BASIC AUTHENTICATION, OAUTH2, JWT - CO WYBRAĆ?

EDYCJA DRUGA

PRZMYSŁAW BYKOWSKI
BYKOWSKI.PL

Spis treści

O autorze

Przemysław Adam Bykowski

Teacher | Developer | Author | Consultant

Zajmuję się organizowaniem warsztatów oraz szkoleń z zakresu tworzenia i rozwoju oprogramowania. Specjalizuję się w Spring Framework. Odpowiednie kierunkowe wykształcenie zestawione z wieloletnim doświadczeniem pozwalają mi świadczyć usługi na najwyższym możliwym poziomie.

Dzielenie się wiedzą sprawia mi niemałą przyjemność. Niejednokrotnie już współpracowałem z Odbiorcami w różnym wieku oraz o różnym zakresie umiejętności. Działalność akademicka sprzyja poznawaniu i wdrażaniu coraz to efektywniejszych patentów pomocnych w nauce innych, natomiast codzienna praca z projektami programistycznymi umożliwia mi aktualizowanie wiedzy z zakresu najnowszych rozwiązań technologicznych.

Zawodowo i z wykształcenia łączę sobie wiedzę z zakresu IT i zarządzania, co pozwoliło mi wygrać lub być laureatem wielu konkursów programistycznych a także biznesowych na szczeblu krajowym.

Jestem zawsze pełny pozytywnej energii do działania! Sport mnie nakręca 😉

Moją Twórczość znajdziesz również:

BLOG → http://bykowski.pl

YOUTUBE → https://www.youtube.com/channel/UCjWnQvpQgSRLeDEYQC0ZuLg

FACEBOOK → https://www.facebook.com/programujzbykiem

GRUPA → https://www.facebook.com/groups/byczazagroda/

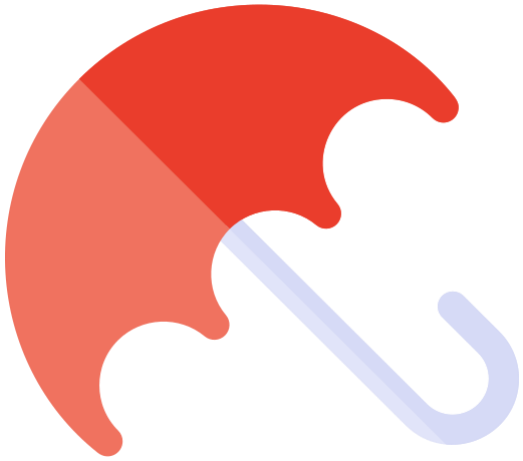
GITHUB → https://github.com/bykowski

Wstęp

Zabezpieczanie systemów informatycznych ma niebagatelny wpływ na bezpieczeństwo:

- danych użytkownika;
- przechowywanych informacji;
- stabilnego działania usługi;
- wielu innych rzeczy za który odpowiedzialny jest Twój system.

W zależności od przeznaczenia systemu informatycznego należy odpowiednio dobierać narzędzia i sposoby, które pozwolą na zabezpieczenie tworzonego sytemu. W tej książce poznasz wybrane sposoby zabezpieczenia aplikacji internetowych.



Odpowiedzialność

Dlaczego zabezpieczanie jest tak istotne? Jeśli tworzysz aplikację internetową, do której dostęp mają mieć użytkownicy, to musisz pamiętać, że ciąży na Tobie odpowiedzialność właściwego przechowywania ich danych! Informacje o sposobie zabezpieczania danych należy ująć w polityce bezpieczeństwa umieszczonej w serwisie, który tworzysz.

