BEZPIECZEŃSTWO API INTERNETOWYCH, PODATNOŚCI W SIECI WEB.

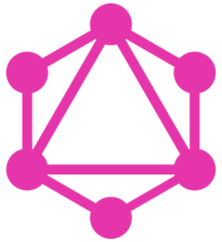
Spis Treści

- Wprowadzenie do bezpieczeństwa API
- Typowe podatności i zagrożenia związanie z API
- Standardy i najlepsze praktyki zabezpieczania API

API (Application Programming Interface) to zestaw reguł i mechanizmów umożliwiających komunikację między różnymi aplikacjami lub systemami. W kontekście aplikacji internetowych API pozwala na wymianę danych między frontendem (interfejsem użytkownika) a backendem (serwerem) lub integrację zewnętrznych usług, takich jak płatności, mapy czy media społecznościowe. Dzięki API aplikacje stają się modularne, skalowalne i mogą korzystać z funkcji innych systemów bez potrzeby znajomości ich wewnętrznej struktury.

TYPY API

Najpopularniejszy styl architektury API. Opiera się na protokole HTTP i wykorzystuje standardowe metody, takie jak GET, POST, PUT, DELETE. REST jest prosty, skalowalny i łatwy w użyciu.



Elastyczne API, które pozwala klientowi precyzyjnie określić, jakie dane chce otrzymać. Idealne do pracy z rozbudowanymi i dynamicznymi strukturami danych.

Starszy, bardziej sformalizowany protokół oparty na XML. Często stosowany w korporacyjnych środowiskach wymagających wysokiego poziomu bezpieczeństwa i niezawodności.

ZNACZENIE BEZPIECZEŃSTWA API

Co grozi brakiem zabezpieczeń?

Wycieki danych wrażliwych – ujawnienie danych użytkowników, takich jak hasła, numery kart czy dane osobowe.

Nieautoryzowany dostęp – możliwość przejęcia kontroli nad zasobami lub usługami przez osoby nieuprawnione.

Utrata reputacji – brak zaufania klientów i partnerów biznesowych po incydencie bezpieczeństwa

Ataki typu DDoS – przeciążenie API wskutek braku ograniczeń liczby żądań (rate limiting).

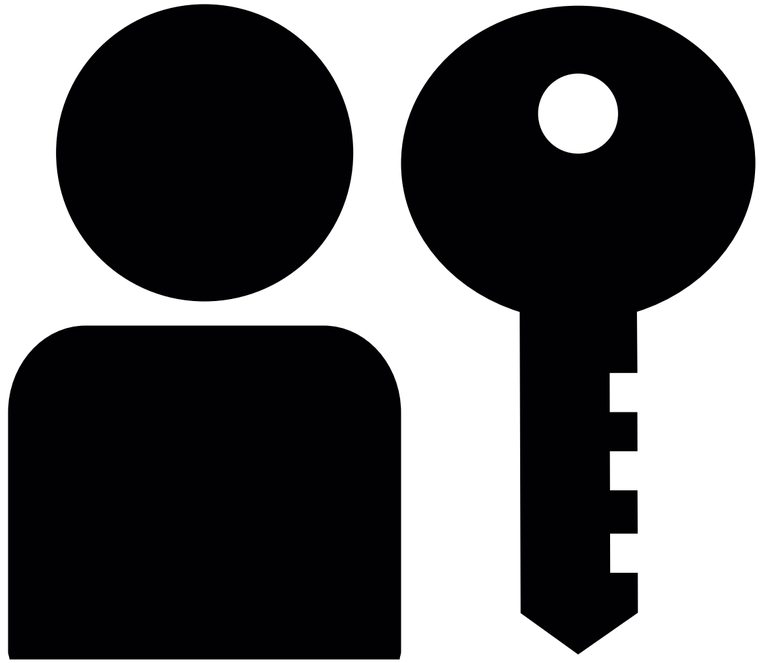
Manipulacja danymi – zmiana, usunięcie lub wprowadzenie nieprawidłowych danych przez złośliwe żądania.

