



# Secure Coding

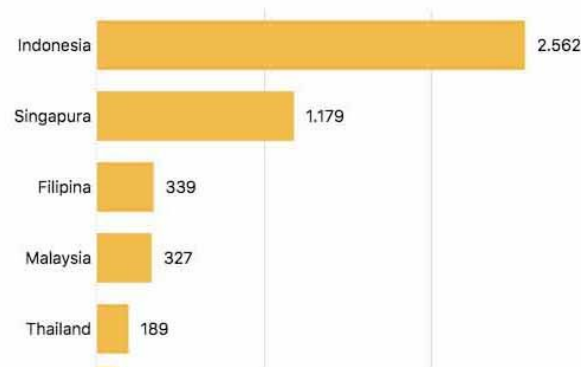
Secure API  
Development



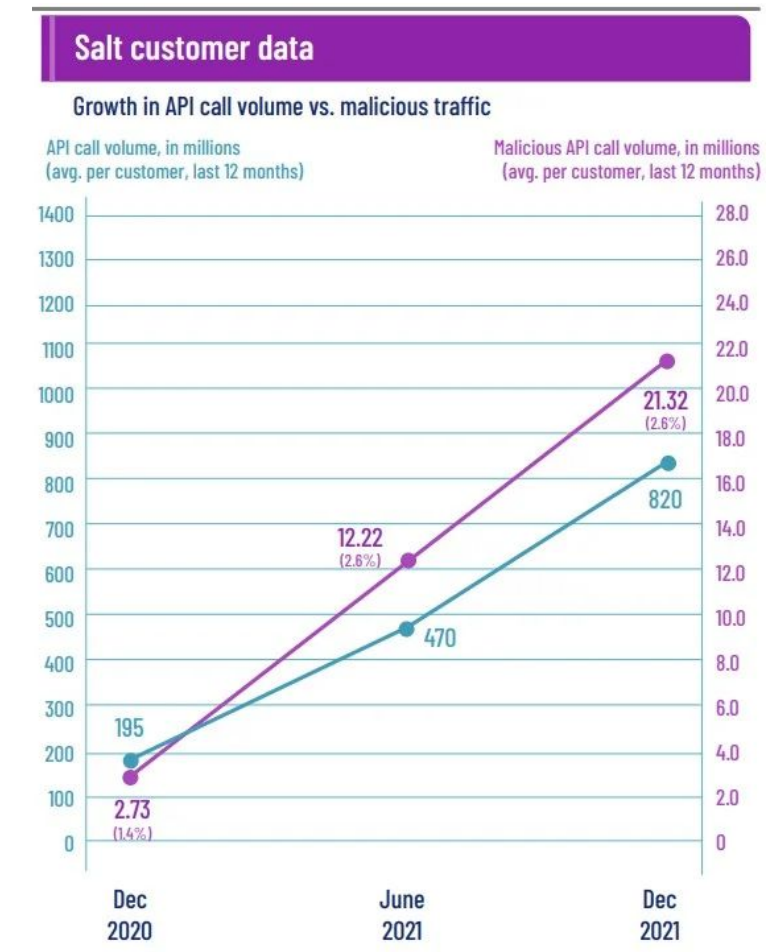
# Pentingnya Keamanan API di Indonesia

Pertumbuhan startup dan e-commerce di Indonesia (contoh: Gojek, Tokopedia) bergantung pada API.

- **Risiko unik di Indonesia:**
  - Serangan pada layanan keuangan (contoh: pembobolan API bank digital).
  - Pelanggaran data pribadi (UU PDP No. 27/2022).
- Serangan yang menyalahgunakan API pemrograman tumbuh lebih dari **600%** pada tahun 2021. (sumber: Salt).



Sumber: databoks.katadata.co.id



# API 01:2023 – Broken object level authorization

Penyerang **mensubstitusi** ID sumber daya milik mereka sendiri dalam panggilan API dengan ID sumber daya milik pengguna lain. Kurangnya pemeriksaan **otorisasi** yang tepat memungkinkan penyerang untuk mengakses sumber daya yang ditentukan. Serangan ini juga dikenal sebagai **IDOR (Insecure Direct Object Reference)**.

## Use case (Kasus Penggunaan)

- Parameter panggilan API menggunakan ID sumber daya yang diakses melalui API `/api/shop1/financial_info`.
- Penyerang mengganti ID sumber daya mereka dengan ID lain yang mereka tebak melalui `/api/shop2/financial_info`.
- API tidak memeriksa **permission** dan meloloskan panggilan tersebut.
- Masalah diperparah jika ID dapat di**enumerasi** `/api/123/financial_info`.

## How to prevent (Cara Mencegah)

- Implementasikan pemeriksaan **otorisasi** dengan **user policy** dan hierarki.
- Jangan bergantung pada ID yang dikirim oleh klien.
- Gunakan ID yang disimpan dalam **session object** sebagai gantinya.
- Periksa **otorisasi** untuk setiap permintaan klien untuk mengakses **database**.
- Gunakan ID acak yang tidak dapat ditebak (**UUIDs**).
- Implementasikan **test framework** yang kuat untuk secara spesifik menguji jenis kerentanan ini.



## API1:2023 - Broken Object Level Authorization



# API 02:2023 – Broken authentication

Implementasi **otentikasi** API yang buruk memungkinkan penyerang untuk mengambil alih identitas pengguna lain, atau mengakses sumber daya tanpa **otentikasi** sama sekali.

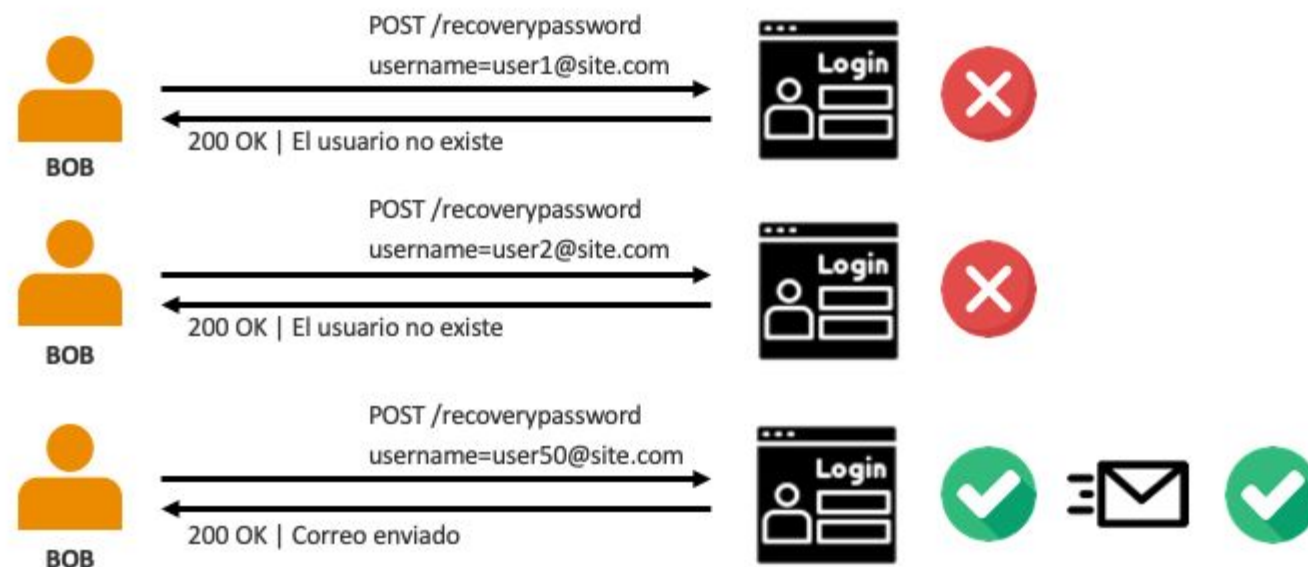
## Use case (Kasus Penggunaan)

- API yang tidak terlindungi yang dianggap "internal"
- **Autentikasi** lemah yang tidak mengikuti praktik terbaik industri
- **API key** lemah yang tidak dirotasi
- **Password** yang lemah, teks biasa (**plain text**), dienkripsi, di-*hash* dengan buruk, dibagikan, atau **password default**
- **Autentikasi** rentan terhadap serangan **brute force** dan **credential stuffing**
- **Credential** dan **key** disertakan dalam URL Kurangnya validasi **access token** (termasuk validasi **JWT**)
- **JWT** tanpa tanda tangan atau tanda tangan lemah yang tidak kedaluwarsa

## How to prevent (Cara Mencegah)

- Periksa semua kemungkinan cara untuk melakukan **otentikasi** ke semua API.
- API untuk pengaturan ulang **password** dan tautan satu kali juga memungkinkan pengguna untuk melakukan **otentikasi**, dan harus dilindungi dengan sama ketatnya.
- Gunakan **otentikasi** standar, pembuatan **token**, penyimpanan **password**, dan **multi-factor authentication (MFA)**.
- Gunakan **access token** dengan masa berlaku singkat (**short-lived**).
- Autentikasi aplikasi Anda (sehingga Anda tahu siapa yang berkomunikasi dengan Anda).

## API2:2023 - Broken Authentication



# API 03:2023 – Broken Object Property

## Level Authorization

**Endpoint** API dapat rentan terhadap serangan berdasarkan datanya: baik mereka dapat mengekspos lebih banyak data daripada yang dibutuhkan untuk tujuan bisnis mereka (**excessive information exposure**), atau mereka secara tidak sengaja menerima dan memproses lebih banyak data daripada yang seharusnya (**mass assignment**).

### Use case (Kasus Penggunaan)

- API mengembalikan objek data lengkap seperti yang disimpan dalam **backend database**.
- Aplikasi klien memfilter respons dan hanya menampilkan data yang benar-benar perlu dilihat oleh pengguna.
- Penyerang memanggil API secara langsung dan mengambil data sensitif yang akan difilter oleh UI.
- API bekerja dengan struktur data tanpa pemfilteran yang tepat.
- **Payload** yang diterima secara membabi buta diubah menjadi objek dan disimpan.

```
var user = new User(req.body);  
  
user.save();
```

- Penyerang dapat menebak *field* dengan melihat data permintaan **GET**.