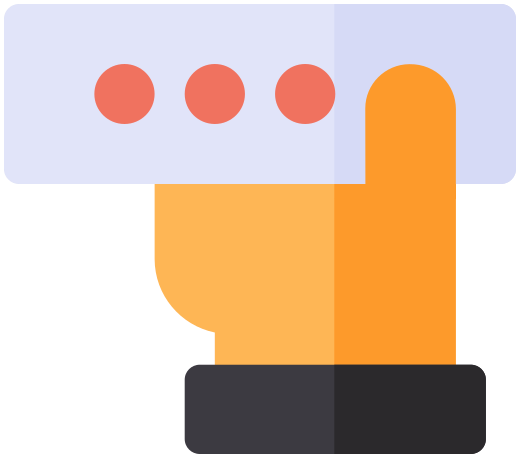
Jeśli Twoja usługa stanie się ofiarą ataku hackerskiego i dojdzie do wycieku danych, to poszkodowany użytkownik ma prawo ubiegać się o odszkodowanie – gwarantują mu to przepisy ujęte w ustawie RODO. Twoją linią obrony jest udowodnienie, że podjąłeś wszystkie środki zapobiegawcze stosując odpowiednie zabezpieczenia i nie ponosisz winy za zaistniałą sytuację. Oczywiście dodatkowo spełnić trzeba kilka formalnych warunków. Między innymi w przypadku odnotowanego ataku masz 72 godziny na poinformowanie użytkowników i specjalnych organów o ataku. Ale nie o tym dzisiaj! Dziś pokażę Ci jak wykonać zabezpieczenia aplikacji informatycznych. A jeśli chcesz wiedzieć więcej na temat formalności jakie musi spełnić bezpieczna aplikacja, to napisz do mnie wiadomość, że chcesz poczytać o tym w kolejnym ebooku 😊

Jednak odpowiedzialność jaka spoczywa na twórcach aplikacji ma nie tylko wypływać z bojaźni przed karą. Przede wszystkim jest to odpowiedzialność moralna każdego specjalisty, aby realizowane przez nas projekty były na jak najwyższym poziomie, zachowując przyjęte standardy, przepisy przy wykorzystaniu najlepszych znanych nam sposobów i narzędzi. Jednak do takich działań jest potrzebna nam świadomość na temat ich istnienia - o tym, jak to robić, dowiesz się właśnie z tego ebooka.

Kiedy system jest bezpieczny?

„System informatyczny jest bezpieczny, jeśli jego użytkownik może na nim polegać, a zainstalowane oprogramowanie działa zgodnie ze swoją specyfikacją.”

Simson Garfinkel, Gene Spafford Practical Unix and Internet Security



Basic authentication

Jest powszechnie znanym i stosowanym zabezpieczanie usług przed niepowołanym dostępem. Spotykany w większości serwisów internetowych, takich jak Facebook, Instagram, Twitter.

Dostęp do zasobu odbywa się poprzez podanie loginu i hasła. Wcześniej użytkownik musi dokonać rejestracji w danym serwisie, gdzie weryfikacja odbywa się z wykorzystaniem adresu mailowego.

Mechanizm działania

1. Użytkownik wysyła żądanie do serwera. W przypadku, kiedy zasób wymaga autoryzacji, klient otrzymuje odpowiedz 401 (brak autoryzacji).
2. Użytkownik wysyła kolejne żądanie wraz z poświadczeniami, takimi jak login i hasło, zakodowane z wykorzystaniem Base64. Wartość ta przekazywana jest w nagłówku *Authorization.*
3. W przypadku, kiedy na serwerze nie ma SSL, wówczas żądanie przekazywane jest z wykorzystaniem protokołu HTTP. Żądanie takie nie jest bezpieczne i można je podejrzeć.

4. W przypadku, kiedy na serwerze działa obsługa SSL, wówczas żądanie przekazywane jest poprzez HTTPS.

HTTP + SSL = HTTPS. Obecnie jednak protokół SSL jest zastępowany przez TSL (rozwinięcie protokołu SSL).

Uwaga

Garść ciekawostek, o które musisz zadbać dla bezpieczeństwa użytkowników swojej aplikacji.

