Sicherheitsvorgaben für die Absicherung von APIs für den Schutzbedarf Normal

> 👎 👍 → Feedback und Anmerkungen zu diesem Dokument geben

Zusammenfassung

Das vorliegende Dokument definiert Mindestanforderungen und empfiehlt sinnvolle Sicherheitspraktiken auf Basis des "Best Current Practice for OAuth 2.0 Security" [RFC9700] Dokuments für Absicherung von APIs, die dem Schutzbedarf normal entsprechen.

Einleitung

Das vorliegende Dokument definiert Mindestanforderungen und nennt sinnvolle Sicherheitspraktiken auf Basis von "Best Current Practice for OAuth 2.0 Security" [RFC9700] für die Absicherung von APIs, die dem Schutzbedarf normal entsprechen. Viele Sicherheitsempfehlungen, die in [RFC9700] genannt werden, werden in OAuth 2.1 [draft-ietf-oauth-v2-1-14] übernommen oder sogar verschärft. Einige Sicherheitsempfehlungen von OAuth 2.1, die jetzt schon als stabil angesehen werden können, wurden schon in das vorliegende Dokument übernommen. OAuth 2.1 ist aktuell im Draft Status, sobald OAuth 2.1 einen finalen Status erhält, wird dieses Vorgabendokument überarbeitet und entsprechend OAuth 2.1 als zentral einzuhaltende Vorgabenquelle referenzieren, da OAuth 2.1 nach aktuellem Kenntnisstand alle bekannten Sicherheitsmaßnahmen konsolidiert.

Zum besseren Verständnis der unterliegenden Sicherheitsüberlegungen und Darstellung weiterer Sicherheitsmaßnahmen, wurde auf Basis [RFC9700] das Dokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“ als Begleitdokument veröffentlicht. Dieses Dokument muss nicht zwingend beachtet werden, aber es wird empfohlen, um die Umsetzung von Anforderungen mit Empfehlungscharakter besser zu bewerten und die IT-Sicherheit von API-Absicherungen weiter zu erhöhen.

Schlüsselwörter

Die Schlüsselwörter „MUSS“ (DARF NUR), „DARF NICHT“, „SOLLTE“, „SOLLTE NICHT“, „DARF“ und „KANN“ in diesem Dokument sind gemäß DIN-Norm DIN 820-2 - 2022-12 [DIN 820-2] zu interpretieren. Diese Schlüsselwörter werden nicht als Wörterbuchbegriffe verwendet, sodass jedes Vorkommen als Schlüsselwort zu interpretieren ist und nicht mit ihrer natürlichen Sprachbedeutung.

1. Geltungsbereich

Dieses Sicherheitsprofil ist für alle APIs verpflichtend zu beachten, deren Anwendungskontext in den Schutzbedarf „Normal“ fällt. Der technische Geltungsbereich gilt grundsätzlich für alle API-Technologievarianten, die OAuth einsetzen können. Um die Anwendung dieses Profils zu vereinfachen, werden für typische API-Technologievarianten ergänzende Hinweise gegeben.

- Für REST APIs sind diese Vorgaben immer als verpflichtend anzusehen.
- Für gRPC und GraphQL wird der Einsatz für OAuth vorgeschrieben, wenn ein Bedarf an der Autorisierungsprüfung der Schnittstellennutzer besteht. Demnach ist auch hier der Einsatz dieses Profils

verpflichtend.
- Für alle Schnittstellen eines Message Broker (wie bspw. Kafka) wird der Einsatz von OAuth empfohlen. Wenn die Nutzung des OAuth Protokolls geplant ist, gilt die verpflichtende Nutzung der Vorgaben in diesem Dokument.
- SOAP APIs sind solange von diesen Vorgaben ausgenommen, solange kein OAuth benutzt wird. Werden jedoch Mechanismen genutzt, um nutzerseitige Autorisierungen abzubilden, die nicht auf OAuth basieren, ist eine gleichwertige Sicherheit nachzuweisen. Falls im Rahmen einer Modernisierung von Legacy SOAP APIs eine Umstellung auf OAuth erfolgt, sind diese Vorgaben ebenfalls als verpflichtend anzusehen.