# API 03:2023 – Broken Object Property

## Level Authorization

### How to prevent (Cara Mencegah)

#### Responses (Respons)

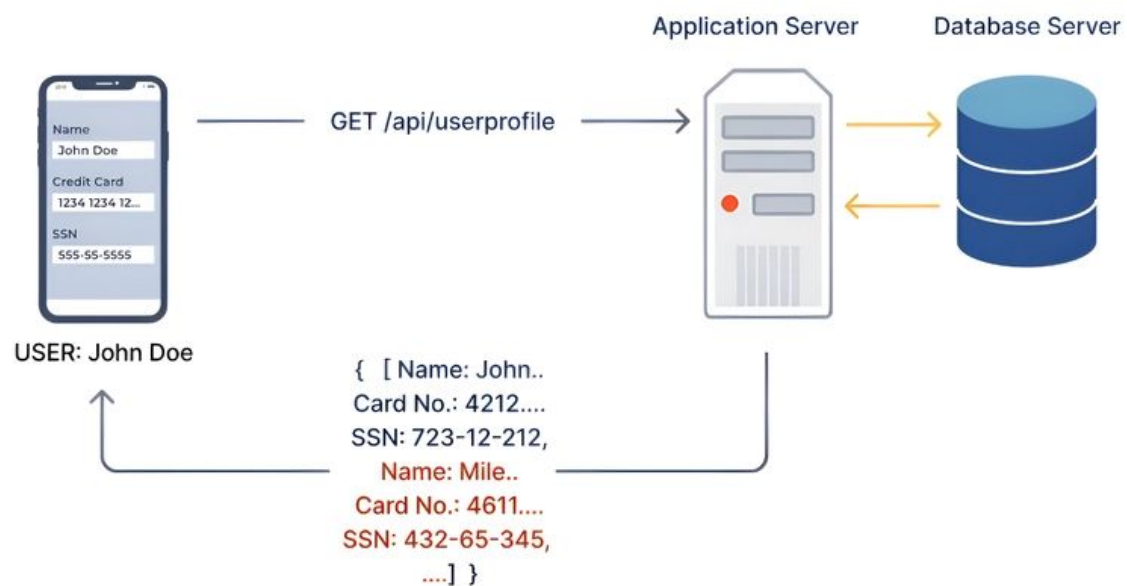
- Jangan pernah mengandalkan klien untuk memfilter data!
- Tinjau semua respons API dan sesuaikan agar sesuai dengan apa yang sebenarnya dibutuhkan oleh konsumen API.
- Definisikan **skema** dengan hati-hati untuk semua respons API.
- Jangan lupa respons kesalahan, definisikan **skema** yang tepat juga.
- Identifikasi semua data sensitif atau **Personally Identifiable Information (PII)**, dan justifikasi penggunaannya menggunakan proses **data governance** yang kuat.

#### Requests (Permintaan)

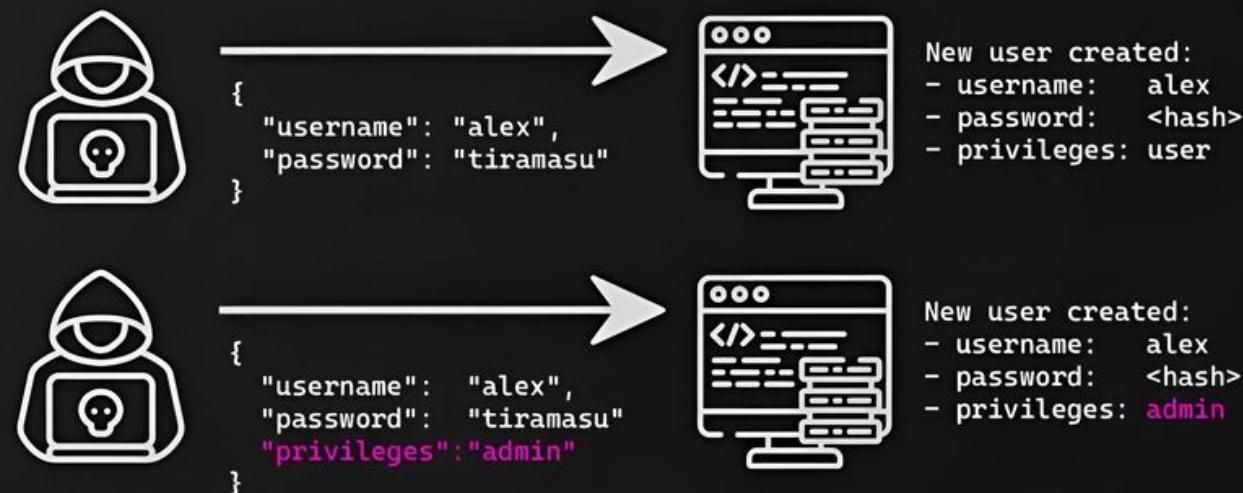
- Jangan secara otomatis mengikat data yang masuk ke objek internal.
- Definisikan secara eksplisit semua parameter dan *payload* yang Anda harapkan.
- Gunakan properti **readOnly** yang diatur ke **true** dalam **skema** objek untuk semua properti yang dapat diambil melalui API tetapi tidak boleh dimodifikasi.
- Definisikan dengan tepat **skema**, tipe, dan pola yang akan Anda terima dalam permintaan pada saat desain dan terapkan pada saat *runtime*.



## excessive information exposure



## Exploiting Mass Assignment



# API 04:2023 – Unlimited Resource

## Consumption

API tidak dilindungi terhadap jumlah panggilan atau ukuran *payload* yang berlebihan. Penyerang dapat menggunakan ini untuk **Denial of Service (DoS)** dan celah **autentikasi** seperti serangan **brute force**.

### Use case (Kasus Penggunaan)

- Penyerang membebani API dengan mengirimkan lebih banyak permintaan daripada yang dapat ditanganinya.
- Penyerang mengirimkan permintaan pada tingkat yang melebihi kecepatan pemrosesan API, sehingga membuatnya tersumbat.
- Ukuran permintaan atau beberapa *field* di dalamnya melebihi apa yang dapat diproses oleh API.
- Seorang penyerang mengirimkan permintaan dengan *payload* yang sangat besar atau kueri yang kompleks yang menyebabkan API mencapai *bottleneck* dan menjatuhkan permintaan.

### How to prevent (Cara Mencegah)

- Terapkan kebijakan **rate limiting** ke semua *endpoint*.
- Berikan perhatian khusus pada *endpoint* yang terkait dengan **autentikasi** yang merupakan target utama peretas.
- Sesuaikan **rate limiting** agar sesuai dengan apa yang dibutuhkan atau seharusnya diizinkan untuk diambil oleh metode API, klien, atau alamat.
- Alamat IP dapat dengan mudah dipalsukan, sebisa mungkin, konfigurasi **rate limiting** pada *key* yang berbeda, seperti *fingerprint*, atau *token*.
- Batasi ukuran *payload*, dan kompleksitas kueri.
- Tetapkan batas CPU/memori untuk *container* dan sumber daya komputasi.
- Batasi kompleksitas kueri (terutama dalam GraphQL) untuk mencegah komputasi berlebihan pada server.
- Batasi jumlah data yang dapat diambil oleh kueri dengan memberlakukan batasan pada ukuran **pagination**, atau jumlah halaman.
- Manfaatkan perlindungan **DDoS** dari penyedia *cloud* Anda.

<b>Legitimate</b> – max_return and page_size request attributes are normal	<b>Attack</b> – Attackers modify the request to return an abnormally high response size
POST /example/api/v1/provision/user/search HTTP/1.1 User-Agent: AHC/1.0 Connection: keep-alive Accept: */* Content-Type: application/json; charset=UTF-8 Content-Length: 131 X-Forwarded-For: 10.93.23.4	POST /example/api/v1/provision/user/search HTTP/1.1 User-Agent: AHC/1.0 Connection: keep-alive Accept: */* Content-Type: application/json; charset=UTF-8 Content-Length: 131 X-Forwarded-For: 10.93.23.4
{ "search_filter": "user_id=exampleId_100", "max_return": "250", "page_size": "250", "return_attributes": [ ] }	{ "search_filter": "user_id=exampleId_100", "max_return": "20000", "page_size": "20000", "return_attributes": [ ] }

# API 05:2023 – Broken function level authorization

API mengandalkan klien untuk menggunakan API tingkat pengguna atau tingkat admin/hak istimewa sebagaimana mestinya. Penyerang menemukan metode API admin yang "tersembunyi" dan memanggilnya secara langsung.

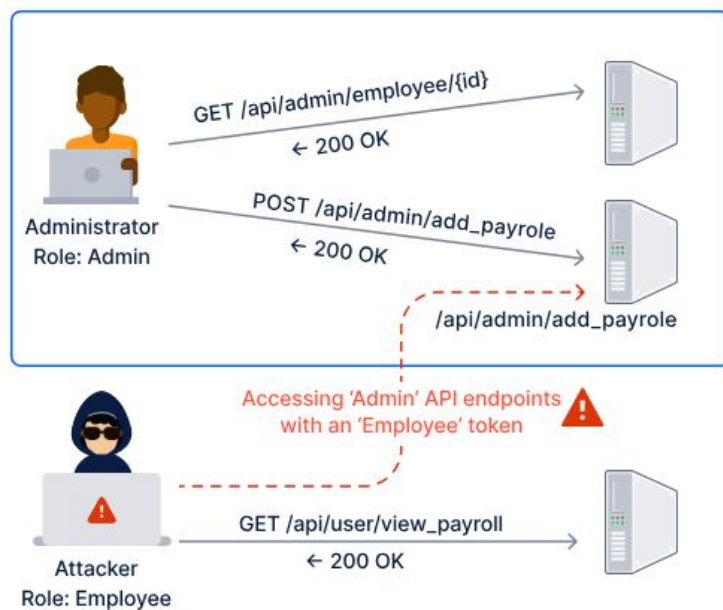
## Use case (Kasus Penggunaan)

- Beberapa fungsi administratif diekspos sebagai API.
- Operasi sensitif seharusnya hanya tersedia secara internal (misalnya menghapus sumber daya).
- Pengguna non-hak istimewa dapat mengakses fungsi-fungsi ini tanpa **otorisasi** jika mereka tahu caranya.
- Bisa jadi hanya masalah mengetahui URL, atau menggunakan *verb* yang berbeda atau parameter:
  - `/api/users/v1/user/myinfo`
  - `/api/admins/v1/users/all`

## How to prevent (Cara Mencegah)

- Jangan mengandalkan klien untuk memberlakukan akses admin.
- Terapkan kebijakan "**deny all**" (tolak semua) akses secara *default*.
- Hanya izinkan operasi kepada pengguna yang termasuk dalam grup atau *role* yang sesuai.
- Implementasikan **otorisasi** yang dirancang dan diuji dengan benar.





