SEEBURGER

Whitepaper | SEEBURGER BIS API-Funktionen

API-Sicherheit – Eine kurze Einführung

Inhalt

APIs heute und morgen **3**

Die Auswirkungen einer schlechten Implementierung von API-Sicherheitsmaßnahmen **4**

Informationen und Daten, die unsichere APIs liefern **5**

Mechanismen und Technologien zum Schutz Ihrer Daten **6**

Auth (n/z) 6

OAuth2.0 6

JSON Web Token 7

Dreh- und Angelpunkt der API-Sicherheit: Das API-Gateway **8**

Das API-Gateway der SEEBURGER BIS Plattform **9**

Policy-Architektur **10**

Rate-limit-by-key 10

Validate-jwt 11

Aktuelle Risiken für die API-Sicherheit **12**

APIs heute und morgen

APIs sind nach wie vor der häufigste Angriffsvektor für Unternehmen und ihre Daten. Dies belegen umfassende Marktanalysen und Erkenntnisse zahlreicher Branchenanalysten. Durch die rasant zunehmende Nutzung von APIs sind daher robuste und zuverlässige Sicherheitsmaßnahmen wichtiger denn je, damit die Systeme und vertraulichen Daten Ihres Unternehmens vor potenziellen Angriffen geschützt sind. Nur durch proaktive Maßnahmen kann die ständig wachsende API-Landschaft zuverlässig vor Cyberangriffen und verbundene Risiken geschützt werden.

Was ist API-Sicherheit und warum ist sie so wichtig?

Mit dem Begriff API-Sicherheit werden sämtliche Prozesse und Richtlinien beschrieben, die dem Schutz von APIs vor Cyberangriffen, unbefugtem Zugriff und Datenschutzverletzungen dienen. APIs dienen dem Datenaustausch und der Interaktion zwischen Software-Systemen und werden für die reibungslose Kommunikation zwischen Anwendungen eingesetzt. Da APIs mittlerweile breite Anwendung in digitalen Ökosystemen finden, ist es essenziell, ihre Sicherheit umfassend zu gewährleisten.

API-Sicherheit wird zunehmend wichtiger – zu den Hauptgründen zählen:

Rasche Zunahme der API-Nutzung: APIs werden von vielen Branchen in hohem Maße genutzt, wodurch sich die Angriffsfläche für mögliche Bedrohungen deutlich vergrößert hat. Es wird immer wichtiger, diese Schnittstellen zu sichern, da immer mehr Anwendungen beim Austausch von Funktionen und Daten auf sie angewiesen sind.

Sensibilität der Daten: Über APIs werden viele sensible Daten wie Finanz-, Anwender- und geschäftskritische Daten verarbeitet. Der unbefugte Zugriff auf sensible Daten kann schwerwiegende Folgen haben, darunter die Verletzung der Privatsphäre, finanzielle Verluste und die Schädigung des Rufs eines Unternehmens.

Drohende Zunahme von Cyberangriffen: Die sich ständig wandelnde Bedrohungslage hinsichtlich Cybersicherheit stellt Unternehmen vor erhebliche Herausforderungen. Böswillige Akteure entwickeln ständig neue, ausgefeilte Methoden zur Ausnutzung von API-Schwachstellen, so dass Unternehmen durch die Einführung strenger Sicherheitsmaßnahmen immer einen Schritt voraus sein müssen.

Einhaltung von Vorschriften: Das Aufkommen von Datenschutzgesetzen wie GDPR, HIPAA und anderen erfordert strengere Kontrollen bei der Verarbeitung und Übertragung sensibler Daten. Unternehmen müssen die API-Sicherheit gewährleisten, um diesen Vorschriften zu entsprechen und sich vor finanziellen und rechtlichen Konsequenzen zu schützen.

Business Continuity: APIs sind für den Betrieb von Geschäftsprozessen und deren Funktionieren von wesentlicher Bedeutung. Dabei kann jeder Sicherheitsverstoß zur Unterbrechung des Betriebes führen, was nicht nur den Verlust hoher Geldsummen und beträchtlichen Zeitaufwand zur Folge haben kann. Häufig wird hierdurch sogar der Ruf eines Unternehmens nachhaltig geschädigt.

Um diese Themen anzugehen, investieren Unternehmen in bewährte Verfahren und API-Sicherheitslösungen wie Verschlüsselung, Zugriffskontrollen, Authentifizierungsmethoden und regelmäßige Sicherheitsaudits. In einer Welt, die immer vernetzter und datengesteuerter wird, schützen proaktive API-Sicherheitsmaßnahmen nicht nur vor möglichen Bedrohungen, sondern stärken auch die allgemeine Belastbarkeit der digitalen Infrastruktur.