2. Normative Verweise

Auf die folgenden Dokumente wird im Text in einer Weise verwiesen, dass ihr Inhalt ganz oder teilweise Anforderungen dieses Dokuments darstellen. Bei datierten Verweisen gilt nur die zitierte Ausgabe. Bei undatierten Verweisen gilt die neueste Ausgabe des referenzierten Dokuments (einschließlich aller Änderungen).

Normative Verweise finden Sie in Kapitel 6.1.

Weitere informative Referenzen finden sich in Kapitel 6.2.

3. Begriffe und Definitionen

Für die Zwecke dieses Dokuments gelten die in [RFC6749], [RFC6750], [RFC7636], [OIDC] und [ISO29100] definierten Begriffe. Zum besseren Verständnis werden im vorliegenden Dokument OAuth und OpenID Connect spezifische technische Begriffe in ihrem englischen Terminus verwendet.

4. Symbole und Abkürzungen

API – Application Programming Interface

BCM – Basin, Cremers, Meier

BCP – Best Current Practice

CAA – Certificate Authority Authorization

CIBA – Client Initiated Backchannel Authentication

CSRF – Cross-Site Request Forgery

DNS – Domain Name System

DNSSEC – Domain Name System Security Extensions

DPoP – Demonstrating Proof-of-Possession

HTTP – Hyper Text Transfer Protocol

IETF – Internet Engineering Task Force

JAR – JWT-Secured Authorization Request

JARM – JWT-Secured Authorization Response Mode

JWK – JSON Web Key

JWKS – JSON Web Key Sets

JWT – JSON Web Token

JOSE – JavaScript Object Signing and Encryption

JSON – JavaScript Object Notation

MTLS – Mutual Transport Layer Security

OIDF – OpenID Foundation

PAR – Pushed Authorization Requests

PKCE – Proof Key for Code Exchange

QR – Quick Response

RSA – Rivest-Shamir-Adleman

REST – Representational State Transfer

SOAP – Simple Object Access Protocol

TLS – Transport Layer Security

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

5. Empfehlungen und Vorgaben

In diesem Abschnitt werden die wichtigsten Sicherheitsmechanismen und -maßnahmen beschrieben, die zum Zeitpunkt der Erstellung dieses Dokuments als bewährte Verfahren gelten. Einzelheiten zu diesen Sicherheitsmechanismen und -maßnahmen (einschließlich detaillierter Beschreibungen der Angriffe) sowie Anforderungen für weniger häufig verwendete Optionen finden Sie in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“.

5.1. Verbotene Grant Types

Aufgrund von erheblichen Sicherheitsproblemen DÜRFEN folgende Grant Types nicht benutzt werden:

- Implicit Grant
- Resource Owner Password Credentials Grant

Diese Grant Types sind zwar noch in der aktuellen Version 0Auth 2.0 vorhanden, aber werden mit der nächsten Version OAuth 2.1 aufgrund der identifizierten Sicherheitsprobleme entfernt.

5.2. Schutz von Redirect-basierten Abläufen

Beim Vergleich von Client Redirection URI mit vorab registrierten URIs MÜSSEN Authorization Server eine exakte Zeichenfolgenübereinstimmung verwenden, mit Ausnahme von Portnummern in `localhost` Client Redirection URI von nativen Apps (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.1.3). Diese Maßnahme trägt dazu bei, das Durchsickern von Autorisierungscodes und Access Token zu verhindern (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.1). Sie kann auch dabei helfen, Verwechslungsangriffe zu erkennen (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.4).

Clients und Authorization Server DÜRFEN NICHT URLs offenlegen, die den Browser des Benutzers zu beliebigen URIs weiterleiten, die aus einem Abfrageparameter (offene Redirectoren) stammen, wie in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.11 beschrieben. Offene Redirectoren können die Exfiltration von Autorisierungscodes und Access Token ermöglichen.