<b>Legitimate</b> – POST method is correctly requested	<b>Attack</b> – Request is modified to send a DELETE method
<b>POST</b> /example/api/v1/provision/user/search HTTP/1.1 User-Agent: AHC/1.0	<b>DELETE</b> /example/api/v1/provision/user/search HTTP/1.1 User-Agent: AHC/1.0
Connection: keep-alive Accept: */* Content-Type: application/json; charset=UTF-8 Content-Length: 131 X-Forwarded-For: 10.93.23.4  <pre>{   "search_filter":   "user_id=exampleId_100",   "max_return": "250",   "page_size": "250",   "return_attributes": [   ] }</pre>	Connection: keep-alive Accept: */* Content-Type: application/json; charset=UTF-8 Content-Length: 131 X-Forwarded-For: 10.93.23.4  <pre>{   "search_filter":   "user_id=exampleId_100",   "max_return": "250",   "page_size": "250",   "return_attributes": [   ] }</pre>

# API 06:2023 – Unrestricted Access to Sensitive Business Flows

A set of APIs exposes a business flow and an attacker abuses these APIs using automated methods to achieve a malicious intent, such as exfiltrating data or manipulating market or price data.

## Use case (Kasus Penggunaan)

- Seorang penyerang menemukan API untuk membeli produk secara daring dan menggunakan otomatisasi untuk membeli semua item produk yang baru dirilis dalam jumlah besar, yang kemudian mereka jual kembali.
- Informasi harga situs web *real-estate* dapat di-*scrape* dari waktu ke waktu untuk memprediksi tren harga rumah di suatu area.
- Penyerang dapat menggunakan otomatisasi untuk melakukan tindakan lebih cepat daripada pengguna manusia dan mendapatkan keuntungan yang tidak adil di situs lelang, atau yang serupa.

## How to prevent (Cara Mencegah)

- Pahami alur bisnis yang mungkin sensitif terhadap penyalahgunaan dan tambahkan lapisan perlindungan ekstra pada alur tersebut dan pastikan **autentikasi** diperlukan, menggunakan alur OAuth yang direkomendasikan, seperti `authorization_code`.
- Pastikan bahwa API sepenuhnya dilindungi dengan **rate-limiting** yang kuat di depan API.
- Pantau akses API dan batasi klien yang menggunakan perangkat mencurigakan atau berasal dari alamat IP yang berisiko.
- Identifikasi pola penggunaan non-manusia seperti transaksi yang sangat cepat, dan masukkan **Captcha** atau kontrol deteksi manusia lainnya.



# API 07:2023 – Server Side Request

## Forgery

Terjadi ketika sebuah API mengambil sumber daya jarak jauh tanpa memvalidasi URL yang diberikan oleh pengguna. Ini memungkinkan penyerang untuk memaksa aplikasi mengirimkan permintaan yang dibuat-buat ke tujuan yang tidak terduga, bahkan ketika dilindungi oleh *firewall* atau VPN.

### Use case (Kasus Penggunaan)

- Sebuah API menerima URL sebagai parameter untuk pengalihan (*redirection*), dan seorang penyerang menemukan bahwa mereka dapat menggunakan ini untuk mengalihkan respons ke situs jahat yang mampu mencuri data API sensitif.
- Seorang penyerang dapat memaksa API untuk memuat sumber daya dari server di bawah kendali mereka; ini adalah dasar dari serangan *key injection* pada JWT.
- Sebuah API mengizinkan akses ke *localhost* yang memungkinkan penyerang menggunakan permintaan yang salah format untuk mengakses sumber daya lokal.

### How to prevent (Cara Mencegah)

- Definisikan dengan tepat **skema**, tipe, dan pola yang akan Anda terima dalam permintaan pada saat desain dan terapkan pada saat *runtime*.
- Cegah server API Anda mengikuti pengalihan HTTP (*HTTP redirections*).
- Gunakan daftar putih (*allow list*) dari pengalihan atau akses yang diizinkan.
- Batasi rentang skema dan *port* URL yang diizinkan.
- Gunakan implementasi standar untuk *library* yang bertanggung jawab untuk memuat sumber daya, pastikan *library* tersebut tidak dapat mengakses *localhost*, dan menggunakan URL yang telah disanitasi dari *parser* URL yang aman.



# API 07:2023 – Server Side Request

## Forgery

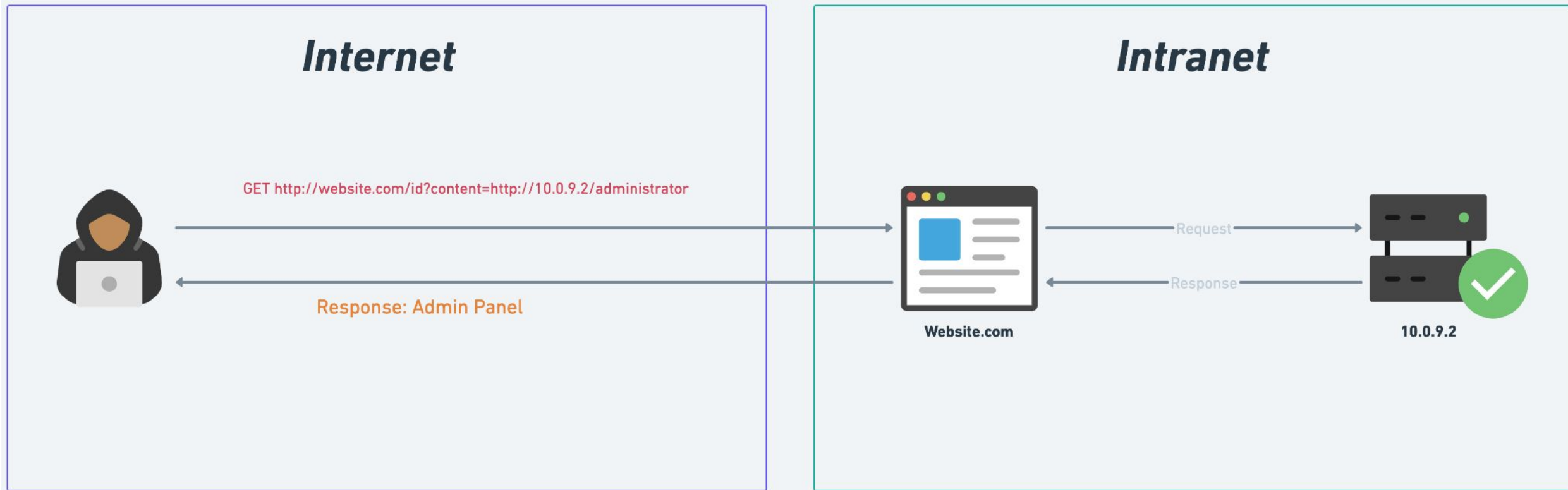
Terjadi ketika sebuah API mengambil sumber daya jarak jauh tanpa memvalidasi URL yang diberikan oleh pengguna. Ini memungkinkan penyerang untuk memaksa aplikasi mengirimkan permintaan yang dibuat-buat ke tujuan yang tidak terduga, bahkan ketika dilindungi oleh *firewall* atau VPN.

### Use case (Kasus Penggunaan)

- Sebuah API menerima URL sebagai parameter untuk pengalihan (*redirection*), dan seorang penyerang menemukan bahwa mereka dapat menggunakan ini untuk mengalihkan respons ke situs jahat yang mampu mencuri data API sensitif.
- Seorang penyerang dapat memaksa API untuk memuat sumber daya dari server di bawah kendali mereka; ini adalah dasar dari serangan *key injection* pada JWT.
- Sebuah API mengizinkan akses ke *localhost* yang memungkinkan penyerang menggunakan permintaan yang salah format untuk mengakses sumber daya lokal.

### How to prevent (Cara Mencegah)

- Definisikan dengan tepat **skema**, tipe, dan pola yang akan Anda terima dalam permintaan pada saat desain dan terapkan pada saat *runtime*.
- Cegah server API Anda mengikuti pengalihan HTTP (*HTTP redirections*).
- Gunakan daftar putih (*allow list*) dari pengalihan atau akses yang diizinkan.
- Batasi rentang skema dan *port* URL yang diizinkan.
- Gunakan implementasi standar untuk *library* yang bertanggung jawab untuk memuat sumber daya, pastikan *library* tersebut tidak dapat mengakses *localhost*, dan menggunakan URL yang telah disanitasi dari *parser* URL yang aman.



