DOI: https://doi.org/10.30837/csitic52021231864

Securing Bearer token in OAuth2.0

Oleksandr Sievierinov[1]

[1]Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine, e-mail: oleksandr.sievierinov@nure.ua

Oleh Kholosha[2]

[2]Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine, e-mail: oleh.kholosha@nure.ua

Short abstract. *This article provides information about the bearer token in Oauth2.0. Considered the threats to which the bearer token is exposed in Oauth, suggested recommendations for the safe use of this type of token.*

Keywords: *Oauth, registration, authorization, security, token, bearer token.*

I. Introduction and Problem Statement

Today, when humanity actively uses the Internet, every person is registered on various sites and resources. Registration is required to use any web-service and users are faced with the issue about safe storing credentials [1].

OAuth 2.0 is an authorization protocol that allows one service to grant user access rights to another service. The protocol allows not to store the login and password on the application server and gives the user limited set of rights [2, 3].

II. Problem Solution and Results

Bearer token is defined by OAuth protocol as one of the form of credentials, which can be used by any party, regardless of who that party is. A Bearer token is a random sequence that an authentication server provides to all users. When the user logs on to the application, the authentication server generates the token. A user that uses bearer token does not need to prove his identity. The bearer token is the most used type of token in OAuth.

The bearer token has the same characteristics as session cookies. If an attacker hijacks an access token, the attacker accesses all features related to the token. The OAuth bearer token is exposed to the following threats:

1. Token forgery. The attacker creates a token or changes the real one, which allows him to access the information. For example, the availability of administrator data.
2. Reusing token. The attacker uses a token that is out of use. In this case, the resource server returns an error rather than real data.
3. Token redirection. The attacker uses a token created for use on a single resource server to access another server, which identifies the token as authentic to it.
4. Decryption of tokens. The token contains confidential information about the system. Due to decryption, the attacker receives important data about the system.

It is necessary to ensure that access tokens, which are treated as bearer tokens, are secured. The peculiarity of OAuth protocol is that access tokens are protected by end-to-end privacy – SSL/TLS. This security system is a protocol that provides a secure Internet connection [4].

The user token must include a minimum amount of rights. For example, if all users request information about the resource owner profile, only the profile scope should be provided. These minimal capabilities, which are limited for the user, allow protecting information. It is recommended to store access tokens in temporary memory to reduce the risks of certain attacks. If the attacker gets access to the database, he will no longer receive information about access tokens[5].

The authorization server should store access token hashes, not token text. Then, if a database with all access parameters is stolen, the information will not be able to be used. It is recommended to store passwords with hash salting. At the same time, it is better to reduce the validity of access tokens to minimize the risk of data leakage from one access token. Therefore, if an attacker reveals the contents of a token, the short life of the latter makes it useless. If long access to the resource is required, a refresh token on the authorization server should be issued by the client.

The resource server must be designed to limit token scope, respecting the “minimal privilege” principle. The resource server must validate all tokens. Although it is recommended to cache the status of the token, it must also assess the disadvantages and benefits of stored token data. Rate limiting and other API protections techniques help protect information from an attacker [6].

III. Conclusions

Bearer tokens greatly simplify the operation of OAuth protocol for authorization. They allow to easily and correctly implement the protocol. However, this ease of use requires some token protection.

1. Security of access tokens should be provided by safe transportation tools, for example, TLS.
2. The requested information must be minimal.
3. The authorization server must store token hashes, not its text.
4. An authorization server should reduce the lifetime of the access token to minimize the risks of misuse of information.
5. The resource server must place access tokens in temporary (i.e. transient) memory [7].

REFERENCES

[1] Richard Smit, Authentication methods, // Authentication: From Passwords to Public Keys. 2008. C. 190 – 210

[2] RFC 6749, The OAuth 2.0 Authorization Framework [Електронний ресурс]: Режим доступу: https://tools.ietf.org/html/rfc6749

[3] Власов, А.В., О.В. Сєвєрінов, and О.В. Слиш. Впровадження децентралізованої системи ідентифікації. НТУ «ХПІ», 2020.

[4] Justin Richer Antonio Sanso, OAuth 2 in Action, March 2017 Publisher(s): Manning Publications, ISBN: 9781617293276

[5] Нігель Чепмен, Classification of methods of authorization and authentication, // Authentication and Authorization on the Web. 2012. C. 140-153

[6] Ертем Османоглу, Identity management in the modern world, // Identity and Access Management: Business Performance. 2013. C. 97

[7] OAuth protocol specification [Електронний ресурс]: Режим доступу: https://oauth.net/2/