Koszty finansowe – wysokie kary regulacyjne (np. RODO) oraz koszty związane z usuwaniem skutków ataku.

Typowe podatności i zagrożenia związane z API

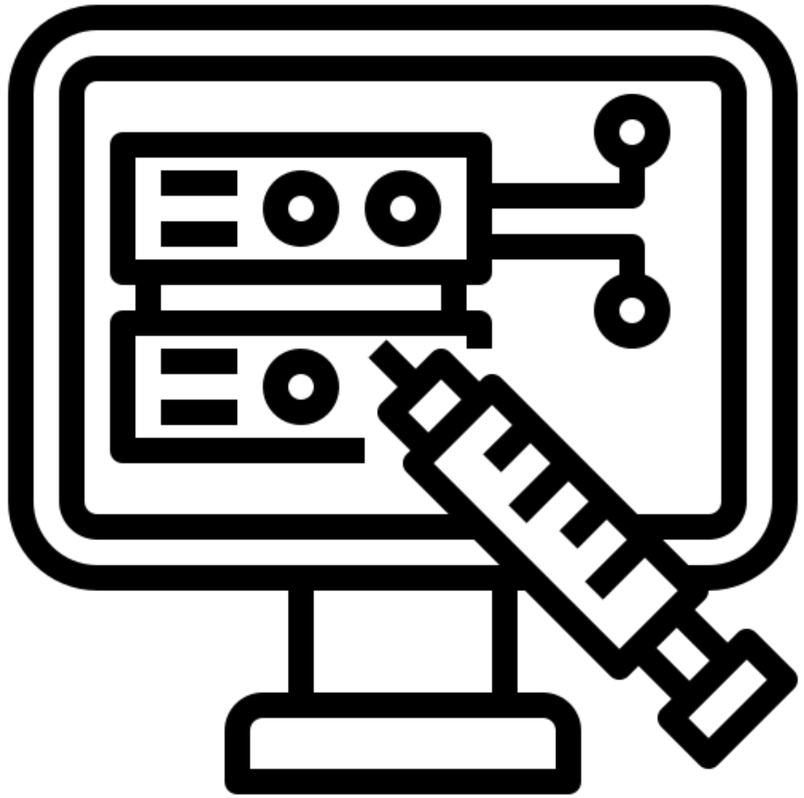
Brak autoryzacji uwierzytelniania

- **Nieautoryzowany dostęp** – np. do danych użytkowników, funkcji administracyjnych lub zasobów systemu.
- istnieją programy np Gobuster które mogą przeprowadzić atak brute-force na strukturę katalogów na serwerze i przez co też wykryć niezabezpieczone endpointy
- **Ujawnienia poufnych danych** – brak weryfikacji tożsamości użytkownika może umożliwić dostęp do danych wrażliwych.
- **Wykorzystania API przez boty** – bez zabezpieczeń, takich jak klucze API, CAPTCHA lub mechanizmy limitujące, API może być nadużywane przez zautomatyzowane skrypty.



Injections (SQL, NoSQL, XML, Command Injection)

- Ataki które korzystają z niepoprawnej walidacji wprowadzanych danych albo też braku jakiejkolwiek walidacji
- Wycieki danych
- Utrata Integralności
- Nieautoryzowany dostęp