# API 08:2023 – Security misconfiguration

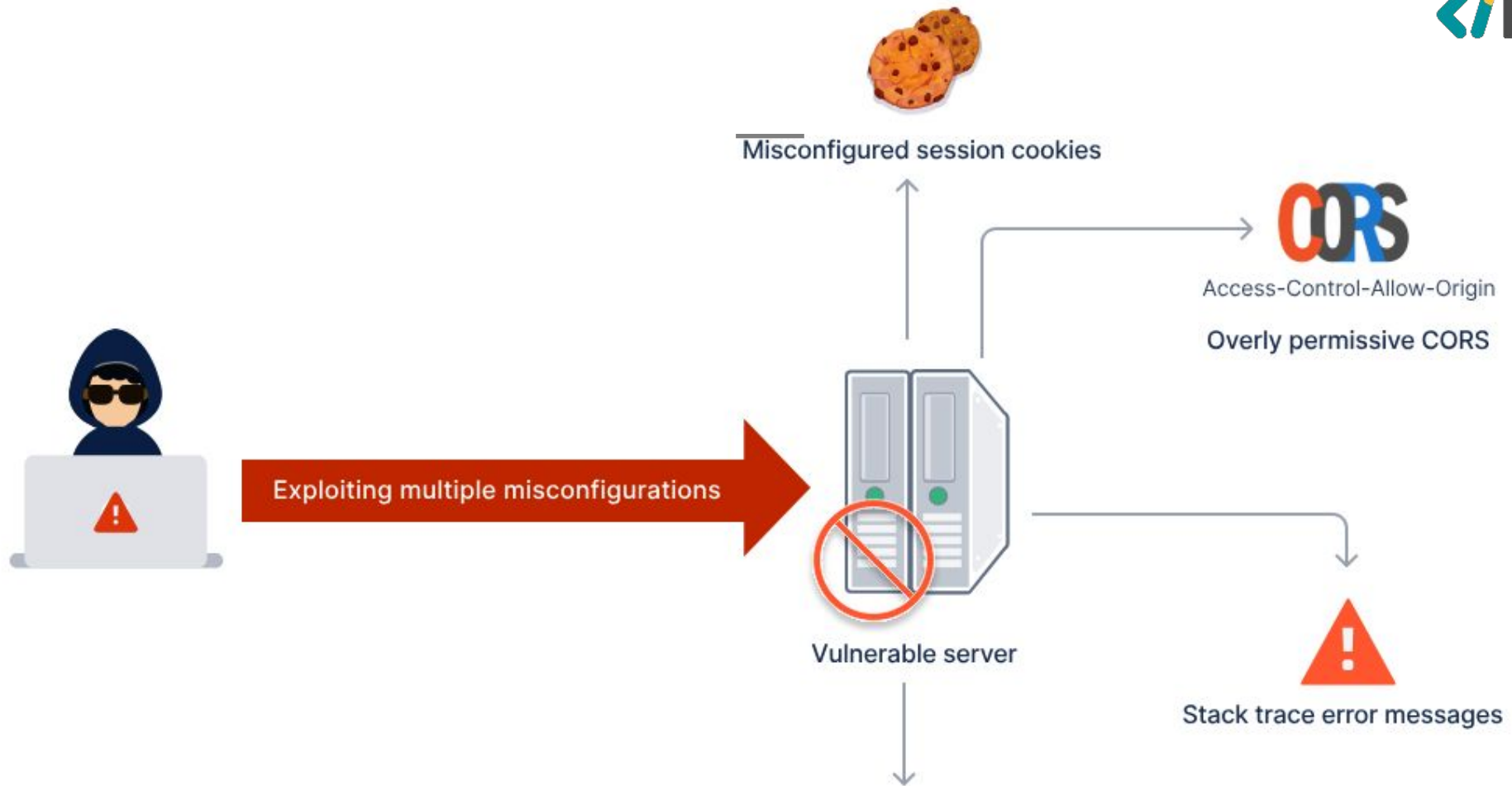
Konfigurasi server API yang buruk memungkinkan penyerang untuk mengeksploitasinya.

## Use case (Kasus Penggunaan)

- Sistem yang tidak di-*patch*.
- File dan direktori yang tidak terlindungi.
- *Image* yang tidak diperkeras (*unhardened*).
- TLS yang hilang, kedaluwarsa, atau salah konfigurasi.
- Panel penyimpanan atau manajemen server yang terekspos.
- Kebijakan CORS atau *security header* yang hilang.
- Pesan kesalahan dengan *stack trace*.
- Fitur yang tidak perlu diaktifkan.

## How to prevent (Cara Mencegah)

- Otomatiskan proses *hardening* dan *patching* dari seluruh *stack* API (kode, *library*, *container*).
- Otomatiskan pengujian *endpoint* API untuk kesalahan konfigurasi (versi TLS, *cypher*, *verb* yang buruk).
- Nonaktifkan fitur yang tidak perlu.
- Batasi akses administratif.
- Definisikan dan terapkan semua *output*, termasuk kesalahan.





# API 09:2023 – Improper inventory management

Penyerang menemukan versi non-produksi dari API (misalnya, *staging*, pengujian, beta, atau versi sebelumnya) yang tidak terlindungi sebaik API produksi, dan menggunakannya untuk meluncurkan serangan mereka.

## Use case (Kasus Penggunaan)

- DevOps, *cloud*, *container*, dan Kubernetes memudahkan untuk memiliki banyak *deployment* (misalnya, *dev*, *test*, *branch*, *staging*, dan versi lama).
- Keinginan untuk mempertahankan kompatibilitas ke belakang (*backward compatibility*) memaksa API lama tetap berjalan.
- Versi lama atau non-produksi tidak dipelihara dengan baik, tetapi *endpoint* ini masih memiliki akses ke data produksi.
- Setelah berhasil melakukan **otentikasi** dengan satu *endpoint*, penyerang dapat beralih ke *endpoint* produksi lainnya.

## How to prevent (Cara Mencegah)

- Pertahankan inventaris terkini dari semua *host* API.
- Batasi akses ke apa pun yang seharusnya tidak bersifat publik.
- Batasi akses ke data produksi, dan pisahkan akses ke data produksi dan non-produksi.
- Implementasikan kontrol eksternal tambahan, seperti *firewall* API.
- Pensiunkan dengan benar versi lama API atau *backport* perbaikan keamanan ke dalamnya.
- Implementasikan **otentikasi**, pengalihan (*redirect*), CORS, dan lain sebagainya yang ketat.

## IMPROPER INVENTORY MANAGEMENT



# API 10:2023 – Unsafe Consumption of APIs

Sistem berbasis API modern cenderung sangat saling terhubung, sering kali menggunakan API *upstream*. Sayangnya, API *upstream* ini sendiri mungkin rentan dan membahayakan konsumennya.

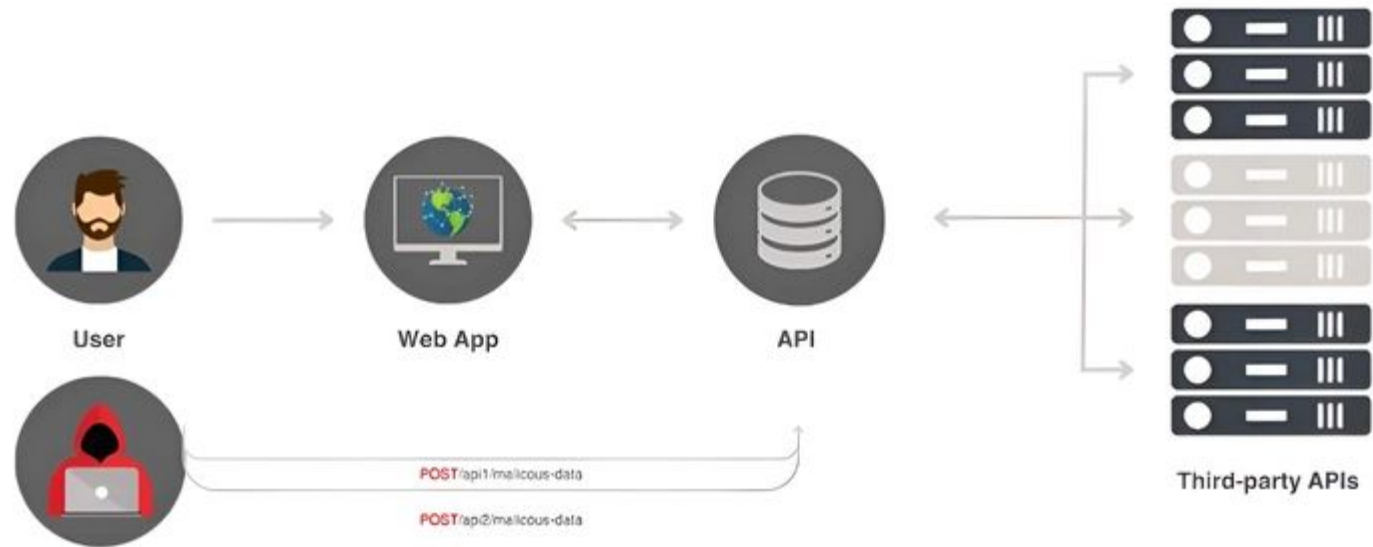
## Use case (Kasus Penggunaan)

- Sebuah API *upstream* mungkin secara tidak sengaja menyimpan data yang diberikan kepadanya oleh konsumen, sehingga melanggar peraturan tata kelola data konsumen.
- Penyedia API *upstream* mungkin diserang dan diretas (*compromised*) dan kemudian meneruskan data berbahaya ke konsumennya karena kurangnya kontrol internal yang memadai. Contoh tipikal adalah serangan **SQL injection**.

## How to prevent (Cara Mencegah)

- Sama seperti kasus *input* pengguna, jangan percayai data API *upstream*.
- Filter dan sanitasi setiap data *input* terlepas dari asalnya, terutama terhadap serangan *injection*.
- Pastikan bahwa penyedia API *upstream* menentukan kontrak API mereka, dan gunakan mekanisme *runtime* untuk memberlakukan kontrak ini.
- Asumsikan penyedia API *upstream* adalah bagian dari rantai pasokan Anda dan verifikasi proses pengembangan internal mereka.
- Gunakan saluran komunikasi yang aman setiap saat.

