament der BFF-Architektur	440
10.2.5	Session-Verwaltung: Serverseitig oder clientseitig?	441
10.2.6	CSRF-Schutz: Notwendige Ergänzung zur Cookie-Sicherheit	442
10.2.7	Fazit: Warum BFF alle Token-Probleme löst	443
10.2.8	Zielkonflikte und Überlegungen	444
11	OAuth 2.1 und moderne Sicherheits-erweiterungen	447
11.1	OAuth 2.1: Die Konsolidierung der Best Practices	448
11.1.1	Die wichtigsten Änderungen im Überblick	449
11.1.2	Rückwärtskompatibilität	452
11.2	Sender-Constrained Tokens: DPoP und mTLS	453
11.2.1	Das Problem mit Bearer-Tokens	453
11.2.2	Zwei unterschiedliche Ansätze	454
11.2.3	DPoP – Demonstrating Proof-of-Possession	455
11.2.4	mTLS – Mutual TLS Certificate-Bound Access-Tokens	465
11.2.5	DPoP vs. mTLS: Wann welche Lösung?	477
11.3	Absicherung des Authorization-Requests: PAR und JAR	477
11.3.1	Schwachstellen des traditionellen Authorization-Requests	478
11.3.2	JAR – JWT Secured Authorization Request	480
11.3.3	PAR – Pushed Authorization-Requests	484
11.4	Zusammenfassung und Kernbotschaften	491

12	Praktische Implementierung von OAuth 2.0 und OIDC	493
12.1	Authorization-Server im Vergleich	495
12.1.1	Deployment-Modelle: Self-Hosted, Cloud und Embedded	495
12.1.2	Vergleichstabelle: Die vier Lösungen im Überblick	500
12.1.3	Terminologie-Übersicht der vier Authorization-Server-Lösungen	502
12.2	Voraussetzungen und Setup	505
12.3	Szenario 1: Authorization-Code-Flow mit PKCE für Browseranwendungen	506
12.3.1	Architektur-Überblick	506
12.3.2	Auth0-Konfiguration	507
12.3.3	Die Anwendung in Aktion	508
12.3.4	Angular-Client: Authorization-Code-Flow mit PKCE	511
12.3.5	.NET Core C# API: Access-Token-Validierung	516
12.4	Szenario 2: Client-Credentials-Flow für die Service-to-Service-Kommunikation	522
12.4.1	Architektur-Überblick	522
12.4.2	Keycloak-Konfiguration	524
12.4.3	Java-Client: Token-Beschaffung mit Client-Credentials	528
12.4.4	application.properties: OAuth2-Client-Konfiguration	529
12.4.5	Java-API: Token-Validierung im Resource-Server	536
12.5	Szenario 3: Token-Exchange in Microservice-Architekturen	543
12.5.1	Architektur-Überblick	544
12.5.2	IdentityServer-Setup	546
12.5.3	Order-Service: Token-Exchange ausführen	554
12.5.4	Inventory-Service: Token-Validierung und User-Kontext	561
12.6	Szenario 4: Das Backend-for-Frontend-(BFF-)Pattern	564
12.6.1	Architektur-Überblick	564
12.6.2	Keycloak-Konfiguration	565
12.6.3	Angular-Frontend: session-basierte Kommunikation	566
12.6.4	BFF-Backend: Authorization-Code-Flow und Token-Management	568
12.6.5	BFF-Proxy-Endpoints mit YARP	572
12.7	Production-Readiness: Von der Implementierung zum produktiven Betrieb	575
12.7.1	Skalierung und Hochverfügbarkeit	575
12.7.2	Monitoring und Observability	576

12.7.3	Security-Hardening	578
12.7.4	Compliance und Auditing	579
12.7.5	Incident-Response	580
12.7.6	Fazit: Production-Readiness als kontinuierlicher Prozess	581
13	Zusammenfassung und Ausblick	583
13.1	Von der Theorie zur Praxis: Was Sie gelernt haben	584
13.2	Entscheidungshilfen: Die richtigen Technologien wählen	585
13.2.1	Welcher OAuth-Flow ist der richtige?	585
13.2.2	Klare Empfehlungen für die Zukunft	587
13.2.3	Was Sie nicht mehr nutzen sollten	587
13.3	Financial-grade API (FAPI): Höchste Sicherheit für kritische Anwendungen	588
13.3.1	Was ist FAPI und warum existiert es?	588
13.3.2	Das Problem, das FAPI löst	588
13.3.3	FAPI 2.0: Die aktuelle Generation	589
13.3.4	FAPI-2.0-Security-Profile im Detail	590
13.3.5	Vergleich zwischen Standard-OAuth-2.0 und FAPI 2.0	591
13.3.6	Einsatzgebiete und Anwendungsfälle von FAPI	591
13.3.7	Praktische Implementierungshinweise	593
13.4	Zukünftige Entwicklungen: Was kommt nach OAuth 2.1?	593
13.4.1	Rich Authorization-Requests (RAR)	594
13.4.2	Continuous Authentication und Step-Up Authentication	597
13.4.3	Decentralized Identity und Verifiable Credentials	600
13.5	Weiterführende Ressourcen und Communities	606
13.5.1	Offizielle Standards und Spezifikationen	606
13.5.2	Praktische Tools und Libraries	607
13.5.3	OAuth-Implementierungen und Libraries: Die zentrale Ressource	608
13.5.4	Blogs, Podcasts und Communities	611
13.5.5	Weiterbildung und Zertifizierungen	612
13.6	Abschließende Worte	614
	Index	617