Die Auswirkungen einer schlechten Implementierung von API-Sicherheitsmaßnahmen

Einer der bekanntesten Fälle dafür, was bei mangelhafter API-Sicherheit mit den gesammelten Daten passieren kann, ist die öffentliche API von Venmo. Venmo gehört zu PayPal und ist Anbieter mobiler Zahlungsdienstleistungen. Nach Anlegen eines Nutzerkontos können mithilfe der mobilen App Überweisungen an andere Nutzer getätigt werden. Bei der Gründung des Unternehmens lag der Schwerpunkt auf dem Geldverkehr zwischen Freunden, die sich eine Rechnung teilten, z. B. bei einem gemeinsamen Kinobesuch oder einem Abendessen. Um Venmo zu nutzen, muss ein Konto erstellt werden, und sowohl der Sender als auch der Empfänger müssen in den USA leben.


```
{"payment_id": 2141499719669514525, "permalink": "", "via": "",
"transactions": [{"target": {"username": "JohnD", "picture": "",
"is_business": false, "name": "Doe", "firstname": "John", "lastname": "D",
"cancelled": false, "date_created": "2020-02-02T16:57:40", "external_id":
"2141499719669514525", "id": "2141499719669514525"}}], "story_id":
"2141499719669514525", "comments": [], "updated_time": "2020-02-
02T16:57:40", "audience": "public", "actor": {"username": "JaneD",
"picture": "", "is_business": false, "name": "Doe", "firstname": "D",
"lastname": "D", "cancelled": false, "date_created": "2020-02-02T16:57:40",
"external_id": "2141499719669514525", "id": "2141499719669514525"}, "type":
"payment", "created_time": "2020-02-02T16:57:40", "mentions": [],
"message": "Miss you", "action_links": {}, "likes": {"count": 0, "data":
[]}},
```


Bei der Erstellung des Benutzerkontos, was entweder über die mobile App oder die Website erfolgen kann, müssen grundlegende Informationen eingegeben werden. Dazu gehören Bankkontodaten, Debitkarten, Kreditkarten-informationen, ein Benutzername, Telefonnummern und/oder E-Mail. Über diese Daten kann der Empfänger dann gefunden werden.

Venmo verwendet eine öffentliche API. Sofern dies nicht in den Standardeinstellungen geändert wird, sind alle Transaktionen zwischen zwei Personen öffentlich für alle einsehbar. Eine Transaktion enthält Vor- und Nachname, Profilfoto, eine entsprechende Nachricht, und die Facebook-ID, das Transaktionsdatum sowie den Status der Transaktion – all dies konnte eingesehen werden.

Die Sicherheitsforscherin Hang Do Thi Duc nutzte diese Möglichkeit im Jahr 2018 und wertete alle Datensätze aus dem Jahr 2017 aus. Die API lieferte ein breites Spektrum an Informationen. Durch die Analyse und Auswertung dieser Daten war es möglich, detaillierte Profile über Personen zu erstellen, einschließlich potenzieller Gesundheitsprobleme.

Dies gelang ihr, weil die öffentliche API private Informationen bereitstellt und nicht sicher konfiguriert war, und weil Datenfilter, Authentifizierungs- und Autorisierungs-mechanismen für den Abruf von Daten nicht verfügbar waren. Es gab zudem weitere schwerwiegende Sicherheits-fehler, die in der OWASP API Security Top 10 am Ende des Papers aufgeführt sind.

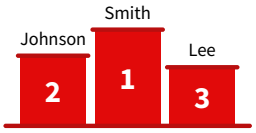
Informationen und Daten, die unsichere APIs liefern:

Nicht auszudenken, es handelte sich hierbei um Ihre wichtigen Unternehmensdaten.
Die Grafik illustriert alle Informationen, die Hang Do Thi Duc dank der unsicheren Venmo-API nutzen konnte, anschaulich.