## UNSAFE CONSUMPTION OF API'S







# Auth

**TOKEN BASED  
AUTHORIZATION, JSON WEB  
TOKENS**

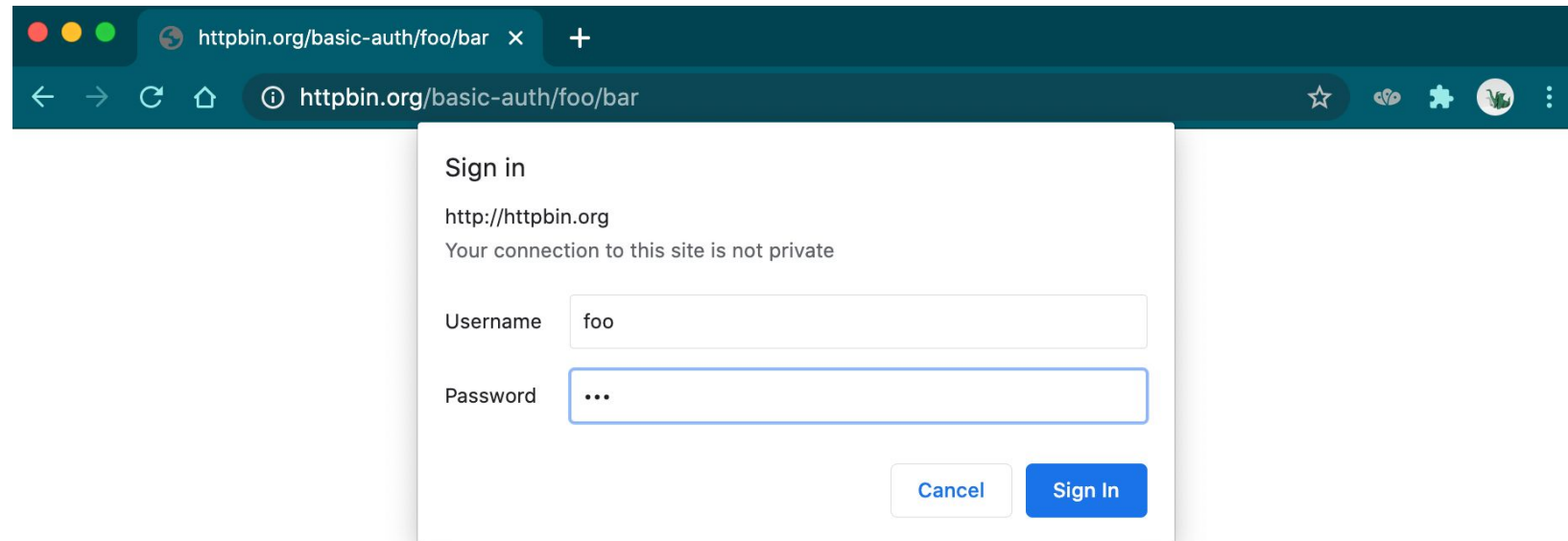
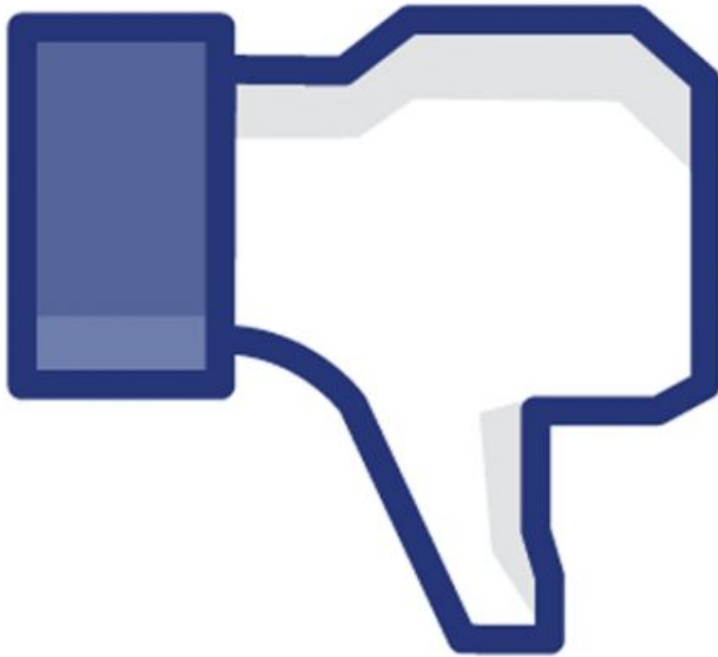


## AGENDA

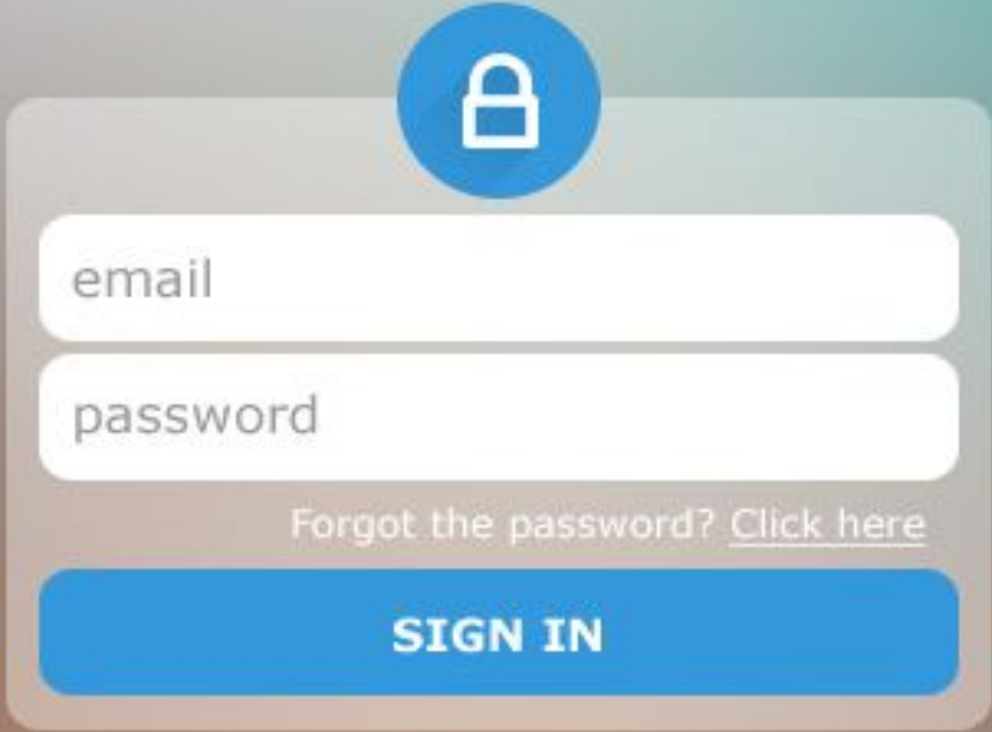
---

- 1 Token based auth
- 2 Json Web Tokens
- 3 OAUTH2
- 4 Live demo

# HISTORY: BASIC AUTHENTICATION



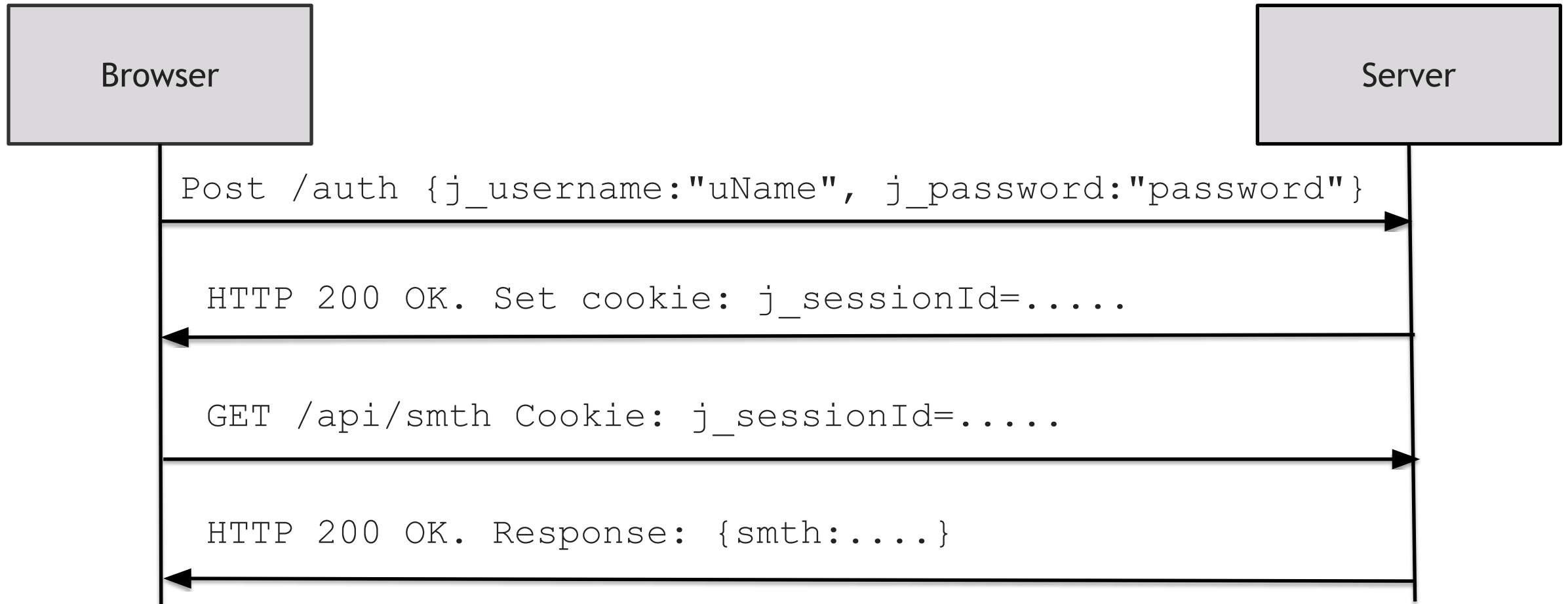
## HISTORY: COOKIE-BASED (FORMS)



