2022

Research Document

INDIVIDUAL PROJECT

MITOV,LACHEZAR

Contents

Abstract

Nowadays, the Internet has been widely adopted by everyone. And with the mass adoption there comes questions about the security of the users. In consequence, the interest with cybersecurity has grown significantly. Henceforth, almost all web applications use authentication and for this authentication to work there needs to be a way for the front-end of the application to store user identity data in a secure fashion. One very common way of achieving that is by using JSON Web Tokens (JWT) and this paper intends to explain how to implement JWT correctly.

Document Approach

This paper is written following a concrete structure, which will be explained here. Firstly, the main question, that the document is intended to answer and thus help solve the problem from the Abstract section is:

1. How can you secure your web application properly using the JSON Web Token technology?

Secondly, the main question will be split into sub questions, which aim to answer the main question:

1. Why should you secure your web application?
2. How does implementing authentication help secure your application?
3. What is JWT?
4. How does the JWT technology work?
5. How JWT helps secure your web application?
6. How to implement the JWT technology correctly?

The document will use the DOT framework methods to answer the sub questions above. Moreover, the answers to the sub question will follow the APA formatting, so that the used sources in this document can be given credit.

Finally, a conclusion about the results of the research will be given and recommendations to show how the research can be applied to (for example) this semester's group or individual project will be written.

Why should you secure your web application?

Every application on the Internet contains huge amount of data about its customers. For instance, social media apps store so much information about the users that are on the servers that if a hacker manages to get its hands on this data there could be fatal consequences for both the users and the company behind the app. The lack of proactive security strategy can lead to spreading and escalation of malware, attacks on other websites, networks, and other IT infrastructures.

Moreover, if hacker attack spread from computer to computer, that would make it very difficult to find the origin of this attack. Henceforth, security should be one of the main priorities when making a web application.

Finally, this question was answered using the Literature study method, which follows the Literature strategy of the DOT framework, to find material and summarize it into a relevant paragraph.

How does implementing authentication help secure your application?

Authentication is part of the web security so the reasons on why you should implement one in your web application interchange with the ones said in the above paragraph. If your website authentication process is lacking, you run the risk of unauthorized users gaining access to sensitive user information. These data breaches can hurt individual users when their personal information is taken, but they can also ruin YOUR company's brand and financial health. Therefore, it is of the upmost importance to implement authentication in your web application.

Again, this question was answered following the Literature study method by identifying keywords, finding relevant information, and then summarizing the findings.

What is JWT?

Firstly, before we explain how it works, we must describe what exactly is a JSON Web Token (JWT). JWT is an open standard (RFC 7519) for securely transmitting information between parties as JSON object. In its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are:

- Header
- Payload
- Signature

Let's break down the different parts:

Header

The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA. For example:

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

Payload

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims:

- Registered claims
- Public claims
- Private claims

An example for a payload would be:

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

Signature

To create the signature part, you must take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that. For example, if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

The signature is used to verify the message wasn't changed along the way, and, in the case of tokens signed with a private key, it can also verify that the sender of the JWT is who it says it is. When you put all the parts together the JWT will look like this:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Through finding relevant resources and references, identifying key information, and filtering the findings this paragraph was written. This follows the Literature study method of the DOT framework.

How does the JWT technology work?

In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned. The process of using JSON Web Tokens for authentication can be narrowed to 4 steps:

1. User sign-in using username and password.
2. Authentication server verifies the credentials and issues a JWT signed using either a secret salt or a private key.
3. User's Client uses the JWT to access protected resources by passing the JWT in HTTP Authorization header.
4. Resource server then verifies the authenticity of the token using the secret salt/ public key.

This question uses the Lab strategy. Following the Usability testing method I implemented the JWT in my individual project and through experience with this technology I managed to construct this paragraph.

How JWT helps secure your web application?

As we said earlier, the JSON Web Tokens consists of 3 parts (Header, Payload and Signature). And the Signature is the most important part. It consists of private key or a *secret*, which is used by the issuer to sign the JWT. The receiver of the JWT will verify the signature to ensure that the token hasn't been altered after it was signed by the issuer. This makes it very secure and reliable because it is difficult for unauthenticated sources to guess the signing key and attempt to change the claims within the JWT.

Moreover, JWT is stateless, which means all the data that are needed to verify the token and identify the user are stored in the token itself. There is no need to maintain any record of the token in the server, like store the token in a database. This stateless nature gives us the big security benefit that comes with JWTs: The server that issues the JWT and the server that validates it does not have to be the same.

Following the Literature study method, the paragraph above was constructed by finding and judging material, looking for interesting references and summarizing the findings.

How to implement the JWT technology correctly?

Now that we know what JSON Web Token is and how it works, we will talk how to implement it correctly so that you do not make your web application vulnerable.

Firstly, since tokens are credentials, great care must be taken to prevent security issues. Moreover, tokens should not be kept longer than required.

Secondly, JWT needs to be stored in a safe place inside the user's browser. If the token is stored inside localStorage, it's accessible by any script inside your page and this is as bad as it sounds; an XSS attack could give an external attacker access to the token. In order to keep the token save, they should be stored inside an httpOnly cookie. This is a special kind of cookie that's only sent in HTTP requests to the server. It's never accessible (both for reading or writing) from JavaScript running in the browser.

Finally, when implementing JWT you should always have this in mind, to make the application as secure as possible.

This question was asked by using the Lab strategy. Through personal experience of implementing the JWT technology correctly, I learned from previous unsuccessful attempts and summarized the relevant findings.

Conclusion

Securing a web application is very important not only for the users of the application, but for the people behind this software as well. That is why developers should always adopt proactive security strategy. One way of achieving this strategy is by using JWT technology. Moreover, JWT makes it so that it is not rocket science to implement this correctly and it offers great security. This makes this technology so trusted and the go-to, when it comes to web security, for many developers.

Bibliography

Copes, F. (2021, June 17). *JWT authentication: Best practices and when to use it.* Retrieved from blog.logrocket.com: https://blog.logrocket.com/jwt-authentication-best-practices/

CWatch. (2021, December 2). *Secure Website: Why is Website Security Important?* Retrieved from cWatch.com: https://cwatch.comodo.com/blog/website-security/why-is-website-security-important/

JWT. (n.d.). *Introduction to JSON Web Tokens.* Retrieved from jwt.io: https://jwt.io/introduction#:~:text=JSON%20Web%20Token%20(JWT)%20is,because%20it%20is%20digitally%20signed.

Kumar, S. (2022, April 25). *How JWT (JSON Web Token) authentication works?* Retrieved from DEV.com: https://dev.to/kcdchennai/how-jwt-json-web-token-authentication-works-21e7#:~:text=Authentication%20server%20verifies%20the%20credentials,the%20secret%20salt%2F%20public%20key.

Martines, J. C. (2020, July 31). *A Brief Introduction to Securing Applications with JWT.* Retrieved from Live Code Stream: https://livecodestream.dev/post/a-brief-introduction-to-securing-applications-with-jwt/

Swoopnow. (2020, July 8). *Website Authentication: The Complete Guide with FAQs.* Retrieved from swoopnow.com: https://swoopnow.com/website-authentication/