Brak Rate Limiting

- Podatność na ataki DDos
- Podatność na ataki bruteforce
- Generowanie kosztów

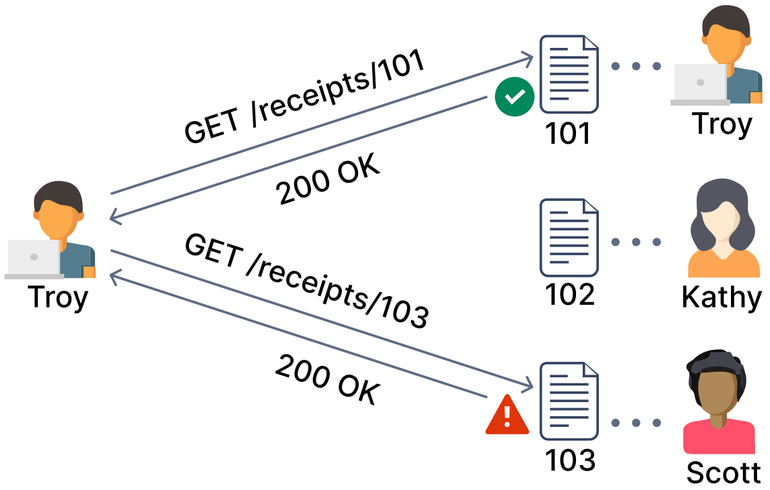
Data Exposure

- Nadmiernie szczegółowe odpowiedzi API - Zagrożenie
- Nadmierne logowanie szczegółów operacji w API może prowadzić do gromadzenia się wrażliwych danych w logach systemowych
- Brak odpowiedniego maskowania lub szyfrowania wrażliwych danych w odpowiedziach API naraża je na przechwycenie
- Brak właściwej filtracji i walidacji danych wyjściowych może skutkować wyciekiem danych osobowych użytkowników systemu

Broken Object Level Authorization (BOLA)

- Nieprawidłowe sprawdzanie lub brak, czy użytkownik ma uprawnienia do dostępu do konkretnego obiektu lub zasobu

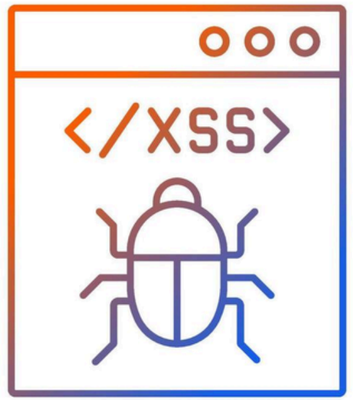
- **PRZYKŁAD**: user 123 ma dostęp i prawo dostępu do /profile/user123, user 123 może spróbować dostać się do zasobu /profile/user456 przez brak lub błędną autoryzację udaje mu się to



Cross-Site Scripting (XSS)
Cross-Site Request Forgery (CSRF)

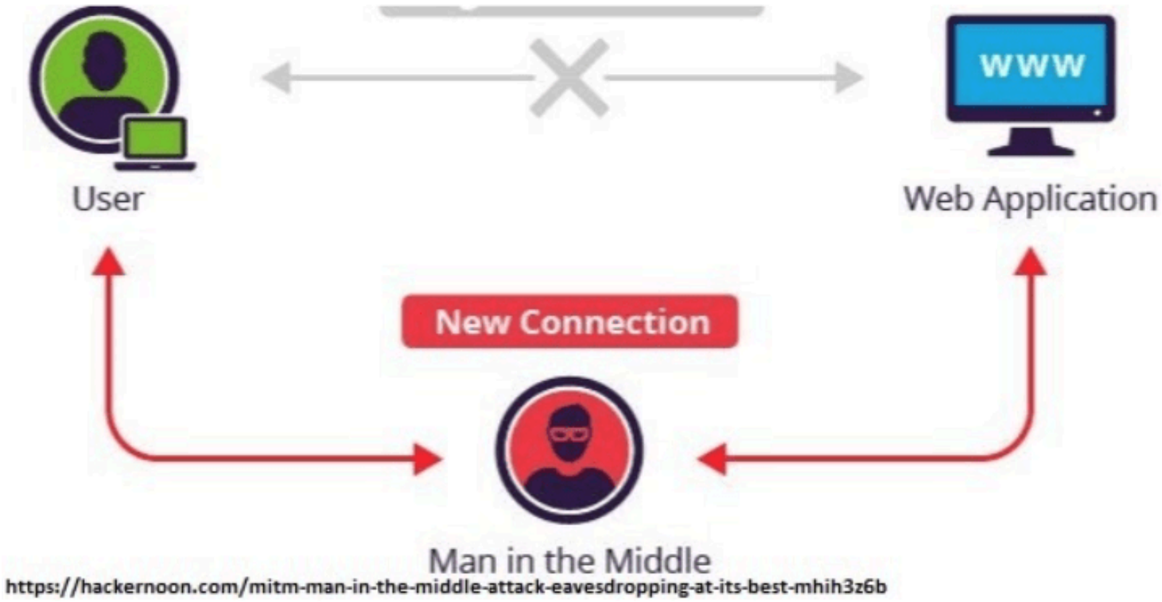
- XSS to atak polegający na wstrzyknięciu złośliwego kodu JavaScript do zaufanej witryny. Gdy inni użytkownicy odwiedzają zainfekowaną stronę, ich przeglądarki nieświadomie wykonują ten kod, co może prowadzić do kradzieży danych sesji, cookies lub manipulacji zawartością strony.