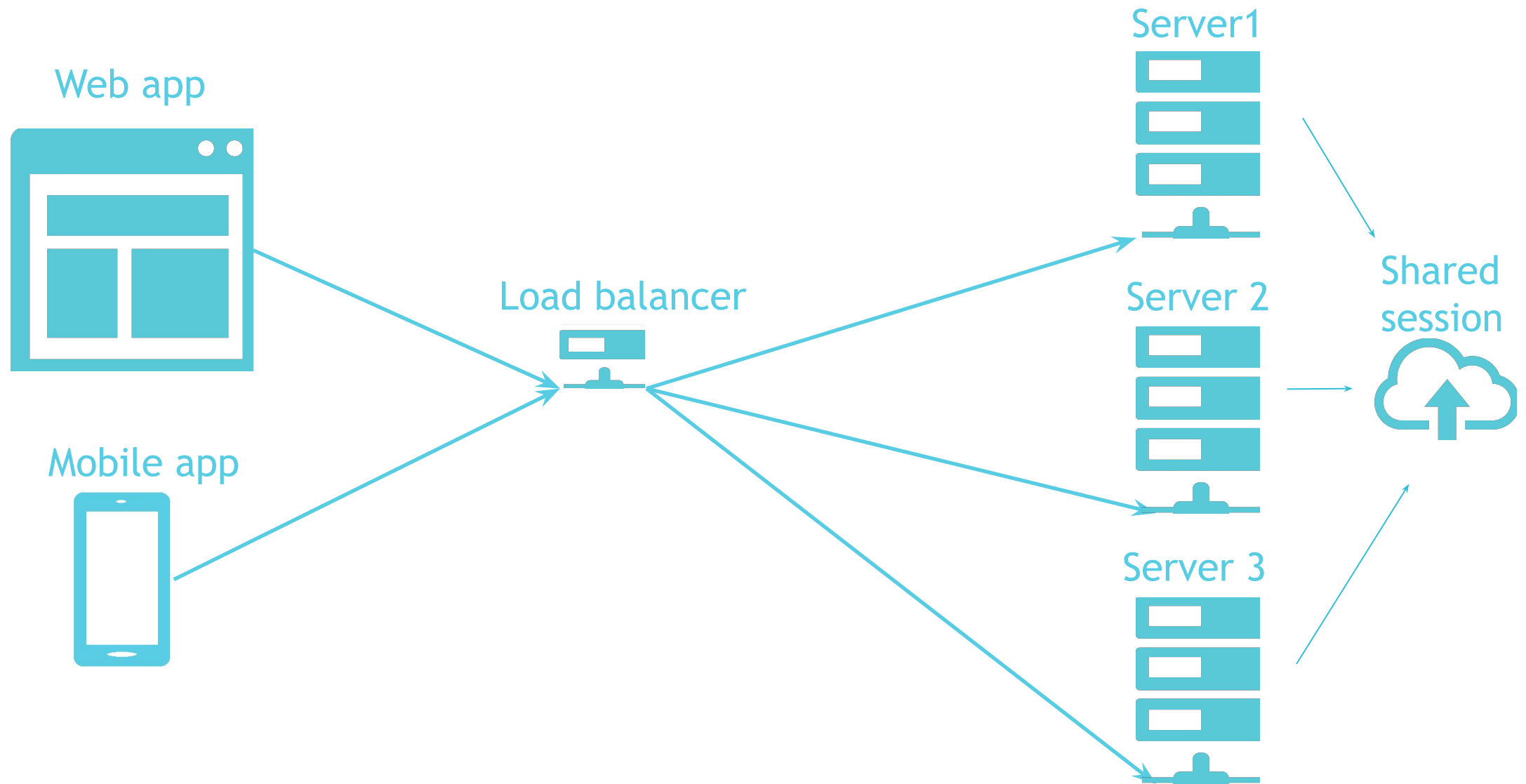
A login form illustration centered on a background with a teal-to-brown gradient. The form is a light gray rounded rectangle. At the top center is a blue circle containing a white padlock icon. Below it are two white rounded rectangular input fields. The first field contains the text 'email' in a light gray font. The second field contains the text 'password' in a light gray font. Below the password field is the text 'Forgot the password? [Click here](#)' in a light gray font. At the bottom of the form is a solid blue rounded rectangular button with the text 'SIGN IN' in white, bold, uppercase letters.

# HISTORY: COOKIE-BASED (FORMS): SEQUENCE

## DIAGRAM



# HISTORY: COOKIE-BASED (FORMS)





## HISTORY: COOKIE-BASED (FORMS): SUMMARY

- ✓ Mudah, transparan
  - ✓ Didukung oleh banyak framework
  - ✓ Possible to design forms
- 
- No browser clients
  - Distributed applications
  - Cross-domain apps



# TOKEN BASED AUTHORIZATION



# TOKEN BASED AUTHORIZATION

---

Accept:application/json, text/plain, \*/\*

Accept-Encoding:gzip, deflate, sdch, br

Accept-Language:ru-RU, ru;q=0.8,en-US;q=0.6,en;q=0.4

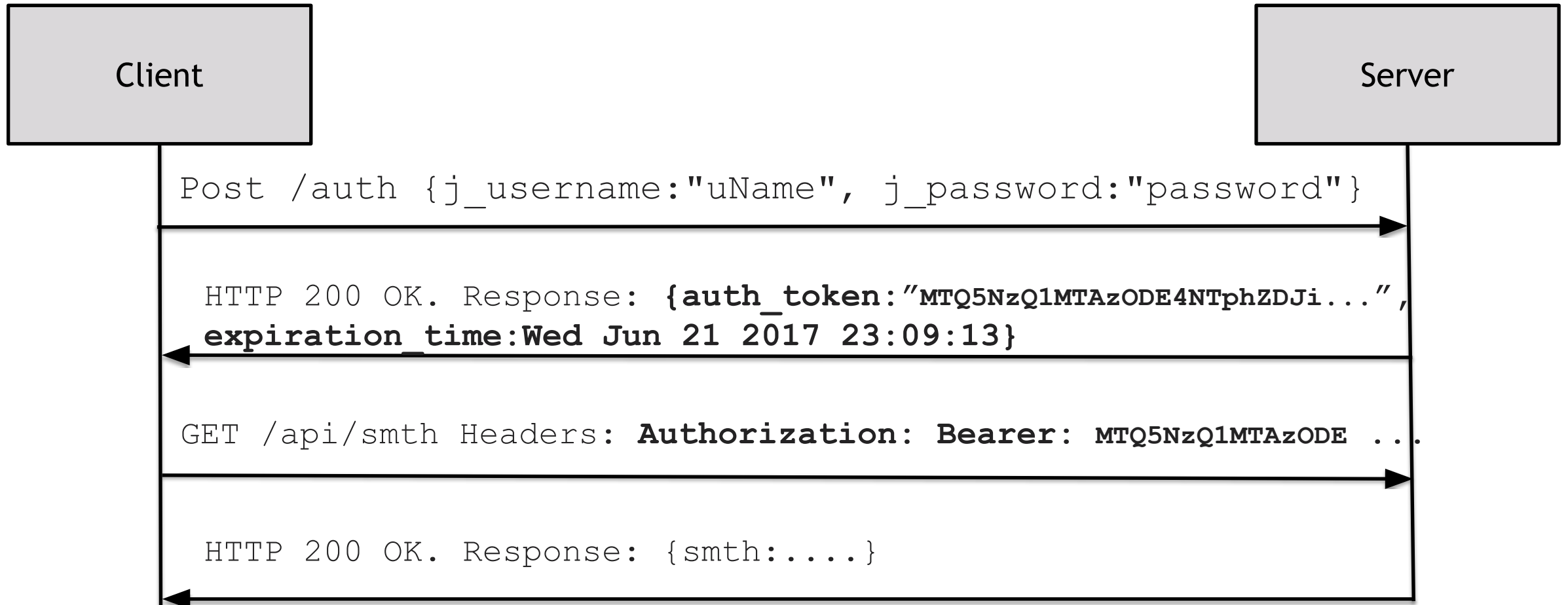
Cache-Control:no-cache

Connection:keep-alive

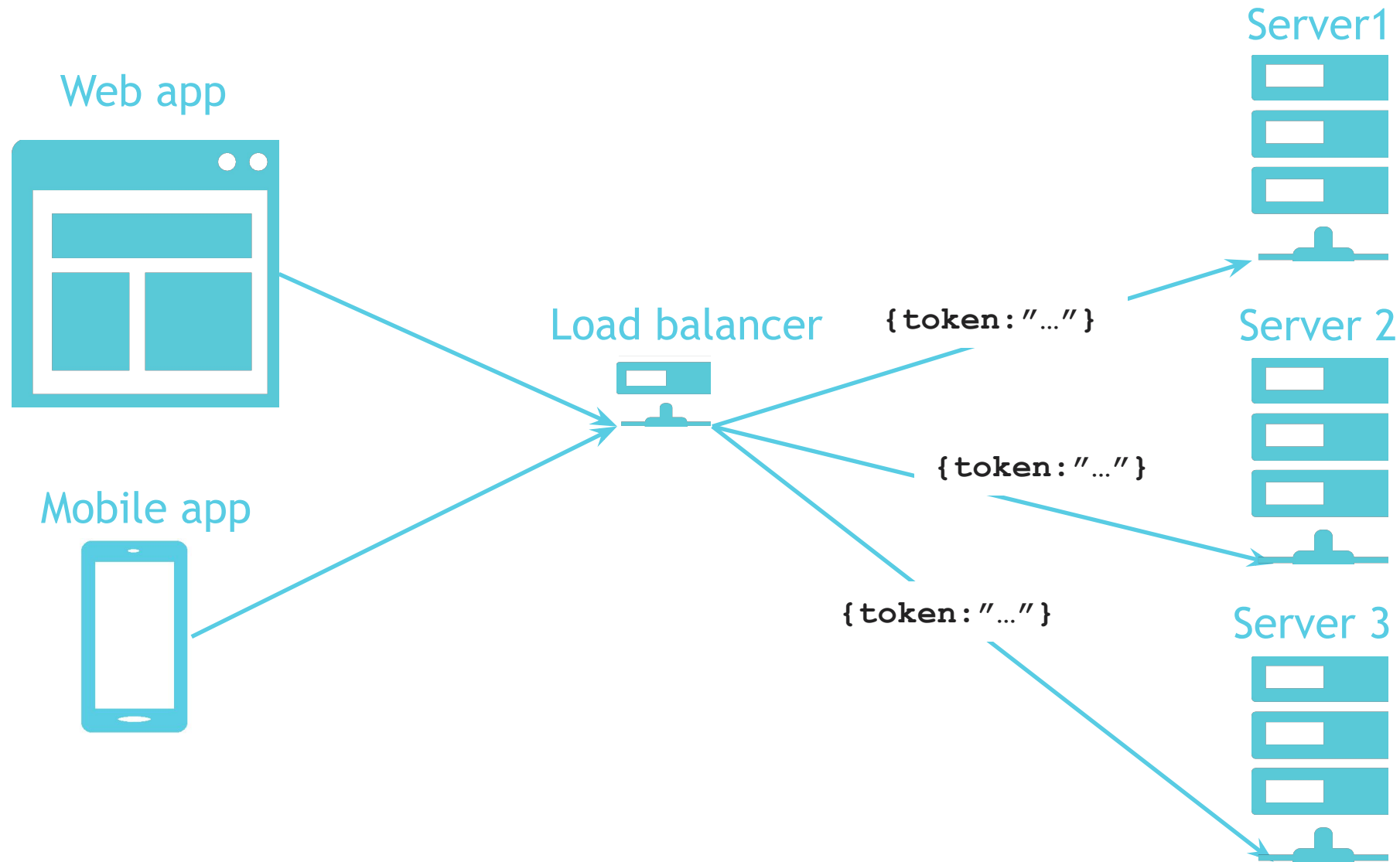
**Authorization: Bearer**

**MTQ5NzQ1MTAzODE4NTphZDZiODc0NTI0NmQxNmIyZDQ6dXNlc2VhYmNvbTo4YWYz  
Yzg1NWU4OTI4YThkNjYxNGNlOTA3Nzg4NGUzMTlmNDQ3YjQxZDdkNTdhMjc1YTY0MDRm  
YWYyN2FjMmJiMWFkOGQ3YjY0Y2QyZGM4MWQxNGQ0ZTk0MmMwZDhhY2MyY2RjNWQ2MjZl  
NTdhZTJiYTFlMzFhMzI4N2NmNmNmMA==**