Clients MÜSSEN Cross-Site-Request-Forgery (CSRF) verhindern. In diesem Zusammenhang bezieht sich CSRF auf Anfragen an den Redirection Endpoint, die nicht vom Authorization Server, sondern von einem böswilligen Dritten stammen (siehe Kapitel 4.4.1.8 von [RFC6819] für Details). Clients, die sichergestellt haben, dass der Authorization Server Proof Key for Code Exchange (PKCE) [RFC7636] unterstützt, KÖNNEN sich auf den von PKCE bereitgestellten CSRF-Schutz verlassen. In OpenID Connect-Abläufen bietet der Parameter `nonce` CSRF-Schutz. Andernfalls MÜSSEN einmalig verwendbare CSRF-Token, die im Parameter `state` enthalten sind und sicher an den Benutzeragenten gebunden sind, für den CSRF-Schutz verwendet werden (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.7.1).

Wenn ein OAuth-Client mit mehr als einem Authorization Server interagieren kann, ist eine Abwehr gegen Verwechslungsangriffe (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.4) ERFORDERLICH. Zu diesem Zweck SOLLTE ein Client

- den Parameter `iss` als Gegenmaßnahme gemäß [RFC9207] verwenden oder
- eine alternative Gegenmaßnahme basierend auf einem `iss`-Wert in der Autorisierungsantwort verwenden (z. B. den `iss`-Claim im ID-Token in [OpenID.Core] oder in [OpenID.JARM]-Antworten) und diesen Wert wie in [RFC9207] beschrieben verarbeiten.

Sind diese Optionen nicht verfügbar, KÖNNEN Clients stattdessen unterschiedliche Redirection URI verwenden, um Autorisierungsendpunkte und Token-Endpunkte zu identifizieren, wie in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.4.2 beschrieben.

Ein Authorization Server, der eine Anfrage umleitet, die möglicherweise Benutzeranmeldedaten enthält, MUSS vermeiden, diese Benutzeranmeldedaten versehentlich weiterzuleiten (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.12 für Details).

5.2.1. Authorization Code Grant

Clients MÜSSEN Angriffe durch Einfügen von Autorisierungscodes (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.5) und den Missbrauch von Autorisierungscodes verhindern, indem sie eine der folgenden Optionen verwenden:

- Public und Confidential Clients MÜSSEN zu diesem Zweck PKCE [RFC7636] verwenden, wie in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.5.3.1 begründet.
- Mit zusätzlichen Vorsichtsmaßnahmen, die in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.5.3.2 beschrieben sind, KÖNNEN Confidential OpenID Connect [OpenID.Core]-Clients stattdessen den Parameter `nonce` und den entsprechenden Claim im ID-Token verwenden.

In jedem Fall MUSS die PKCE-Herausforderung oder der OpenID Connect `nonce` transaktionsspezifisch sein und sicher an den Client und den Benutzeragenten gebunden sein, in dem die Transaktion gestartet wurde. Authorization Server werden aufgefordert, angemessene Anstrengungen zu unternehmen, um die Verwendung konstanter Werte für die PKCE-Herausforderung oder den OpenID Connect `nonce` zu erkennen und zu verhindern.

Hinweis: Obwohl PKCE als Mechanismus zum Schutz Public Clients entwickelt wurde, gilt dieser Hinweis für alle Arten von OAuth-Clients, einschließlich Webanwendungen.

Bei der Verwendung von PKCE SOLLTEN Clients PKCE-Code-Challenge-Methoden verwenden, die den PKCE-Verifizierer in der Autorisierungsanfrage nicht offenlegen. Andernfalls können Angreifer, die die Autorisierungsanfrage lesen können (vgl. Angreifer (A4) in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 3), die durch PKCE gewährleistete Sicherheit umgehen. Derzeit ist S256 die einzige solche Methode.

Authorization Server MÜSSEN PKCE [RFC7636] unterstützen.

Wenn ein Client einen gültigen PKCE Parameter `code_challenge` in der Autorisierungsanfrage sendet, MUSS der Authorization Server die korrekte Verwendung von `code_verifier` am Token-Endpunkt erzwingen.