- CSRF działa inaczej - wykorzystuje fakt, że użytkownik jest zalogowany na danej stronie. Atakujący tworzy specjalną stronę, która automatycznie wysyła żądania do atakowanego serwisu (np. przelew bankowy), wykorzystując zapisane w przeglądarce dane uwierzytelniające ofiary.



Man in the middle

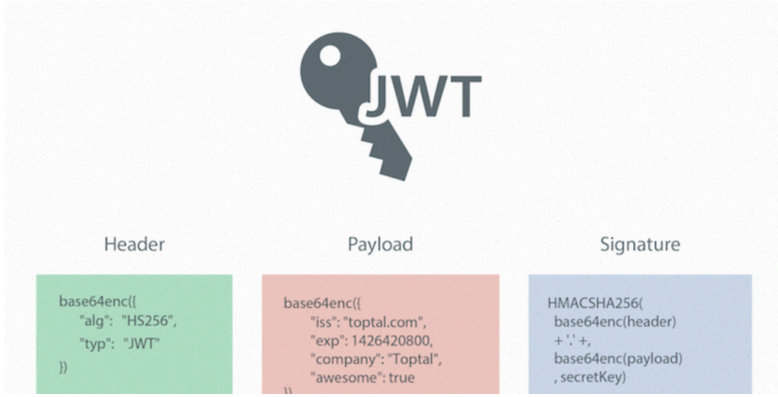
- **Przechwytywanie danych** – atakujący może uzyskać dostęp do wrażliwych informacji, takich jak hasła, numery kart kredytowych czy dane logowania.
- **Modyfikacja danych** – atakujący może manipulować przesyłanymi danymi, np. zmieniać zapytania API, wprowadzać złośliwy kod.
- **Przejęcie sesji** – przechwycenie ciasteczek sesyjnych lub tokenów uwierzytelniających, co pozwala na nieautoryzowany dostęp do aplikacji.
- **Phishing** – atakujący może przekierować użytkownika na fałszywą stronę, zbierając dane logowania.