# TOKEN BASED AUTHORIZATION - SEQUENCE DIAGRAMM



# TOKEN BASED AUTHORIZATION: SEVARAL SERVERS



## TOKEN BASED AUTHORIZATION: TOKEN SIZE

MTQ5NzQ1MTAzODE4NTphZDJIODc0NTI0NmQxNmIyZDQ6dXN1ckB1c2VyLmNvbTo4YWYzYzg1NWU4OTI4YThkNjYxNGNlOTA3NzgxNGUzMTlmNDQ3YjQxZDdkNTdhMjc1YTY0MDRmYWYyN2FjMmJiMWFkOGQ3YjY0Y2QyZGM4MWQxNGQ0ZTk0MmMwZDhhY2MyY2RjNWQ2MjZlNTdhZTJiYTF1MzFhMzI4N2NmNmNmMA==



## TOKEN BASED AUTHORIZATION: SUMMARY

---

- ✓ Didukung oleh banyak framework
- ✓ Implementasi mudah untuk aplikasi terdistribusi
- ✓ Good support for not-browser clients
- ✓ Implementasi fleksibel: konten, ukuran, enkripsi, kedaluwarsa
- ✓ Works with different domains
  - Header permintaan perlu diperluas
  - Tidak standard
  - Be careful with size



# JSON WEB TOKENS



## JWT: STRUCTURE

Header.Payload.Signature

## JWT: STRUCTURE: HEADER

---

```
{ "alg": "HS256", "typ": "JWT" }
```

## JWT: STRUCTURE: PAYLOAD

```
{"sub": "1234567890", "name": "John Doe", "admin": true}
```

iss – address or name of auth center

sub – user id

aud – client name

exp – expiration

nbf – time from

iat – time of token creation

jti – token id

## JWT: STRUCTURE: SIGNATURE

---

`HMACSHA256 (base64UrlEncode (header)`

`+ "." + base64UrlEncode (payload) , secret)`



# JWT: STRUCTURE

---

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0OTc0OTg1NDUsInVzZXJfbmFtZSI6InVzZXIiLCJhdXRob3JpdGllcyI6WyJST0xFX1VTRVIiLCJST0xFOX0FDVFVBVE9SIiwiaWF0IjE0OTc0OTg1NDUsImF1dG8iOiJ1bWVudF9pZCI6ImFjbWUiLCJzY29wZSI6WyJvcGVuaWQiXX0.Tp7xPQozl-01IyPD4tW7F0nb7oYIEgsxHStgcoJT8IBexYcLfcY6j3jNqbKse4aOlrwB8EwVsYbGFvLfL07Nh6rpPKTbDesih99b2fmGApf9ECQlwSeElV9uGy6l2vhkgJbVuxf8YJNmVr7lbPyZ-yA7FdwyiigWS-HSRMOP41konsR3Kj04glZpW0aD5BjvcJNFz5F4tiSeMOFLw2lGVQ8PXfoU9VKtN1eLFgV--JYJmH8tD7OAF6usITQeYPa7gmIIJb49B4J4JDnkPBcMVfCeLfjK9TXyOe5M0wGL615WG8Cc5dl23j1Qin-K7RJTXts1lHe4jriG3TObEfcJg

## JWT: SUMMARY

---

- ✓ Specification for tokens
  - ✓ Self-contained token
  - ✓ Token signature
- 
- Rather big size
  - Konten tidak terenkripsi
- ! Use HTTPS**



# OAuth 2



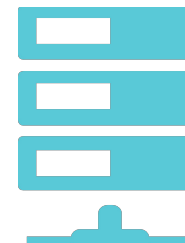
# OAUTH2: WHAT IS IT?