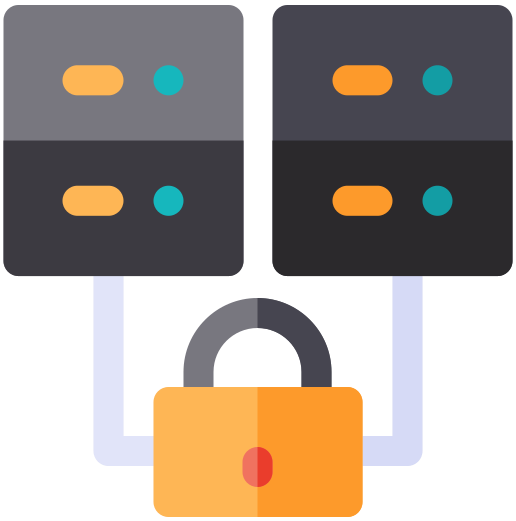
1. Login użytkownika – loginem użytkownika nie powinien być adres mailowy. Tak samo odzyskiwanie hasła nie powinno być realizowane poprzez podanie adresu e-mail. Wykorzystując tak działający serwis, podając adres e-mail można dokonać sprawdzenia, czy użytkownik ma konto w systemie. W trosce o bezpieczeństwo prywatności użytkownika nie powinniśmy udzielać informacji, czy użytkownik ma konto w serwisie. Pamiętaj, że adres e-mail stanowi daną osobową! Rozwiązaniem jest zaproponowanie użytkownikowi unikalnego loginu, na podstawie którego będzie dokonywał logowania, oraz przypominania hasła.
2. Odzyskiwanie hasła – w przypadku, kiedy użytkownik zapomni hasło, to w ramach mechanizmu przypominania hasła **nie powinniśmy wysyłać mu nowego hasła na skrzynkę e-mailową**. Wiadomości mailowe nie są szyfrowane i każdy ma możliwość ich podejrzenia. Najlepszym rozwiązaniem jest przesłanie nowego linku do naszego serwisu, na którym dokona zmiany hasła.
3. Należy dbać o to, aby w serwisie, na którym dochodzi do przekazywania hasła był wdrożony certyfikat SSL.

   Przedstawione punkty często są zaniedbywane w wielu serwisach.

Kiedy stosować?

W przypadku serwisów zorientowanych na użytkowników, takich jak:

a. portale społecznościowe;
b. sklepy internetowe;
c. blogi.



OAuth 2.0

Jest otwartym protokołem pozwalającym autoryzować użytkowników za pośrednictwem innego zaufanego serwisu. Przykładem jest system informatyczny, która wymaga utworzenia konta użytkownika. Natomiast jako twórcy aplikacji zezwalamy użytkownikom, aby zalogowali się w naszym systemie z poziomu innego serwisu np. Facebook lub Google.

Mechanizm działania

W OAuth2 występują powiązane ze sobą komponenty. Należą do nich:

- *resource owner* – właściciel zasobu, najczęściej użytkownik;
- *client* - zwykle aplikacja, która chce uzyskać dostęp do *resource server*;
- *authorization server* - jest to serwer wydający *tokeny* dostępu do klienta po pomyślnym uwierzytelnieniu właściciela zasobu i uzyskaniu autoryzacji;
- *resource server* - serwer obsługujący chronione zasoby, zdolny do akceptowania i odpowiadania na chronione żądania zasobów przy użyciu tokenów dostępu.

Ponadto w OAuh2 ogromne znacznie ogrywają tokeny. Tokeny to losowe ciągi znaków generowane przez serwer autoryzacji. Można wyróżnić typy tokenów:

- *Access Token* - wysyłany z każdym żądaniem, zwykle posiadający określoną żywotność np. godzinę;
- *Refresh Token* - kest używany do zdobycia nowego *Access Token*. Nie jest wysyłany z każdym żądaniem, zwykle żyje dłużej niż token dostępu.