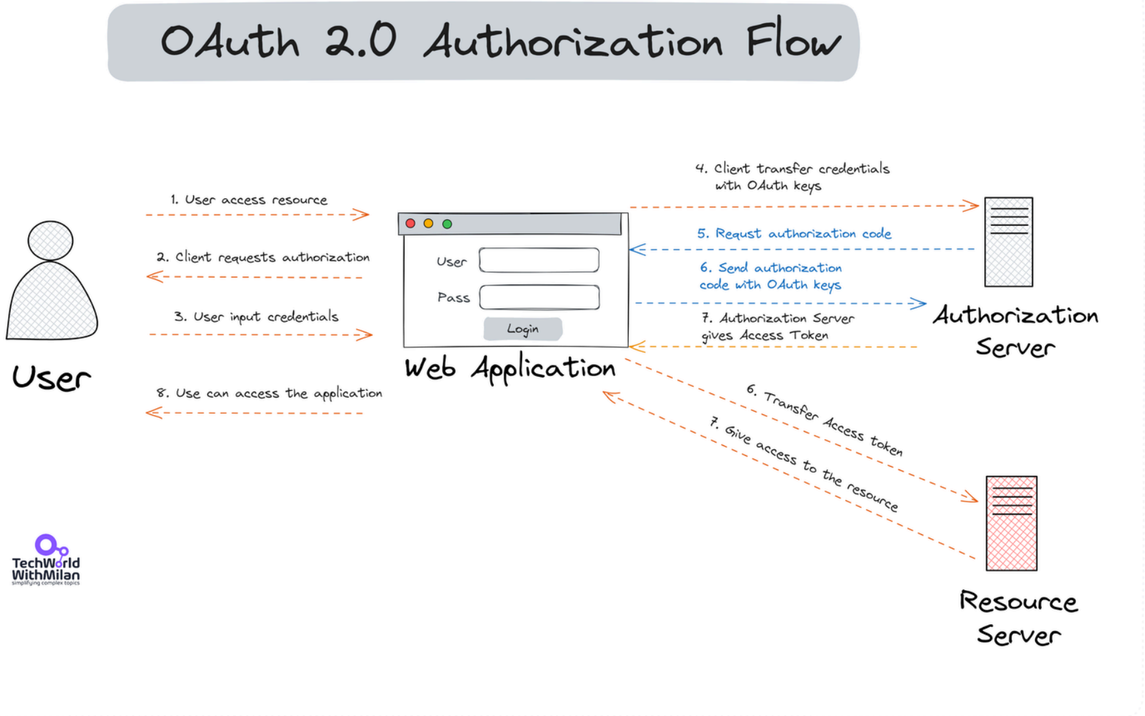


Standardy i najlepsze praktyki zabezpieczania API

Uwierzytelnianie i autoryzacja

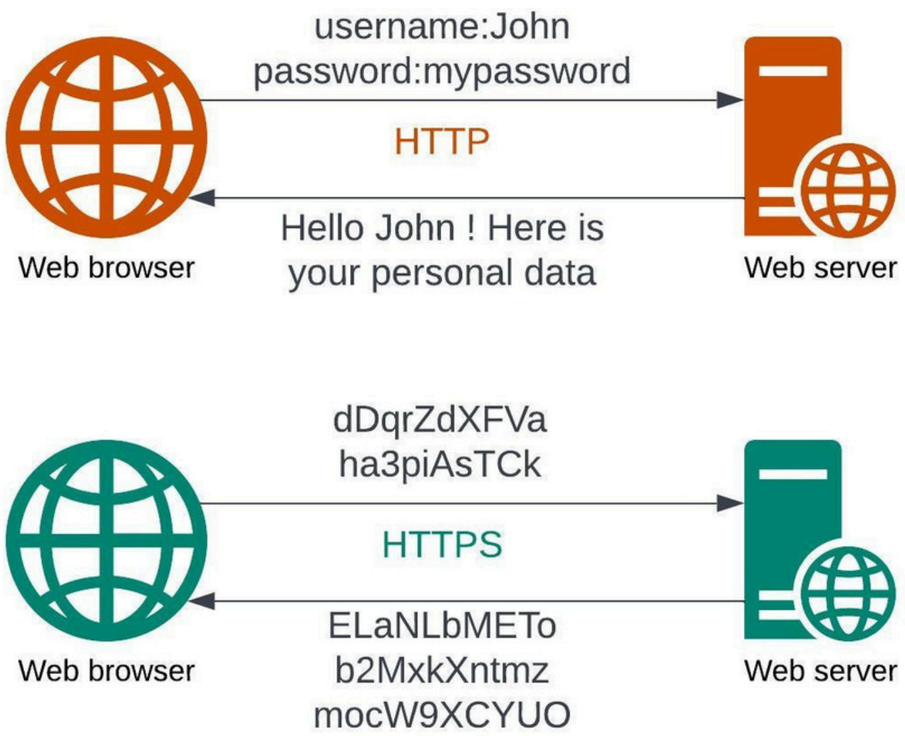
- **JWT (Json Web Token)**
  - **Token w formacie JSON zawierający zaszyfrowane informacje o użytkowniku i czasie ważności.**
  - **Zastosowanie: Przesyłany w nagłówku Authorization jako Bearer Token.**
  - **Zalety: Brak potrzeby przechowywania sesji po stronie serwera, możliwość autoryzacji.**
  - **Wady: Wyciek tokenu oznacza ryzyko jego nadużycia przez atakujących.**
- **OAuth 2.0**
  - **Opis: Standard uwierzytelniania oparty na tokenach dostępu (Access Tokens). Umożliwia delegowanie uprawnień bez konieczności ujawniania danych logowania użytkownika.**
  - **Przykład: Logowanie za pomocą konta Google lub Facebook.**
  - **Zalety: Wysoki poziom bezpieczeństwa, szerokie możliwości konfiguracji.**
  - **Wady: Większa złożoność implementacji.**
- **Klucze API (API Keys)**
  - **Opis: Proste klucze unikalne dla użytkownika lub aplikacji.**
  - **Zastosowanie: Przesyłane w nagłówkach lub parametrach zapytań.**
  - **Zalety: Prosta implementacja.**
  - **Wady: Brak tożsamości użytkownika – klucz jest jedynie identyfikatorem aplikacji.**