Authorization Server MÜSSEN PKCE-Downgrade-Angriffe abwehren, indem sie sicherstellen, dass eine Token-Anforderung, die einen Parameter `code_verifier` enthält, nur akzeptiert wird, wenn ein Parameter `code_challenge` in der Autorisierungsanfrage vorhanden war; siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.8.2 für Details.

Authorization Server MÜSSEN eine Möglichkeit bieten, ihre Unterstützung für PKCE zu erkennen. Es wird EMPFOHLEN, dass Authorization Server das Element `code_challenge_methods_supported` in ihren Authorization Server Metadaten [RFC8414] veröffentlichen, dass die unterstützten PKCE-Challenge-Methoden enthält (die vom Client zur Erkennung der PKCE-Unterstützung verwendet werden können). Authorization Server KÖNNEN stattdessen eine einsatzspezifische Möglichkeit bereitstellen, um die PKCE-Unterstützung durch den Authorization Server sicherzustellen oder festzustellen.

5.3. Verhinderung von Token-Replay Angriffen

5.3.1. Sender-Constrained Access Token

Ein Sender-Constrained Access Token beschränkt die Anwendbarkeit eines Access Tokens auf einen bestimmten Absender. Dieser Absender ist verpflichtet, die Kenntnis eines bestimmten Geheimnisses als Voraussetzung für die Annahme dieses Tokens beim Empfänger (z. B. einem Resource Server) nachzuweisen.

Autorisierung- und Resource Server SOLLEN Mechanismen für Sender-Constraining Access Token verwenden, wie z. B. Mutual TLS for OAuth 2.0 [RFC8705] oder OAuth 2.0 Demonstrating Proof of Possession (DPoP)

[RFC9449] (siehe Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.10.1), um den Missbrauch gestohlener und durchgesickerter Access Token zu verhindern.

5.3.2. Refresh Token

Refresh Token für Public Clients MÜSSEN Sender-Constraining sein oder eine Refresh Token Rotation verwenden, wie in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.14 beschrieben. [RFC6749] schreibt bereits vor, dass Refresh Token für Confidential Clients nur von dem Client verwendet werden dürfen, für den sie ausgestellt wurden.

5.4. Beschränkung der Zugriffsrechte von Access Token

Die mit einem Access Token verbundenen Zugriffsrechte SOLLTEN auf das für die jeweilige Anwendung oder den jeweiligen Anwendungsfall erforderliche Minimum beschränkt werden. Dadurch wird verhindert, dass Clients die vom Resource Owner autorisierten Zugriffsrechte überschreiten. Außerdem wird dadurch verhindert, dass Benutzer ihre durch die jeweilige Sicherheitsrichtlinie autorisierten Zugriffsrechte überschreiten. Zugriffsbeschränkungen tragen auch dazu bei, die Auswirkungen von Access Token-Lecks zu verringern.

Insbesondere SOLLTEN Access Token auf einen bestimmten Resource Server oder, falls dies nicht möglich ist, auf eine kleine Gruppe von Resource Servern beschränkt sein. Um dies umzusetzen, ordnet der Authorization Server den Access Token bestimmten Resource Servern zu, und jeder Resource Server ist verpflichtet, bei jeder Anfrage zu überprüfen, ob der mit dieser Anfrage gesendete Access Token für diesen bestimmten Resource Server bestimmt war. Ist dies nicht der Fall, MUSS der Resource Server die entsprechende Anfrage ablehnen. Der in [RFC9068] definierte aud-Claim KANN verwendet werden, um Access Token auf bestimmte Zielgruppen zu beschränken. Clients und Authorization Server KÖNNEN die in [RFC6749] bzw. [RFC8707] angegebenen Parameter `scope` oder `resource` verwenden, um den Resource Server zu bestimmen, auf den sie zugreifen möchten.

