Information Security Stack Exchange is a question and answer site for information security professionals. It only takes a minute to sign up.

Sign up to join this community

Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top

INFORMATION SECURITY

What is the difference between API keys and API tokens usages?

Asked 2 years, 9 months ago Active 12 months ago Viewed 8k times

DECODE YOUR FUTURE
Online Computer Science Programs
Drexel UNIVERSITY | ONLINE

8

1

From all the research that I have done, I have found that API keys are less secure than access tokens (under usage of oAuth), and should only be used for monitoring purposes.

But from what I understood, the server generates both of them. So if I were to use the same tough algorithm to create my access token, or to create the API key, wouldn't that make them similar? Examples would help the cause of understanding them better as I didn't find any.

Some of the references: https://cloud.google.com/endpoints/docs/when-why-api-key https://zapier.com/engineering/apikey-oauth-jwt/

authentication access-control oauth api oauth2

edited Aug 26 '18 at 10:07

asked Jun 14 '17 at 7:58
Elie Saad
185 1 9

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

The implementation of how the key is created, stored, used, updated, and destroyed is going to be what determines the security of it. – ISMSDEV Jun 14 '17 at 8:06

@ISMSDEV I edited the details, added only those I remember. If you care for more, let me know so I can get them. – Elie Saad Jun 14 '17 at 8:10

1 Do you know the difference between API keys and access tokens? I think you might be confused about what makes the difference between the 2. – schroeder ♦ Jun 14 '17 at 9:11

1 Sorry I didn't mean they were the same at all. I was trying to answer the "which is more secure", and by that I was trying to articulate the answer is how they are used. They are different and therefore a like for like comparison is not straight forward. – ISMSDEV Jun 14 '17 at 12:02

1 tokens let your users safely connect directly to an API, keys require secret-hiding server proxying – dandavis Jun 14 '17 at 15:47

2 Answers

8

API keys are public, by intent. They are an authorisation mechanism, not an authentication mechanism (this is mentioned in your links). It does not matter how they are *generated* but it matters how they are *handled*. In other words: "anyone with this key can enter".

So, you use API keys when you want to authorise and do not need to authenticate. You use authentication tokens, which are *secured in handling*, to authenticate the connection. In other words, "here is your unique key to allow you to enter this time".

answered Jun 14 '17 at 9:18
schroeder ♦
96.3k 38 220 248

Authorizing in the access tokens is then an option? I have been using them to authenticate and then handle the authorization of the user. Just so I know if I am doing any bad practice. [e.g. facebook OAuth2, you sign in (authentication) and it tells you to choose which permissions(authorization) to give] – Elie Saad Jun 14 '17 at 11:32

amazon.jobs Embrace the challenge.

2

A typical API key for a REST-ful application usually happens to be significantly less secure than the access control provided by an OAuth JWT (JSON Web Token) for reasons pertaining to application layer protocol messaging (ordering, syntax, data unit protection--or lack thereof), as opposed to protection resulting only from the use of a particular cryptographic algorithm, mode and/or key size. However, I wouldn't be surprised if API key string generators were more predictable than OAuth access token string generators. API keys are often sent as HTTP **GET** query parameters within an HTTP request's first line as shown here with the Google Maps

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
<script async defer src="https://maps.googleapis.com/maps/api/js
key=YOUR_API_KEY&callback=initMap" type="text/javascript"></script>
```

Due to the fact that the API key string is being passed as an HTTP **GET** query parameter, it is much easier for intermediate web servers (including proxies), and browsers with client-side scripting languages such as JavaScript or ActionScript to gain read and/or write access to the API key. Compare this to other types of HTTP actions such as **PUT** or **POST** where the query parameters are more tightly concealed from the aforementioned technologies. Almost all web server software will write the **src** attribute value in the **script** tag above to **access_log** and/or **error_log** files including the API key, as the query parameter variable values are part of the CGI (Common Gateway Interface) environment variables: **SCRIPT_PATH** and **QUERY_STRING**. Refer to CWE-598 for more information.

OAuth access tokens on the other hand, are generated on a per-session basis. Being granted an access token by a secure authentication provider will not occur, until the provider has received proof that the requesting user is entitled to requested privileges; such proof might be established through knowledge of credentials (i.e. a corresponding username and password pair.) Other times, access control might be more restrictive and access tokens are only provided for a small sub-set of privileges within a particular app/site/API sub-component, area of operation, control sphere, etc. The permissions ultimately granted to the end user can be as fine-grained as the system administrator wishes. Note that access tokens are programmed to expire after a set amount of time and are capable of providing discretionary access control between various users/groups, privileges/capabilities, etc.

Access tokens are often transferred outside of the URL in the HTTP request header's Authorization field, for example. Sometimes, custom authentication framework implementations will cause the token to be transmitted within a cookie that has the **HttpOnly**, **Secure** and **SameSite** flags enabled--or as a custom HTTP request header such as **X-Auth-Token** as publicly documented for Oracle's Cloud Storage SaaS: Oracle's Cloud Storage Service API:

It is extremely rare for HTTP request headers and cookie values to be logged by web browser/server software; they're also more difficult to access programatically due to CORS (Cross Origin Resource Sharing.) In comparison, the API keys passed as HTTP GET parameters can be extracted with client-side JavaScript from the DOM (Document Object Model).

For these reasons, the complexity required to obtain access tokens from an authentication framework such as OAuth is much higher than what is needed to log the usage of an API key. Furthermore, the robustness of authentication and authorization frameworks allows the access token to be encapsulated within the HTTP protocol in ways that it is rather difficult to view or tamper with the token.

answered Jun 14 '17 at 9:23

Derek Callaway
71 ● 3

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.