Jacob Lipps

Box # 1734

Professor Mason

20 March 2018

Web API Authentication

Web API Authentication is a very interesting field and a topic that is not really mentioned that much. But, it is a very important topic, so it should be brought up more. This paper will dive more deeply into the topic of Web API Authentication, and hopefully, it will teach you something new and make you want to learn more about the topic. Something you do not want to get confused with, is that Web API Authentication and Web API Security are almost the same thing.

To start off, we will first look at the foundation of Web API Authentication, which is API (Application Programming Interface). It has to do with computer programming. The definition of it is a group of subroutine definitions, protocols and tools that are put together to form application software. So, to be more clear, an API is a group of laid-out defined methods of “talk” between different types of software. To have a good API is very important because to a computer programmer, it helps to neatly put together all the building blocks. API’s can be used with OS or operating systems, with a DB or a database system, as well as with computer hardware and software. One API compared to another API could be very different depending on what it is designed for. But, most API’s have routines, data structures, object classes, variables, and remote calls. Some systems that it is used with are POSIX, Windows API, and ASPI. Many computer science students do not like writing or documentation. But, documentation is very important not just for API but for Web API and Web API Authentication. For instance, if you ever must look back at past notes, at least you have them and can review over your information. You can see things that you did and did not do, as well as what you need to do, to document your work. This also helps with catching any mistakes. If a problem arises, then you can look back and see what was done. Or, if you are a new joiner to the company, you can see past things that someone else did so that you do not have to start from nothing.

Next, a step up from API is Web API, which could be used as a web server and or a web browser. A Web API is on the client side as well as on the side of the server. This helps to make web programming easier because the programmers can create their own web application on top of the high-level interface.

Now for a closer look at Web API Security, which is about authenticating the programs and the user who is working with the Web API. There is AUTHN, which stands for authenticate and AUTHZ, which stands for authorize. Depending on your company, you need to secure your data and technologies a certain way. There can be differences between a bank or a pharmaceutical company, for example. With these types of topics, often things are on a “need to know access basis” regarding the AUTHN and the AUTHZ. It is extremely important to have the correct AUTHN to correctly identify the user. Also, it is crucial to have the correct AUTHZ where the program correctly identifies the status or the level of information that the person is supposed to have. Or, just so the user has what they need to do their work, and nothing more.

There are three main ways to authenticate and to authorize, they are, Static string, Dynamic tokens, and User-delegated tokens. Static strings are one of the most basic types of authentication. This is because it is one of the easiest types of authentication. The easiest types of authentication can be the weakest types of security. This is because with the Static strings, every single password and each message is sent across in the clear. Basically, everything is sent across in clear text, which is very bad because packet sniffers can see what is going across in plain text. The API caller or the user uses a string for the token as the request. A more sophisticated example of this that is not basic is using an SSL, or a Secure Socket Layer. An SSL is just one example of a lower-level protocol. An SSL encrypts the packets so that the packet sniffers can no longer see what is inside of the packets. To the packet sniffers, it is just a bunch of jumbled text.

Dynamic tokens work differently. Dynamic tokens are based off of a time-based system. It has a TTL, or a time to live, which means when the time runs out, the user must get a new token (the token is only active for a period of time). It is set for a certain amount of time based off of an algorithm, so when the token expires, you will have no more access to something that you once had access to. An example of this token is the JWT or the JSON Web Token. A JSON Web token can be described as a JSON-based open standard for allowing entrance of tokens to help with a lot of claims. For example, a server can make a token that reads as “logged in as admin,” which would then be displayed to the client. Next, it would go into the logs and it could be shown as proof that the client logged in as administrator. All it shows is that a particular id or Username and Password logged in as administrator. It does not say that the person with the ID, or the person with the Username and Password logged in. But, back to the original point, the reason this is proof, is because the tokens are signed into the server key so that the client and server can say that the token is authentic. These tokens are small tokens, as well as URL-safe. Also, they are not made to be used in a web browser as a single sign-on or SSO. So, what the JWT is mostly used for is to pass the identity of the authenticated user through the identity provider and a service provider. Additionally, they are not just authenticated, but also encrypted. Other JSON-based standards that all work together are JSON Web Signatures and JSON Web Encryption. Inside of the JSON Web Token are three parts: the header, the payload, and the signature.

Finally, there is the User-delegated token, a type of authentication that is a three-legged system where applications can get to the API for the user, so that user ISs and passwords are not shown to the application. The user is the one in charge of where the token is encapsulated and the user is the one that gives or denies privilege for an application to handle the API. An example of this type of token is OAuth 2.0, which is an authorized framework that allows third-party applications to have a little access to the HTTP service. The owner is in control because the owner can allow communication between the owner and the HTTP service. Or, it could take place by the owner allowing the third-party application to automatically get the access that it needs by itself.

To summarize, there are three types of tokens under Web API Authentication. The Static string is very similar to passwords that are put in place by the API for the user. The Dynamic tokens are based on a set time that tokens can be retrieved by the caller, all from the authentication service. Lastly, the User-delegated tokens are tokens that are similar to the OAuth authentication example, which is permission that is given on the grounds of a user authentication.

Bibliography

* <https://en.wikipedia.org/wiki/Web_API_security>
* <https://www.cl.cam.ac.uk/~rja14/Papers/SEv2-c18.pdf>
* <https://oauth.net/2/>
* <http://unixpapa.com/auth/basic.html>
* <https://tools.ietf.org/html/rfc7519>
* <https://tools.ietf.org/html/rfc6749>
* <https://en.wikipedia.org/wiki/Web_API>
* <https://en.wikipedia.org/wiki/OAuth>
* <https://en.wikipedia.org/wiki/Authentication>

* <https://en.wikipedia.org/wiki/JSON_Web_Token>
* <https://en.wikipedia.org/wiki/Application_programming_interface>
* <https://en.wikipedia.org/wiki/Server_Application_Programming_Interface>
* <https://www.asp.net/web-api>
* <https://www.youtube.com/watch?v=s7wmiS2mSXY>
* <https://www.google.com/search?ei=HKuvWr32Kany5gLK8YHoCw&q=api+definiation&oq=api+definiation&gs_l=psy-ab.3..0i10k1l10.8619.12383.0.12859.15.15.0.0.0.0.132.1716.2j13.15.0....0...1c.1.64.psy-ab..0.15.1709...0j35i39k1j0i20i263k1j0i67k1.0.WzuAwNPF7QI>
* <http://www.dotnettricks.com/learn/webapi/what-is-web-api-and-why-to-use-it->
* <https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx>
* <https://www.google.com/search?ei=LKuvWvy8I67z5gKAnrHQAg&q=web+api+definition&oq=web+api+definition&gs_l=psy-ab.3..35i39k1j0i7i30k1j0i5i30k1l2j0i8i30k1l3.28467.29007.0.29706.4.4.0.0.0.0.157.502.0j4.4.0....0...1c.1.64.psy-ab..0.4.499...0i8i7i30k1j0i7i5i30k1j0i5i10i30k1j0i8i10i30k1.0.ox7fjV9xLRA>
* <https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/individual-accounts-in-web-api>
* <https://www.asp.net/web-api/overview/security>
* <http://www.dotnetcurry.com/aspnet/1223/secure-aspnet-web-api-using-tokens-owin-angularjs>