1.731.783
Facebook-IDs

207.984.218
Transaktionen

18.429.464
Personen



Häufigste
Nachnamen

Meist frequentiertes Wochenende
1-3. Dezember 2017
2.342.411 Transaktionen

2.979.616
Transaktionen für Pizza

Profile

"YOLOist"

- Weiblich mit griechischem Namen
- Die Freunde leben in Texas und Mexico City
- 865 Transaktionen für Soda, Alkohol, Fast Food und Süßigkeiten in 8 Monaten
- Nutzt 1-2 Wörter oder ein Emoji für die Beschreibung von Dingen
- Folgt Gewohnheiten

Der amerikanische Jedermann

- Lebt als Paar
- Besitzt ein Auto und einen Hund
- Bevorzugt Shakey's Pizza
- Lebt in Orange County, CA
- Kauft Markenartikel
- Geht ins Theater
- Kauft am liebsten bei Walmart
- Geht mehr als einmal die Woche einkaufen, jedoch nie mittwochs oder sonntags

Mechanismen und Technologien zum Schutz Ihrer Daten

Auth (n/z)

Auth (n/z) steht für Authentifizierung und Autorisierung. Authentifizierung und Autorisierung sind weit gefasste Begriffe, die mehrere Ansätze zur Sicherung, Identifizierung und Durchsetzung der Rechteverwaltung beinhalten. So geht es im Kontext von Authentifizierung um die Identität des Anwenders. Zur Authentifizierung werden Mechanismen eingesetzt, die zwei Fragen beantworten: Ist die Person diejenige, die sie vorgibt zu sein, und wer ist die Person? Autorisierung hingegen stellt die Frage, wozu ist diese Person befugt? Sie entscheidet, welche Daten die aufrufende Instanz einsehen darf.

Nehmen Sie als praktisches Beispiel eine Reise mit dem Zug oder dem Flugzeug. Beim Check-in wird der Personalausweis vorgezeigt, der die Person authentifiziert, da er eindeutig zeigt, wer die Person ist. Das Flug- oder Bahnticket stellt die Berechtigungsmarke dar. Es gibt an, was die Person tun darf. In diesem Fall zeigt es an, in welchem Flugzeug sie sich befindet und wo die Person sitzen darf. Innerhalb von APIs gibt auth (n/z) an, wer die entsprechende API aufruft und welche Rechte diese Person hat, d. h. was diese Person tun darf oder welche Daten eine aufrufende Instanz sehen darf.

> “Hier ist mein Ausweis.
> Hallo, ich bin Max Mustermann.”
>
> Authentifizierung

> “Hier ist mein Flugticket.
> Ich bin befugt, auf diesem Platz zu sitzen.”
>
> Autorisierung

OAuth2.0

OAuth2.0[1] stellt den Standard für die Autorisierung einer dritten Partei dar, um Zugang zu Ressourcen zu erhalten, die einer anderen Person gehören – RFC 6749. OAuth2.0 ermöglicht Websites oder Anwendungen den Zugriff auf die Daten einer anderen Person, ohne dass das eigentliche Passwort an diese weitergeleitet wird.

OAuth2.0 hat vier Rollen:

01 Ressourceneigentümer:
Eigentümer der angeforderten Ressource,
der dem Client die Nutzung der Ressource gestattet.

02 Client:
Die anfordernde Instanz, die eine Ressource nutzen möchte.

03 Ressourcenserver:
Der Ressourcenserver hostet die geschützte Ressource,
die vom Client angefordert wurde
und dem Ressourceneigentümer gehört.

04 Autorisierungsserver:
Identifiziert und authentifiziert den Benutzer
und gibt Zugriffstoken an den Client aus.

Ein bekanntes OAuth2.0-Beispiel zeigt die Verbindung zwischen Pinterest und Facebook. Innerhalb der Online-Pinnwand Pinterest können Nutzer ihre eigenen Facebook-Freunde als Kontakte hinzufügen. Zu diesem Zweck muss Pinterest auf die Facebook-Informationen des Nutzers zugreifen, ohne jedoch die Facebook-Login-Daten des Nutzers zu erhalten. An dieser Stelle kommt OAuth2.0 ins Spiel.

Dem Nutzer (Ressourcenbesitzer) wird von Pinterest (Client) ein Fenster für die Anmeldung bei Facebook angezeigt. Der Benutzer meldet sich an und gewährt Pinterest Zugriff. Pinterest kann nun ein Zugriffstoken vom Authorization Server von Facebook anfordern und mit Hilfe dieses Tokens die Ressourcen – in diesem Fall die Kontakte des Nutzers – vom Resource Server anfordern. Pinterest erhält nun die Ressource als geschützte Ressource. Pinterest hat nur die Rechte, die es angefordert hat und die der Ressourceneigentümer während des gesamten Prozesses genehmigt hat, und weiß nichts von den Anmeldedaten des Nutzers auf Facebook.