Darüber hinaus SOLLTEN Access Token auf bestimmte Ressourcen und Aktionen auf Resource Server oder Ressourcen beschränkt sein. Um dies zu erreichen, verknüpft der Authorization Server den Access Token mit der jeweiligen Ressource und den jeweiligen Aktionen, und jeder Resource Server ist verpflichtet, bei jeder Anfrage zu überprüfen, ob der mit dieser Anfrage gesendete Access Token für diese bestimmte Aktion auf dieser bestimmten Ressource verwendet werden sollte. Ist dies nicht der Fall, muss der Resource Server die Bearbeitung der entsprechenden Anfrage ablehnen. Clients und Authorization Server KÖNNEN den in [RFC6749] angegebenen Parameter `scope` und den in [RFC9396] angegebenen Parameter `authorization_details` verwenden, um diese Ressourcen und/oder Aktionen zu bestimmen.

5.5. Client-Authentifizierung

Authorization Server SOLLEN die Client-Authentifizierung durchsetzen, wenn es in der jeweiligen Bereitstellung möglich ist, einen Prozess für die Ausstellung/Registrierung von Anmeldedaten für Clients einzurichten und die Vertraulichkeit dieser Anmeldedaten zu gewährleisten.

Es wird EMPFOHLEN, für die Client-Authentifizierung asymmetrische Kryptografie zu verwenden, wie z. B. Mutual TLS for OAuth 2.0 [RFC8705] oder signierte JWTs („Private Key JWT“) gemäß [RFC7521] und [RFC7523]. Letzteres ist in [OpenID.Core] als Client-Authentifizierungsmethode `private_key_jwt` definiert. Wenn asymmetrische Kryptografie für die Client-Authentifizierung verwendet wird, müssen Authorization Server

keine sensiblen symmetrischen Schlüssel speichern, wodurch diese Methoden robuster gegen das Durchsickern von Schlüsseln sind.

Wenn ein hoher Bedarf an die Sicherheit über die Identität des Clients im Sinne eines Confidential Clients bestehen, MÜSSEN die oben genannten Verfahren Client-Authentifizierungsmethode eingesetzt werden.

Browser basierte Clients (bspw. Single Page Apps) oder native Applikationen sind grundsätzlich als Public Clients im Sinne von OAuth zu bewerten und dürfen nicht als vertrauenswürdig anzusehen, außer wenn entsprechende Architekturmuster und Frist Party Sicherheitsmechanismen genutzt. Entsprechende Empfehlungen sind dem Begleitdokument „Leitfaden für den Umgang mit Public Clients“ *(noch nicht fertigstellt)* zu entnehmen, der entsprechende Absicherungsmaßnahmen aufzeigt, mit dem aktuell oder perspektivisch Browser basierte Clients oder native Applikationen als Confidential Clients behandelt werden können.

5.6. TLS Absicherung von Netzwerkverbindungen

Bei Verbindungen zwischen dem Client, Resource Server und Authorization Server MÜSSEN TLS Absicherungen gemäß [BSI TR-02102-2] verwendet werden. Insbesondere Autorisierungsantworten DÜRFEN NICHT über unverschlüsselte Netzwerkverbindungen übertragen werden. Zu diesem Zweck DÜRFEN Authorization Server keine Redirection URI zulassen, die das http-Schema verwenden, mit Ausnahme von nativen Clients, die eine Loopback Interface Redirection gemäß Kapitel 7.3 von [RFC8252] verwenden.

Es wird EMPFOHLEN, zwischen dem Client und dem Resource Server End-to-End-TLS gemäß den oben genannten Vorgaben zu verwenden. Wenn der TLS-Verkehr bei einem Vermittler wie bspw. Einem Reverse Proxy beendet werden muss, finden Sie weitere Sicherheitshinweise im Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“.

5.7. Weitere Empfehlungen

Die Verwendung von OAuth-Authorization Server Metadaten [RFC8414] kann dazu beitragen, die Sicherheit von OAuth-Implementierungen zu verbessern:

- Es stellt sicher, dass Sicherheitsfunktionen und andere neue OAuth-Funktionen automatisch durch kompatible Softwarebibliotheken aktiviert werden können.
- Sie verringert das Risiko von Fehlkonfigurationen – beispielsweise falsch konfigurierte Endpunkt-URLs (die möglicherweise einem Angreifer gehören) oder falsch konfigurierte Sicherheitsfunktionen.
- Sie kann dazu beitragen, die Rotation von kryptografischen Schlüsseln zu erleichtern und kryptografische Agilität zu gewährleisten.