Zasadę działania najłatwiej przedstawić na scenariuszu. Przykładowy użytkownik – Adam (*Resource owner)*, chce napisać post w Twoim serwisie jakim jest forum (*client)*. Dostarczasz Adamowi narzędzie, które pozwoli mu się zalogować i napisać post poprzez zalogowanie się z konta Facebook (W wielu przypadkach jak i w tym *authorization server i resource server* to ten sam provider*)*.

Użytkownik Adam wybiera przycisk zaloguj i wówczas pojawia mu się formatka logowania ze strony Facebook (*resource server).* Po prawidłowym podaniu loginu i hasła, Facebook (*resource server)* przesyła *Access Token* do klienta, który posiada informacje dotyczące użytkownika. Na podstawie tokenu mamy zidentyfikowanego Adama jako użytkownika FB. Teraz możemy dać mu dostęp do napisania postu.

Access Token ma przeważnie określoną długość życia, po upływie którego traci on swoją ważność i autoryzacja takim tokenem nie będzie możliwa. Aby zapewnić ciągłość działania prac Adam, to w momencie wygaśnięcia *Access Token'u* wysyła się Refresh Token, który wygeneruje nowy *Access Token.*

Uwaga

OAuth2 nie jest wygodnym rozwiązaniem, które można zawsze wykorzystać. Jego problemem jest silne uzależnienie się od serwisu zaufanego (w przypadku Adama był to Facebook). Przed implementacją musimy postawić sobie pytania – czy to dostawca godny zaufania? Co, jeśli zmieni on politykę korzystania lub API?

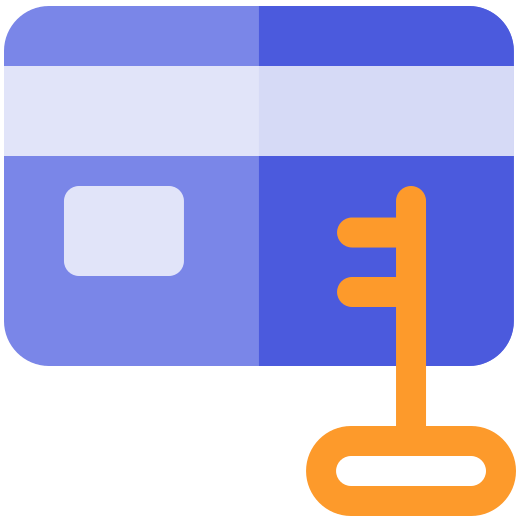
Kiedy stosować?

Cele stosowania są takie same jak w przypadku *Basic authentication*. Zasadniczą przewagą stosowania tego protokołu nad *Basic authentication* jest to, że nie musimy zapewniać mechanizmów takich jak:

- rejestracja;
- odzyskiwanie hasła;
- bezpieczne przechowywanie danych użytkowników.

Wszystkie te odpowiedzialności leżą po stronie zaufanego serwisu. Dodatkową dla użytkowników jest to, że nie muszą tworzyć nowego konta w Twoim serwisie, lecz skorzystają z istniejącego w innym serwisie typu Gogole lub Facebook.

Oczywiście w zależności od polityki tworzonego systemu informatycznego możesz połączyć *Basic authentication* oraz *OAuth2,* aby dać użytkownikowi wolną rękę w sposobie wyboru autoryzacji.



JSON Web Token (JWT)

JSON Web Token jest drugim najpopularniejszym obok Basic Authentication (autoryzacja z wykorzystaniem loginu i hasła) sposobem autoryzacji. Stanowi standard bazujący na JSON, który umożliwia wymianę tokenów. Na przykład serwer może wygenerować token, który przekazuje informacje „zalogowany jako administrator", a następnie dostarczyć go klientowi. Klient może użyć tego tokena, aby udowodnić, że jest zalogowany jako administrator.

Brzmi znajomo? To wszystko dlatego, że OAuth2 jest protokołem, który wykorzystuje mechanikę działania JWT!

Mechanizm działania

Budowa JWT ma określoną strukturę. Posiada on 3-elementową budowę.

- Header – przechowuje on informacje na temat algorytmu szyfrowania oraz typie tokena.