Genehmigungen umfassen

+ Autorisierungscode
+ Implizit
+ Informationen zum Kennwort des Ressourcenbesitzers
+ Client-Anmeldeinformationen

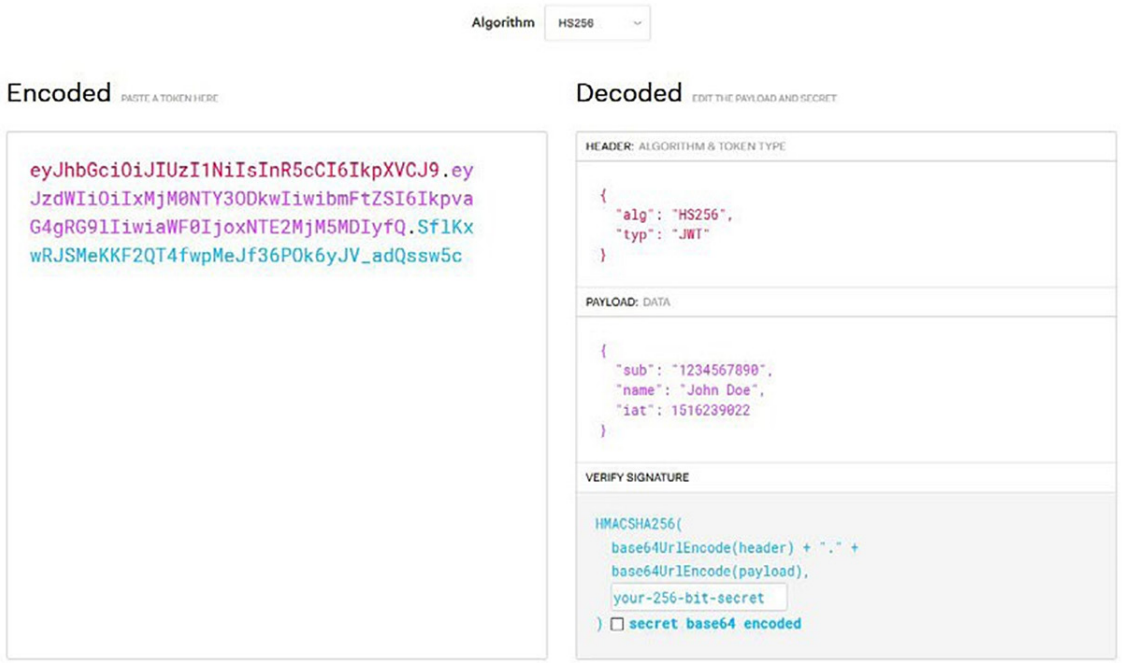
```
+--------+                                +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

1 https://tools.ietf.org/html/rfc6749

JSON Web Token

Ein JSON Web Token (JWT, ausgesprochen /dʒɒt/) ist ein Zugriffstoken, das JSON als Grundlage verwendet. Es kann innerhalb der OAuth2.0-Ströme als Zugriffstoken verwendet werden, das der Autorisierungsserver dem Client gewährt. Das Token ist ein String, der in der Kopfzeile oder als Anfrageparameter übergeben wird. Der Informationsaustausch zwischen zwei Parteien erfolgt anhand der im JWT enthaltenen verifizierten Angaben. Als offener Standard (RFC 7519) bietet JWT eine sichere und kompakte Methode, um diese Informationen auszutauschen. Die Informationen im JWT sind digital signiert und daher vertrauenswürdig. JWT ist eine Art Container, der die Form der Informationen festlegt, die hin- und hergeschickt werden.

Ein JSON-Web-Token besteht immer aus drei Teilen, die in der folgenden Abbildung durch die drei Farben rot, rosa und blau gekennzeichnet sind.



Header (rot)

Die Kopfzeile gibt an, welcher Algorithmus zum Signieren des Tokens verwendet wurde. In diesem Beispiel ist es HMAC SHA256, das durch „alg“ dargestellt wird: „HS256“. Der Token-Typ, JSON Web Token, ist ebenfalls angegeben. Die im JSON-Format gespeicherten Daten sind Base64-kodiert.

Payload (pink)

Der Payload stellt die eigentlichen Informationen im JSON-Format dar. Die Key/Value-Paare, die die Informationen darstellen, werden als Claims bezeichnet. Der Payload im Beispiel enthält die Informationen „sub“, „name“ und „iat“. Genau wie der Header ist auch der Payload Base64-kodiert.

Signatur (blau)

Die Signatur wird aus dem Header abgeleitet und nach dem in RFC 7515 standardisierten Algorithmus definiert. Danach wird die Signatur – genau wie Header und Payload – Base64-encodiert.

2 https://tools.ietf.org/html/rfc7519

Dreh- und Angelpunkt der API-Sicherheit: Das API-Gateway

Ein API-Gateway ist der zentrale Punkt für die Implementierung von Sicherheitsaspekten und deren Überwachung. Innerhalb des Gateways können Sicherheitsthemen wie Auth (n/z) angesprochen werden. Ein API-Gateway schützt vor Verletzungen der Datensicherheit.

Das Gateway dient als eine Art Torwächter, der den Zugang zu den Daten bewacht. Es befindet sich zwischen den Clients und dem eigentlichen Service und stellt den einzigen Zugangspunkt dar.

Die Aufgaben:

+ Schafft Transparenz im API-Datenverkehr
+ Ermöglicht es der IT-Abteilung, die vom Markt und den Industrie gestellten API-Anforderungen zu kontrollieren und zu erfüllen
+ Schützt exponierte Dienste
+ Sichert Daten
+ Gewährleistet die zuverlässige Verarbeitung jedes API-Aufrufs

Um diese Aufgaben zu erfüllen, stehen verschiedene Mechanismen, wie die Umsetzung von Richtlinien zur Drosselung und Authentifizierung, zur Verfügung.

API-Gateways sorgen für Transparenz im API-Datenverkehr und ermöglichen die Erfüllung der vom Markt und der Industrie festgelegten API-Anforderungen.