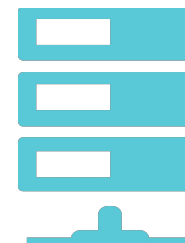
Server1



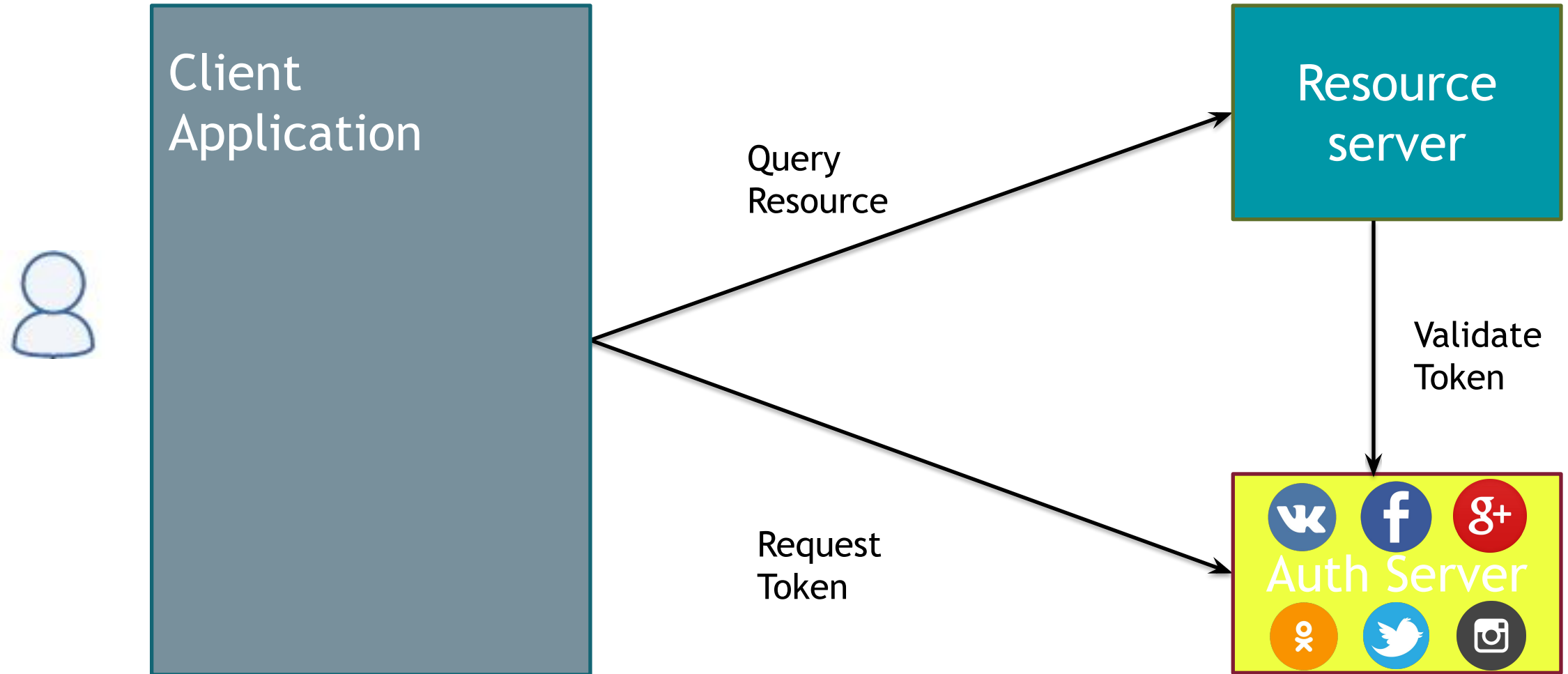
Server 2



Server 3



# OAUTH2: ABSTRACT FLOW



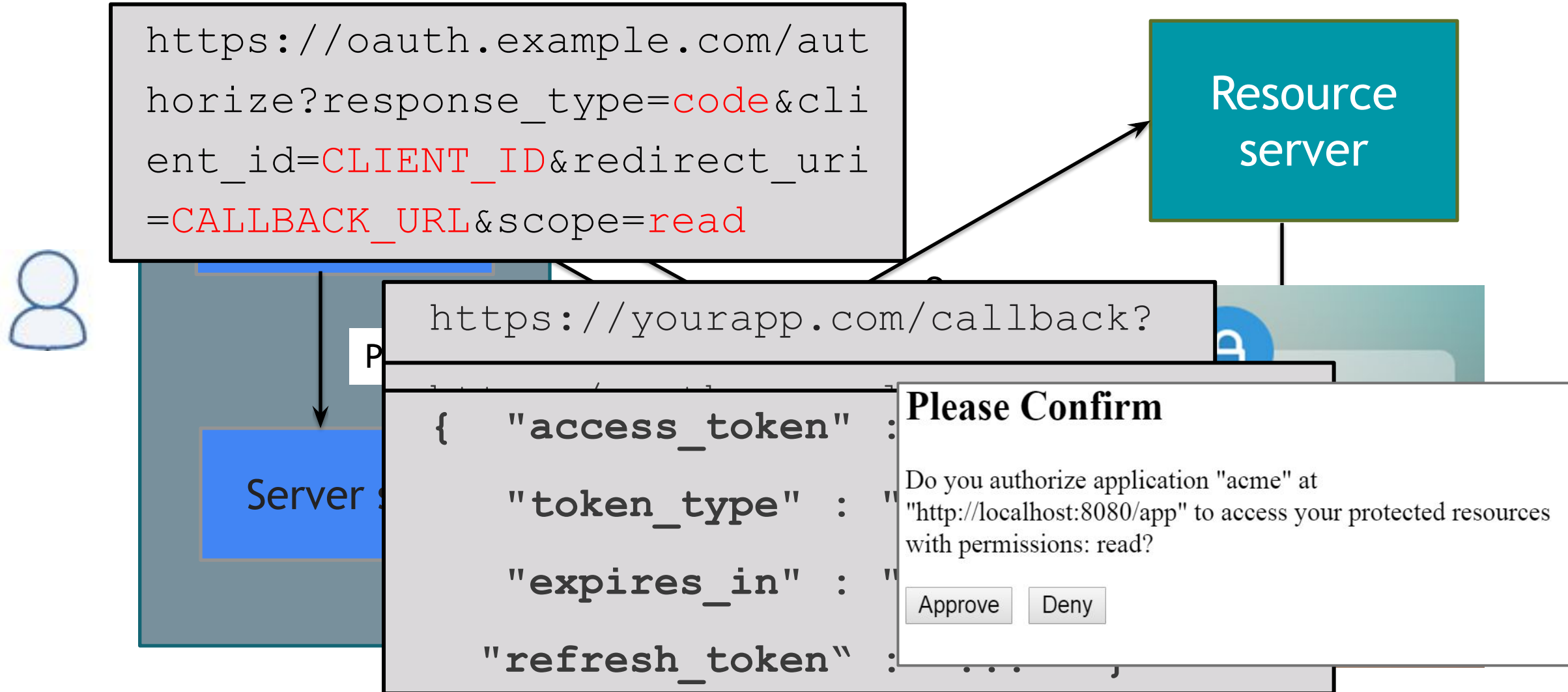
## OAUTH2: GRANT TYPES

---

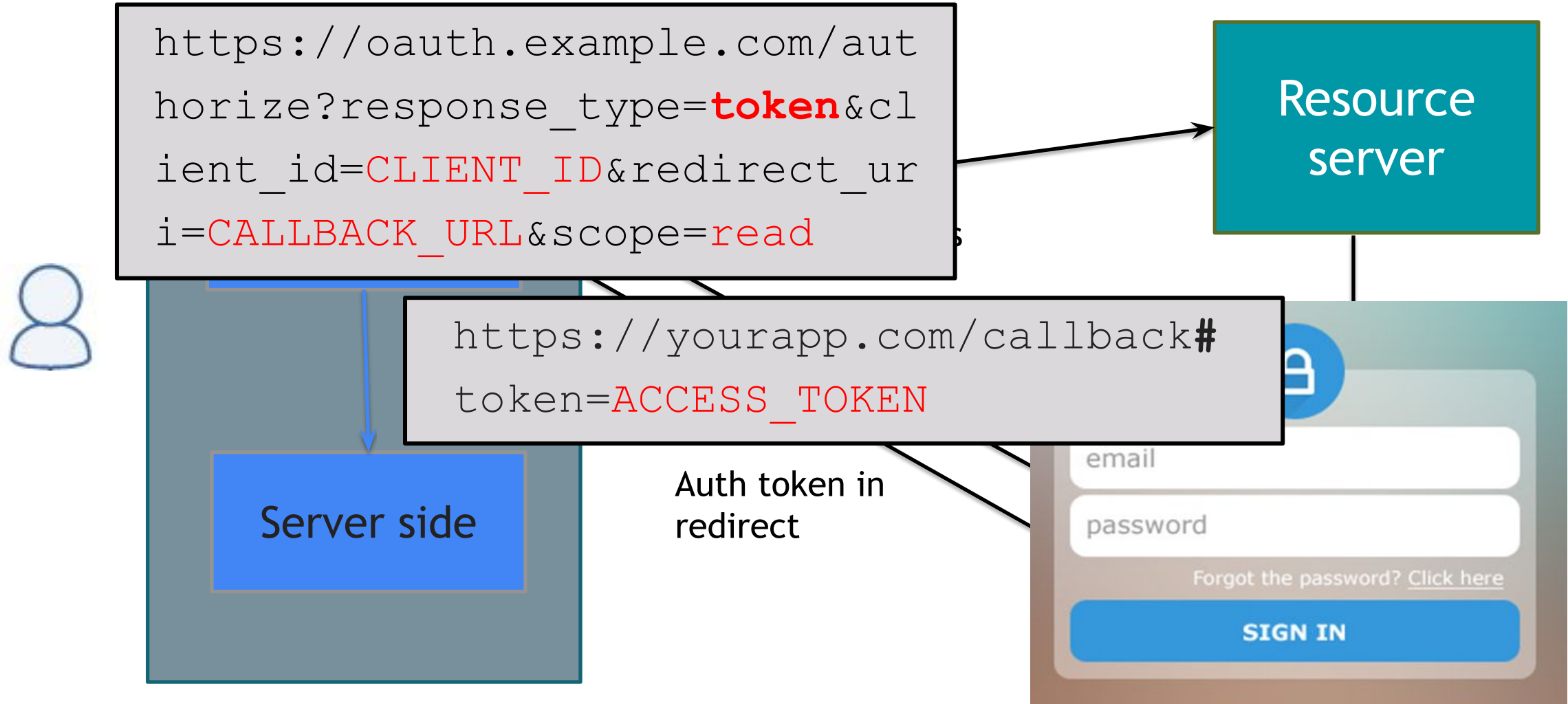
- 1 Authorization code(backend)
- 2 Implicit (frontend)
- 3 Username/password (deprecated)
- 4 Grant for credentials editing



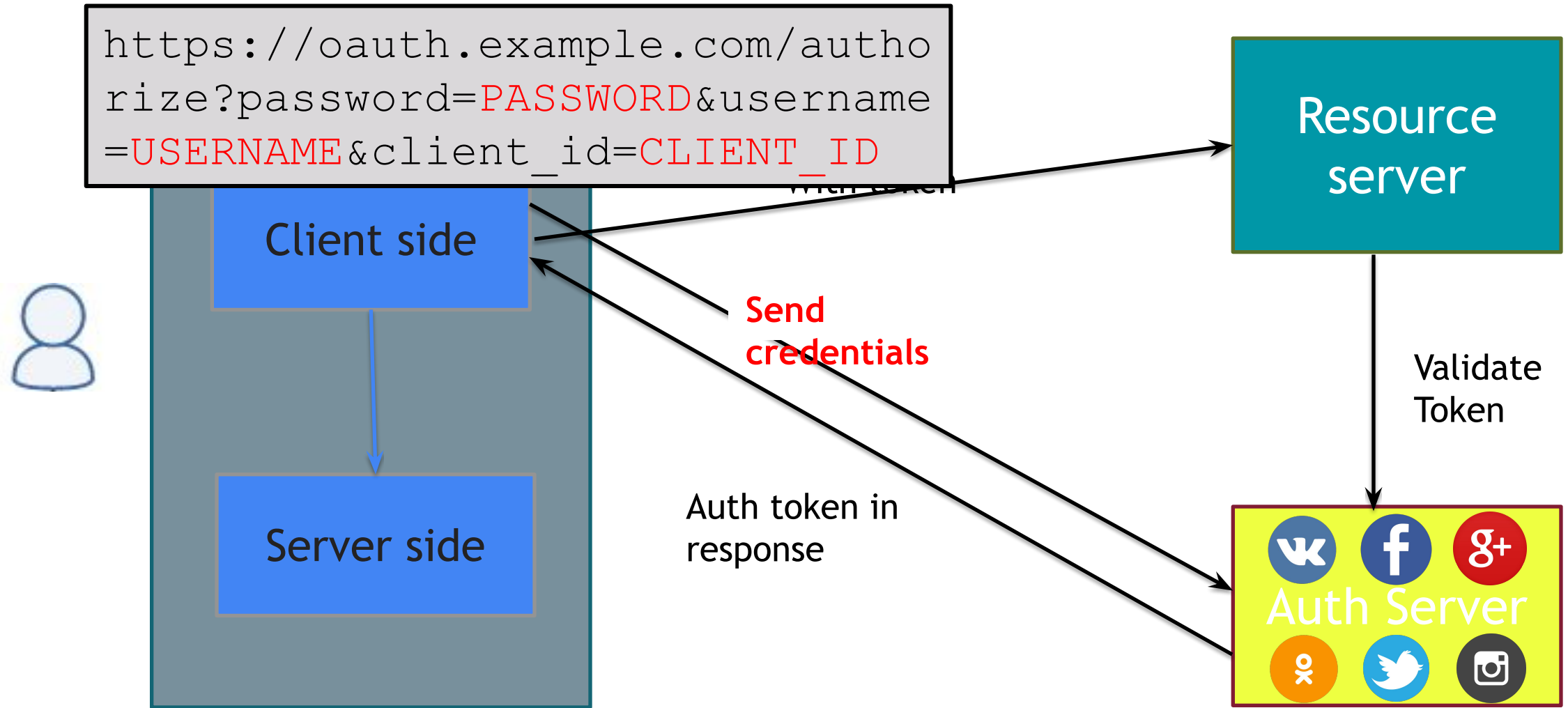
# OAUTH2: AUTHORIZATION CODE - ANIMATION



# OAUTH2: GRANT TYPE IMPLICIT - ANIMATION



# OAUTH2: GRANT TYPE – USER PASSWORD - ANIMATION



## OAUTH2: GRANT TYPE – CLIENT CREDENTIALS

---

**https://**oauth.example.com/token?grant\_type=client\_credentials&client\_id=CLIENT\_ID&client\_secret=CLIENT\_SECRET

# OAUTH2: APP REGISTRATION (ON FB EXAMPLE)

☐ Keep me logged in [Forgot your password?](#)

## Sign Up

It's free and anyone can join

First Name:

Last Name:

Your Email:

New Password:

I am:

Select Sex: ▼

Birthday:

Month: ▼

Day: ▼

Year: ▼

Why do I need to provide this?

[Create a Page](#) for a celebrity, band or business.

# **OAUTH2: SUMMARY**

---

- ✓ Ability to authorize user for other application
  - ✓ Separated login and business
  - ✓ Application can't steal user info
  - ✓ Based on tokens
- 
- Gaps in specification
  - Different implementations in popular services
  - High complexity
  - Possible performance issues
- ! Use HTTPS**

# LINKS

---

[HTTP Authentication: Basic and Digest Access Authentication](#)

link to oauth2 spec <https://tools.ietf.org/html/rfc6749>

JWT spec <https://tools.ietf.org/html/rfc7519>

<https://vk.com/editapp?act=create>

[https://vk.com/dev/mobile\\_apps](https://vk.com/dev/mobile_apps)

[https://vk.com/dev/android\\_sdk](https://vk.com/dev/android_sdk)

<https://jwt.io/introduction/>

<https://oauth2.thephpleague.com/requirements/> - good description

<https://github.com/andrey-radzkov/tech-talk-oauth2-demo> - demo

<https://medium.com/codenx/oauth-2-0-4cddd6c7471f>