Szyfrowanie i TLS

- **Znaczenie stosowania HTTPS w komunikacji - dobrze jest wymuszać stosowanie komunikacji tylko przez HTTPS**
- **Konfiguracja TLS (np. wyłączanie starszych wersji protokołów).**
- **Ochrona przed atakami typu Man-in-the-Middle**

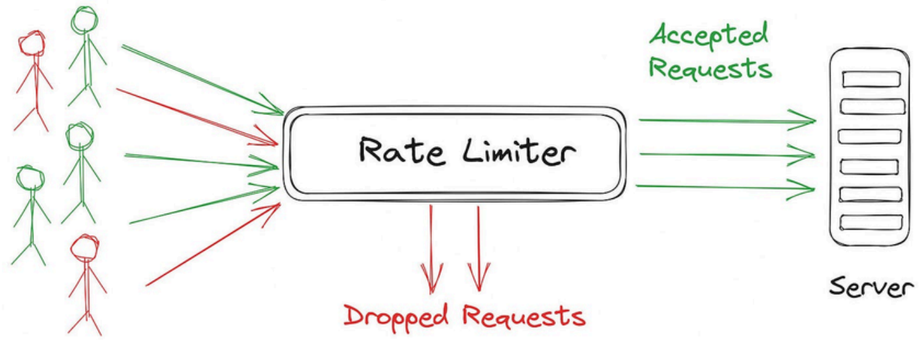


CORS (Cross-Origin Resource Sharing)

- Zarządzanie dostępem między domenami: Kontrola dostępu do zasobów przez różne źródła (np. inne domeny, protokoły).
- Nagłówki bezpieczeństwa: Definiowanie polityk za pomocą nagłówków HTTP, takich jak Access-Control-Allow-Origin.
- Obsługa zapytań preflight: Weryfikacja zapytań metodami HTTP (np. OPTIONS) przed wykonaniem żądania.
- Zapobieganie nadużyciom: Ochrona przed atakami typu CSRF i nieautoryzowanym dostępem.