Das API-Gateway der SEEBURGER BIS Plattform

Das Full-Lifecycle-API-Management, wie es die API-Funktion der SEEBURGER BIS Plattform bietet, hilft bei der Erfassung und Kontrolle der genutzten und bereitgestellten APIs. Eine der wesentlichen Komponenten des API-Managements ist das API-Gateway, das auch im Bereich der API-Sicherheit eine wichtige Rolle spielt.

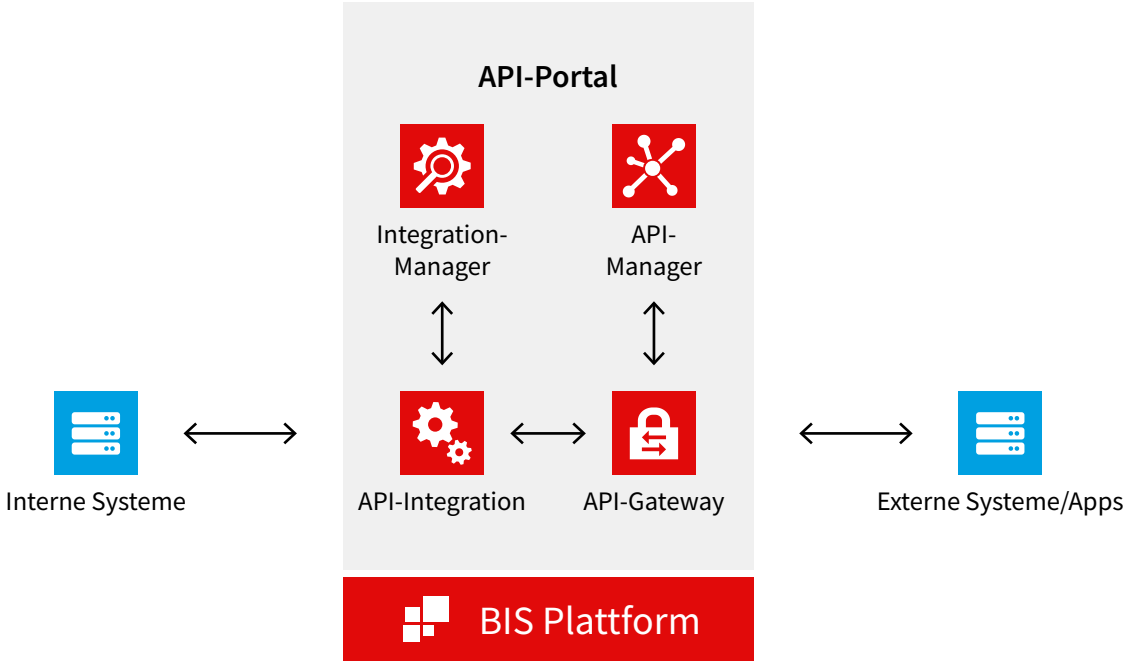
Das BIS API-Gateway ist ein bereits vorkonfigurierter, integrierter Bestandteil des BIS API-Managements. Der gesamte API-Datenverkehr läuft über das Gateway, das den zentralen Punkt der Lösung darstellt. Der Einsatz des API-Gateways erhöht die Transparenz, steuert den Datenfluss, trägt zum Schutz der bereitgestellten Services bei, sichert die Daten und sorgt für eine zuverlässige Verarbeitung jedes API-Aufrufs.

Obwohl mehrere Gateways parallel oder nacheinander eingesetzt werden können, werden sie nicht einzeln sondern zentral über eine einzige GUI – den BIS API-Manager – verwaltet. Mit Hilfe des BIS Landscape-Managers lassen sich das/die Gateway(s) installieren und je nach Bedarf manuell, teilweise oder vollautomatisch aktualisieren.

Die folgende Abbildung zeigt den Datenfluss zwischen aufrufenden Instanzen sowie öffentlichen und privaten APIs über zwei Gateways, die zentral vom BIS API-Manager verwaltet werden.

Folgende Funktionen werden unterstützt:

+ Parallelbetrieb mehrerer Instanzen (Active / Active) für Skalierung, Hochverfügbarkeit und Updates ohne Ausfallzeiten.
+ User-Exit für den Fall, dass eine vom SEEBURGER-Standard abweichende Individualisierung erforderlich ist.
+ Umfassende vordefinierte Richtlinien, die mit den Funktionen des Policy Designers im Developer Studio leicht erweitert werden können.
+ Die Funktionen des BIS API-Gateways basieren auf dem http-Adapter-Layer.
+ Speicherinterne Channel-Verarbeitung.
+ Minimale Latenz.



Policy-Architektur

Policies sind Regeln, die dazu dienen, die zugrundeliegenden Services zu gestalten und zu schützen und somit das API-Verhalten zu steuern. Policies können verschiedenen Gruppen zugeordnet werden, z. B. dem Traffic-Management oder der Sicherheit. Die Struktur einer Policy ist immer die gleiche mit mindestens einer zu prüfenden Regel und der daraus resultierenden auszuführenden Aktion. All dies geschieht innerhalb des API-Gateways, das für die Umsetzung der Policy verantwortlich ist.

So prüft das BIS API-Gateway Policies auf Basis der Informationen der http-Anfrage:

+ Caller IP
+ URL, bestehend aus Protokoll, Host, Port, Pfad, Abfrageparameter (http://petstore.swagger.io:80/v2/pet/5?name=dog)
+ Header = einem Set aus Key/Value-Paaren
+ Inhalt
  + REST: Inhalt = Payload
  + WebService definiert SOAP-Inhalt mit zusätzlichem SOAP-Header, Payload ist enthalten
+ AS2 verwendet S/mime mit zusätzlichen Headern, Payload enthalten

Rate-limit-by-key

Es ist wichtig, die Anzahl der Aufrufe einer API innerhalb einer bestimmten Zeitspanne zu begrenzen, z. B. zum Schutz der Backend-Systeme, die nur eine bestimmte Anzahl von Anfragen in einer bestimmten Zeitspanne verarbeiten können, oder zum Schutz vor DoS-Angriffen.