Es wird daher EMPFOHLEN, dass Authorization Server OAuth-Authorization Server Metadaten gemäß [RFC8414] veröffentlichen und dass Clients diese Authorization Server Metadaten (sofern verfügbar) zur Konfiguration nutzen.

Unter den in Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.15.1 beschriebenen Bedingungen SOLLEN Authorization Server Clients nicht erlauben, ihre `client_id` oder andere Angaben zu beeinflussen, die zu Verwechslungen mit einem echten Resource Owner führen könnten.

Wenn die Autorisierungsantwort mit browserinternen Kommunikationstechniken wie postMessage [WHATWG.postmessage_api] anstelle von HTTP Redirect gesendet wird, MÜSSEN sowohl der Initiator als auch

der Empfänger der browserinternen Nachricht streng gemäß Begleitdokument „Angreifer Model und Schutzmaßnahmenkatalog für den Schutzbedarf Normal“, Kapitel 4.17 überprüft werden.

Um browserbasierte Clients zu unterstützen, KÖNNEN Endpunkte, auf die solche Clients direkt zugreifen, einschließlich des Token-Endpunkts, des Authorization Server Metadaten-Endpunkts, des Endpunkts `jwks_uri` und des dynamischen Client-Registrierungs-Endpunkts, die Verwendung von Cross-Origin Resource Sharing (CORS) [WHATWG.CORS] unterstützen. CORS DARF jedoch nicht am Autorisierungsendpunkt unterstützt werden, da der Client nicht direkt auf diesen Endpunkt zugreift, sondern stattdessen den User Agent dorthin umleitet.

6. Verweise

6.1. Normative Verweise

[BSI TR-02102-2]

BSI TR-02102-2 "Kryptographische Verfahren: Verwendung von Transport Layer Security (TLS)" Version: 2025-1, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf?__blob=publicationFile&v=11.

[DIN 820-2]

Normungsarbeit - Teil 2: Gestaltung von Dokumenten (ISO/IEC Directives - Part 2:2021, modifiziert); Deutsche und Englische Fassung CEN-CENELEC-Geschäftsordnung - Teil 3:2022, https://www.dinmedia.de/de/norm/din-820-2/358748335.

[draft-ietf-oauth-v2-1-14]

Hardt, D., Parecki, A., Lodderstedt, T., "OAuth 2.1 Authorization Framework", Version 14, 19. Oktober 2025, https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/.

[ISO29100]

ISO/IEC, "ISO/IEC 29100 Information technology – Security techniques – Privacy framework", https://www.iso.org/standard/85938.html.

[OIDC]

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1", 8 November 2014, http://openid.net/specs/openid-connect-core-1_0.html

[RFC6749]

Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, https://www.rfc-editor.org/info/rfc6749.

[RFC6750]

Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, https://www.rfc-editor.org/info/rfc6750.

[RFC6819]

Lodderstedt, T., Ed., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", RFC 6819, DOI 10.17487/RFC6819, January 2013, https://www.rfc-editor.org/info/rfc6819.

[RFC7521]

Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, https://www.rfc-editor.org/info/rfc7521.

[RFC7523]

Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, https://www.rfc-editor.org/info/rfc7523.

[RFC8252]

Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, https://www.rfc-editor.org/info/rfc8252.

[RFC8414]

Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, https://www.rfc-editor.org/info/rfc8414.

[RFC8705]

Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, https://www.rfc-editor.org/info/rfc8705.

[RFC9068]

Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", RFC 9068, DOI 10.17487/RFC9068, October 2021, https://www.rfc-editor.org/info/rfc9068.

[RFC9449]

Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, https://www.rfc-editor.org/info/rfc9449.

[RFC9700]

Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "Best Current Practice for OAuth 2.0 Security", BCP 240, RFC 9700, DOI 10.17487/RFC9700, January 2025, https://www.rfc-editor.org/info/rfc9700.

6.2. Informative Verweise

[OpenID.Core]

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 2", The OpenID Foundation, 15 December 2023, https://openid.net/specs/openid-connect-core-1_0.html.