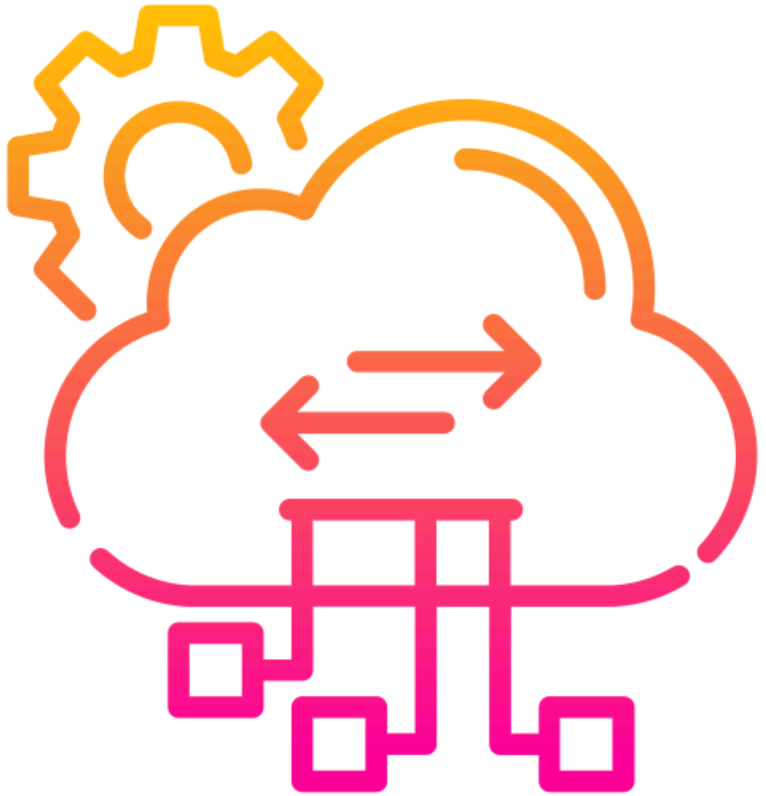
Rate limiting i monitorowanie API

- Wprowadzenie mechanizmów ochrony przed nadużyciami.
- Narzędzia do monitorowania aktywności (np. API Gateway).
- Wykrywanie anomalii w zachowaniu API (np. nagły wzrost ruchu z jednego IP)
- Automatyczne blokowanie podejrzanych adresów IP po przekroczeniu limitów



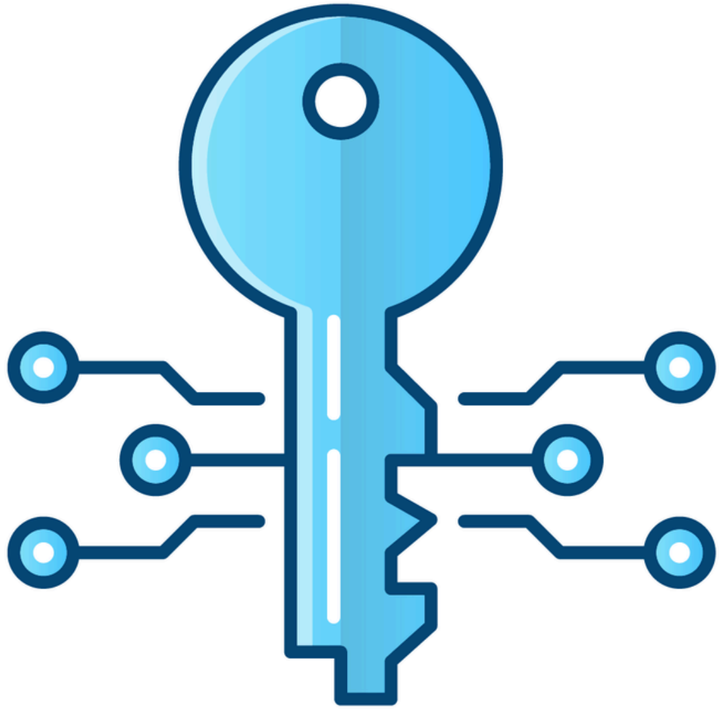
Walidacja danych wejściowych i wyjściowych