```
rate-limit-by-key {
    @calls := 10
    @renewal-period := "M1"
    @refresh-increment := 60
    @counter-key := "key-to-limit-call"
}
```

Parameter	Bechreibung
Calls	Die maximale Anzahl von Aufrufen, die durchgeführt werden können, bevor 429 “too many requests” zurückgegeben wird. Überprüft den Zählerdienst, um festzustellen, ob der Zähler für den angegebenen Schlüssel das Maximum überschritten hat oder nicht.
Renewal-period	Erneuerungszeitraum, in dem der Zählerdienst den Zähler dekrementiert. Mögliche Werte: • “S10” – 10 Sekunden • “S30” – 30 Sekunden • “M1” – 1 Minute • “M10” – 10 Minuten • “M30” – 30 Minuten • “H1” – 1 Stunde
Refresh-increment	Übersteigt die Anzahl der Anrufe die zulässigen Grenzen, werden alle weiteren Anrufe blockiert, bis der Verlängerungszeitraum abgelaufen ist. Danach wird der Zählerwert um den in refresh-increment eingestellten Wert verringert.
Counter-key	Der Name des Keys, der zur Begrenzung der Anzahl der Aufrufe verwendet wird. Bei jedem Aufruf wird ein Zähler für den angegebenen Schlüssel erhöht.

Validate-jwt

Wie bereits erwähnt, sind JWTs eine gängige Art von Zugriffstoken, die Benutzerinformationen und Berechtigungen enthalten. Es ist wichtig, jedes JWT zu validieren. Wenn die Validierung fehlschlägt, wird eine Force Reply gegeben. Andernfalls wird die Anfrage einfach an die nächste Gruppe von Richtlinien weitergeleitet.