[OpenID.JARM]

Lodderstedt, T. and B. Campbell, "Financial-grade API: JWT Secured Authorization Response Mode for OAuth 2.0 (JARM)", The OpenID Foundation, 17 October 2018, https://openid.net/specs/openid-financial-api-jarm.html.

[RFC7636]

Sakimura, N., Ed., Bradley, J., and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", RFC 7636, DOI 10.17487/RFC7636, September 2015, https://www.rfc-editor.org/info/rfc7636.

[RFC8707]

Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707, DOI 10.17487/RFC8707, February 2020, https://www.rfc-editor.org/info/rfc8707.

[RFC9207]

Meyer zu Selhausen, K. and D. Fett, "OAuth 2.0 Authorization Server Issuer Identification", RFC 9207, DOI 10.17487/RFC9207, March 2022, https://www.rfc-editor.org/info/rfc9207.

[RFC9396]

Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, https://www.rfc-editor.org/info/rfc9396.

[WHATWG.CORS]

WHATWG, "CORS protocol", Fetch: Living Standard, Section 3.2, 17 June 2024, https://fetch.spec.whatwg.org/#http-cors-protocol.

[WHATWG.postmessage_api]

WHATWG, "Cross-document messaging", HTML: Living Standard, Section 9.3, 19 August 2024, https://html.spec.whatwg.org/multipage/web-messaging.html#web-messaging.

Anhang A. Hinweise zu Rechten am Dokument

Das vorliegende Dokument ist ein aus dem ‚Best Current Practice for OAuth 2.0 Security' abgeleitetes Dokument, weshalb die dafür genannten Copyright Regeln der IETF zu beachten sind:

„Copyright (c) 2025 IETF Trust

Dieses Dokument unterliegt BCP 78 und den rechtlichen Bestimmungen des IETF Trust in Bezug auf IETF-Dokumente (https://trustee.ietf. org/license-info) in der zum Zeitpunkt der Veröffentlichung dieses Dokuments gültigen Fassung. Bitte lesen Sie diese Dokumente sorgfältig durch, da sie Ihre Rechte und Einschränkungen in Bezug auf dieses Dokument beschreiben. Aus diesem Dokument extrahierte Code-Komponenten müssen den in Abschnitt 4.e der rechtlichen Bestimmungen des Trusts beschriebenen Text der überarbeiteten BSD-Lizenz enthalten und werden ohne Gewährleistung gemäß der überarbeiteten BSD-Lizenz bereitgestellt."

Anhang B. Wesentliche Unterschiede zu BCP OAuth 2.0 Security [RFC9700]

Kapitel	**"Best Current Practice for OAuth 2.0 Security**	**Dieses Dokument**	**Gründe**
Allg.		Dieses Dokument basiert nur auf Kapitel 2 des Originals. Das BCP Attacker Modell aus Kapitel 3 und 4 wurde ins Begleitdokument ausgelagert.	Attacker Model und Details ausgelagert, wie bei Schutzbedarf Sehr Hoch und Hoch.
Allg.	OAuth 2.0	OAuth 2.0 und 2.1	OAuth 2.1 [draft-ietf-oauth-v2-1-14] ergänzt
Schlüssel-wörter		DIN Norm	Deutsch
5.2.1.		Für alle Client Typen ist PKCE verpflichtend.	
5.2.1.	Kapitel 2.1.2. Implicit Grant	Kapitel nicht enthalten.	Wird verboten, daher ist nur der Einleitungstext wichtig.
5.5.		Ergänzende MUSS Anforderungen, wenn Confidential Clients von hoher Bedeutung sind.	
5.6.		Kapitel neu. Der zweite Absatz zu End-to-End Absicherung wurde aus dem Abschnitt weitere Empfehlungen herausgenommen.MUSS Anforderungen im ersten Satz mit Verweis auf BSI TR-02102 ist komplett neu. Satz zwei wurde ebenfalls aus dem weiteren Empfehlungen genommen.	

Kapitel	"Best Current Practice for OAuth 2.0 Security	Dieses Dokument	Gründe
6.		Es sind nur Referenzen aufgeführt, die auch in diesem Dokument verwendet werden.	