- Payload – dowolny przekazywany ładunek. Najczęściej są w nim przechowywane informacje na temat roli i praw użytkownika (tzw. *claims*), czy też długości życia dla tokena.
- Verify – podpis cyfrowy, który składa się z zaszyfrowanego Headera i Paylodu. Stanowi on sumę kontrolną.

Token, który trafia do serwera od klienta jest parsowany, a następnie weryfikowany pod kontem prawidłowości, uprawnień, ważności, TTL i innych.

Na podstawie tokenu, który jest przechowywany po stronie klienta, można uzyskać dostęp do zasobów serwera. Najczęściej JWT jest stosowany do autoryzacji przy dostępie do API.

Przyjmijmy, że nasz system podzielony jest na backend i frontend. Frondend generuje JWT, który posiada wszystkie składowe opisane powyżej np. użytkownik Adam, rola administrator. Token ten jest następnie szyfrowany kluczem symetrycznym lub asymetrycznym. Frontend wysyła zapytanie do backendu. Backend z wykorzystaniem tego samego klucza dokonuje parsowania tokena. Z niego wyciąga informacje, że użytkownikiem jest Adam z prawami administratora i tym samym daje mu dostęp do wszystkich przywilejów tej roli.

Uwaga

Wdrożenie mechanizmu autoryzującego z wykorzystaniem JWT nie jest proste. Sam proces implementacyjny wymaga przewidzenia wielu rzeczy, które wydarzyć mogą się po drodze. Istnieje, wiele materiałów dotyczących trudności w wykorzystywaniu JWT takich jak łatwość odzyskania hasła z JWT w określonych przypadkach, komplikacja użytkowania i częste ataki czasowe na podpis.

Kiedy stosować?

JWT jest najczęściej stosowany:

- do zabezpieczania API lub innych usług sieciowych;
- często idzie w parze z OAuth2.

Podsumowanie

Basic authentication i JSON Web Token to najpopularniejsze sposoby autoryzacji, które świetnie się uzupełniają. W przypadku budowy systemu, który przeznaczony jest z myślą o użytkownikach to dobrym wyborem jest wykorzystanie Basic authentication.

Jeśli tworzysz API i chcesz, aby Twoi klienci mieli wygodny, ale kontrolowany przez Ciebie dostęp do niego, to koniecznie zapoznaj się czym jest JWT, oraz zastosuj go w praktyce!

OAuth2 to natomiast protokół, który bazuje na wymianie komunikatów w formacie JSON. Z jego wykorzystaniem można zaimplementować Single Sign-On (tzw SSO), gdzie użytkownik, może zautoryzować się w serwisie zewnętrznym np. za pomocą konta w serwisie Google.

Praca domowa

Mam nadzieje, że ten ebook dostarczył Ci sporej dawki wiedzy i przede wszystkich chęci do dalszego działania. Zabezpieczanie usług sieciowych jest tematem, który z punktu widzenia systemów informatycznych jest ponadczasowy. Zawsze musimy o nie zabiegać i często aktualizować.

Wiele tematów nie zostało wyjaśnionych w tym podręczniku, dlatego też zachęcam Cię do przewertowania wiedzy na temat innych terminów, które wykorzystywałem lub nie udało się ich poruszyć:

- Saml;
- Passport;
- Paseto;
- JOSE;
- JWE;
- JWS;
- OpenID;
- Szyfrowanie asymetryczne;
- Szyfrowanie symetryczne.

Jeśli spodobał Ci się ten ebook to udostępnij go swojemu znajomemu. Jeśli Tobie przyniósł wartość, to innym również. Na pewno znajomy ucieszy się, że o niego dbasz i on kiedyś zadba o Ciebie.

Pierwszy ebook już za nami 😊 Jeśli chcesz kolejne publikacje z tej serii, to zapraszam Cię do kontaktu przemek@bykowski.pl lub najlepiej dołącz do Naszej grupy na FB!