```
validate-jwt {
    @header-name := "some-header-name"
    @failed-validation-httpcode := 409
    @failed-validation-error-message := "Access Denied"
    @token-value := "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0
    @require-expiration-time := "false"
    @require-signed-tokens := "true"
    issuer-signing-keys {
        key := "secret"
    }
    audiences {
        audience := "test-audience"
    }
    issuers {
        issuer := "test-issuer"
    }
}
```

Parameter	Beschreibung
Header-name	Der Name der Kopfzeile, die das JWT-Token enthält. Wenn das Attribut „token-value“ bereits gesetzt ist, wird dies ignoriert.
Failed-validation- httpcode	Der http-Code, den der Benutzer als Antwort erhält, wenn die Validierung fehlschlägt.
Failed-validation- error-message	Die Nachricht, die der Benutzer als Antwort erhält, wenn die Validierung fehlschlägt
Token-value	Das JWT
Require-expiration-time	“true” oder “false”. Wird dies auf “false” gesetzt, bedeutet dies, dass es erlaubt ist, ein abgelaufenes JWT zu nutzen.
Require-signed-tokens	“true” oder “false”. Wird dies auf “false” gesetzt, bedeutet dies, dass das JWT-Token nicht mit dem „signing-key“ validiert wird.
Issuer-signing-keys	Enthält alle Keys, die zur Validierung des JWT-Tokens verwendet werden. Sie können mehrere Keys hinzufügen, die nacheinander validiert werden, bis der richtige Key gefunden wird oder alle Keys getestet wurden
Audience	Listet die erwarteten Zielgruppen (Audiences) auf. Das Token wird validiert, um festzustellen, ob die erwarteten Zielgruppen mit den Zielgruppen des Tokens übereinstimmen.
Issuers	Listet die erwarteten Emittenten (Issuers) auf. Der Token wird daraufhin überprüft, ob der Emittent des Tokens in der Liste der erwarteten Emittenten gefunden werden kann.

Aktuelle Risiken für die API-Sicherheit

Die in diesem Dokument enthaltenen Informationen über Bedrohungen und Schutzmechanismen sind keineswegs vollständig. Die Möglichkeiten eines Angriffs sind mindestens so vielfältig wie die möglichen Schutzmaßnahmen. Wenn es um die Sicherheit von Webanwendungen und APIs geht, ist das Open Web Application Security Project (OWASP) maßgebend. OWASP ist eine Gemeinschaft, die ihr erstes Sicherheitsdokument im Jahr 2003 veröffentlichte. Seitdem decken Dokumente, Tools, Videos und das OWASP-Forum eine Vielzahl von Sicherheitsthemen ab. Im Jahr 2019 hat OWASP begonnen, sich auf spezifische Themen im Bereich der API-Sicherheit zu konzentrieren, und hat die 10 wichtigsten Sicherheitsrisiken für das Jahr 2021 aktualisiert:

Bedrohung	Beschreibung
A01:2021 – Broken Access Control	Komplexe Policies zur Zugriffskontrolle mit unterschiedlichen Hierarchien, Gruppen und Rollen sowie einer unklaren Trennung zwischen administrativen und regulären Funktionen führen häufig zu Autorisierungsfehlern (Broken Access Control). Durch Ausnutzung dieser Schwachstellen erhalten Angreifer Zugriff auf die Ressourcen anderer Benutzer und/oder administrative Funktionen.
A02:2021 – Cryptographic Failures	Ursprünglich unter dem Begriff “Sensitive Data Exposure” bekannt, bei dem es sich eher um ein weit gefasstes Symptom als um die eigentliche Ursache handelt, liegt der Schwerpunkt jetzt auf kryptographischen Fehlern (Cryptographic Failures) (oder deren Fehlen), die häufig zur Offenlegung sensibler Daten führen.
A03:2021 – Injection	Injection-Fehler, wie SQL, NoSQL, Command Injection usw., treten auf, wenn nicht vertrauenswürdige Daten als Teil eines Befehls oder einer Abfrage an einen Interpreter gesendet werden. Die böswilligen Daten des Angreifers können den Interpreter dazu verleiten, unbeabsichtigte Befehle auszuführen oder auf Daten ohne entsprechende Berechtigung zuzugreifen.
A04:2021 – Insecure Design	Der Begriff “insecure design” umfasst verschiedene Schwachstellen, die als “fehlendes oder unwirksames Kontrolldesign” bezeichnet werden. Es besteht ein Unterschied zwischen unsicherem Design und unsicherer Implementierung, da sie unterschiedliche Ursachen und Gegenmaßnahmen beinhalten (z. B. das Versäumnis, die erforderliche Sicherheitsstufe festzulegen).