- Filtry danych
- sanitizacja
- ograniczenia zakresu wartości.
- Walidacja biznesowa (np. sprawdzanie czy wiek jest realny, czy email ma poprawny format)
- Ochrona przed wstrzykiwaniem kodu SQL i innymi atakami przez odpowiednią walidację



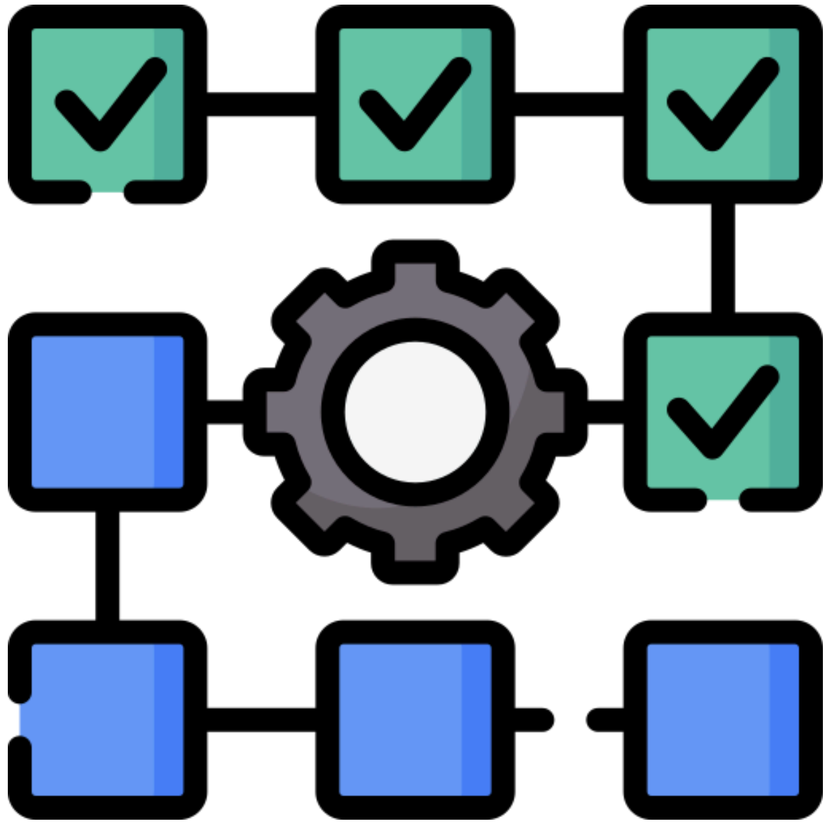
Bezpieczne zarządzanie kluczami API

- Rotacja kluczy
- przechowywanie ich w bezpiecznym miejscu (np. HashiCorp Vault).
- Regularna zmiana kluczy API w określonych odstępach czasu (np. co 90 dni)
- Stosowanie różnych poziomów uprawnień dla kluczy API (np. read-only, full access)
- Monitorowanie użycia kluczy i wykrywanie nieprawidłowości w dostępie



Testy bezpieczeństwa

- Automatyczne testy penetracyjne (np. OWASP ZAP, Burp Suite).
- Regularne przeglądy kodu
- audyty bezpieczeństwa
- Ciągłe skanowanie podatności w czasie rzeczywistym (np. przy każdym wdrożeniu)



THANK YOU

Bibliografia

- www.sekurak.pl
- https://appmaster.io/pl/blog/rest-api-przyklady
- https://owasp.org/www-project-api-security/
- https://owasp.org/API-Security/editions/2023/en/0x11-t10/
- https://web.dev/articles/security-headers?hl=pl
- https://medium.com/blog-transparent-data/bezpiecze%C5%84stwo-aplikacji-webowych-owasp-kr%C3%B3tki-przewodnik-po-b%C5%82%C4%99dach-2021-99b63fc0b5b1