A05:2021 – Security Misconfiguration	Fehlerhafte Sicherheitskonfigurationen (Security Misconfiguration) sind häufig das Ergebnis unsicherer Standardkonfigurationen, unvollständiger oder Ad-hoc-Konfigurationen, offener Cloud-Speicher, falsch konfigurierter HTTP-Header, unnötiger HTTP-Methoden, zulässiger herkunftsübergreifender Ressourcenfreigabe (“Cross-Origin Resource Sharing”, CORS) und weitschweifiger Fehlermeldungen mit vertraulichen Informationen.
A06:2021 – Vulnerable and Outdated Components	Unbekannte Versionen aller verwendeten Komponenten, nicht unterstützte oder veraltete Software, das Versäumnis, regelmäßig nach Schwachstellen zu suchen und die zugrundeliegenden Plattformen, Frameworks und Abhängigkeiten zu fixen oder zu aktualisieren, das Versäumnis, die Kompatibilität aktualisierter, weiterentwickelter oder gepatchter Bibliotheken zu testen oder die Konfigurationen der Komponenten zu sichern - all dies sind Beispiele für anfällige und veraltete Komponenten (Vulnerable and Outdated Components).

Bedrohung	Beschreibung
A07:2021 – Identification and Authentication Failures	Authentifizierungsmechanismen werden häufig falsch implementiert, so dass Angreifer Authentifizierungs-Tokens kompromittieren oder Implementierungsfehler ausnutzen können, um vorübergehend oder dauerhaft die Identität eines anderen Benutzers anzunehmen. Wird die Fähigkeit des Systems, den Kunden/Benutzer zu identifizieren, kompromittiert, ist die API-Sicherheit insgesamt gefährdet.
A08:2021 – Software and Data Integrity Failures	Software- und Datenintegritätsfehler beziehen sich auf Code und Infrastruktur, die nicht vor Integritätsverletzungen schützen, wenn beispielsweise eine Anwendung auf Plugins, Bibliotheken oder Module aus nicht vertrauenswürdigen Quellen, Repositories und Content Delivery Networks (CDNs) angewiesen ist. Eine unsichere CI/CD-Pipeline bietet das Potenzial für unbefugten Zugriff, bösartigen Code oder Systemkompromittierung.
A09:2021 – Security Logging and Monitoring Failures	Sicherheitsprotokollierung und Überwachung, kombiniert mit fehlender oder ineffektiver Integration in den Incident-Response-Prozess, ermöglichen es Angreifern, Systeme weiter anzugreifen, Persistenz aufrechtzuerhalten, auf weitere Systeme zu wechseln, um Daten zu manipulieren, zu extrahieren oder zu zerstören. Die meisten Studien zu Sicherheitsverletzungen zeigen auf, dass bis zur Entdeckung einer Sicherheitsverletzung über 200 Tage vergehen können und diese in der Regel von externen Parteien anstelle interner Prozesse oder Überwachungseinrichtungen aufgedeckt wird.
A10:2021 – Server Side Request Forgery (SSRF)	SSRF-Schwachstellen treten immer dann auf, wenn eine Webanwendung eine Remote-Ressource abruft, ohne die vom Benutzer angegebene URL zu validieren. Dadurch kann ein Angreifer die Anwendung dazu zwingen, eine manipulierte Anfrage an ein unerwartetes Ziel zu senden, selbst wenn sie durch eine Firewall, ein VPN oder eine andere Art von Netzwerk-Zugriffskontrollliste (“Access Control List”, ACL) geschützt ist. Moderne Webanwendungen bieten Endnutzern viele praktische Funktionen, und das Abrufen einer URL ist ein häufiges Szenario. Die Häufigkeit von SSRF nimmt folglich zu. Außerdem wird der Schweregrad von SSRF aufgrund von Cloud-Diensten und der Komplexität der Architekturen immer höher.

SEEBURGER

www.seeburger.com

Disclaimer
Diese Veröffentlichung enthält ausschließlich allgemeine Informationen. SEEBURGER erbringt mit dieser Veröffentlichung keine professionelle Dienstleistung, insbesondere keine rechtliche oder steuerliche Beratungsleistung. Diese Veröffentlichung ist nicht geeignet, um unternehmerische Entscheidungen zu treffen oder Handlungen vorzunehmen. Hierzu sollten sie sich von einem qualifizierten Berater (z. B. Rechtsanwalt und/oder Steuerberater) in Bezug auf den Einzelfall beraten lassen. Es werden keine (ausdrücklichen oder stillschweigenden) Aussagen, Garantien oder Zusicherungen hinsichtlich der Richtigkeit oder Vollständigkeit der Informationen in dieser Veröffentlichung gemacht, und SEEBURGER haftet nicht oder ist nicht verantwortlich für Verluste oder Schäden jeglicher Art, die direkt oder indirekt im Zusammenhang mit Informationen aus der Präsentation.

© SEEBURGER | DE 02/2024