Blog

3 Common Methods of API Authentication Explained (https://nordicapis.com/3-common-methods-api-authentication-explained/)

POSTED BY KRISTOPHER SANDOVAL (HTTPS://NORDICAPIS.COM/AUTHOR/SANDOVALEFFECT/) | FEBRUARY 6, 2018 (HTTPS://NORDICAPIS.COM/API-INSIGHTS/SECURITY/)

Updated on October 19th, 2019

13

Facebook (https://www.facebook.com/sharer/sharer.php?u=https://nordicapis.com/3-common-methods-api-authentication-explained/&t=3+Common+Methods+of+API+Authentication+Explained)

27 Twitter

29 Google+ (https://plus.google.com/share?url=https://nordicapis.com/3-common-methods-api-authentication-explained/)

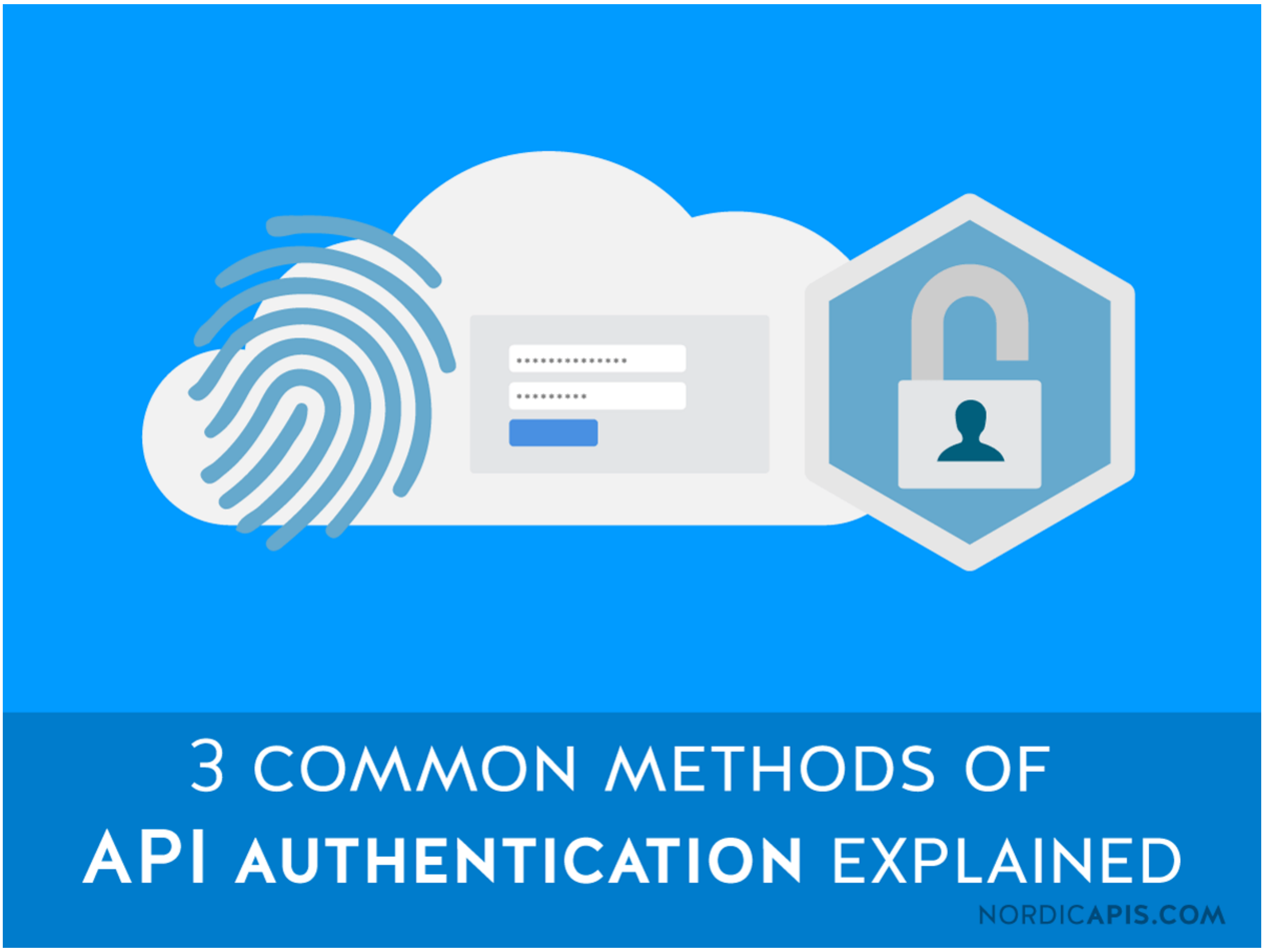
19

in LinkedIn (https://www.linkedin.com/shareArticle?mini=true&ro=true&trk=EasySocialShareButtons&title=3+Common+Methods+of+API+Authentication+Explained&url=https://nordicapis.com/3-common-methods-api-authentication-explained/)

Reddit (http://reddit.com/submit?url=https://nordicapis.com/3-common-methods-api-authentication-explained/&title=3+Common+Methods+of+API+Authentication+Explained)

Y HackerNews (https://news.ycombinator.com/submitlink?u=https://nordicapis.com/3-common-methods-api-authentication-explained/&t=3+Common+Methods+of+API+Authentication+Explained)

Total: 88



APIs handle enormous amounts of data of a widely varying type – accordingly, one of the chief concerns of any data provider is how specifically to **secure** this data. The idea that data should be secret, that it should be unchanged, and that it should be available for manipulation is key to any conversation on API data management and handling.

Today, we're going to talk about **Authentication**. Though an often discussed topic, it bears repeating to clarify exactly what it is, *what it isn't*, and how it functions.

We'll highlight **three major methods of adding security to an API** – **HTTP Basic Auth**, **API Keys**, and **OAuth**. We'll identify the pros and cons of each approach to authentication, and finally recommend the best way for most providers to leverage this power.

Authentication vs Authorization

Before we dive into this topic too deep, we first need to define what authentication actually is, and more importantly, what it's not. As much as **authentication** drives the modern internet, the topic is often conflated with a closely related term: **authorization**.

The two functions are often tied together in single solutions – in fact, one of the solutions we're going to discuss in a moment is a hybrid system of authentication and authorization. As such, and due to their similarities in functional application, it's quite easy to confuse these two elements.

The easiest way to divide authorization and authentication is to ask: *what do they actually prove?* In simple terms, **Authentication is when an entity proves an identity**. In other words, Authentication proves that you are who you say you are. This is akin to having an **identification card** – an item given by a trusted authority that the requester, such as a police officer, can use as evidence that suggests you are in fact who you say you are.

Authorization is an entirely different concept, though it is certainly closely related. In simple terms, **Authorization is when an entity proves a right to access**. In other words, Authorization proves you have the right to make a request. When you try to go backstage at a concert or an event, you don't necessarily have to prove that you are who you say you are – you furnish the **ticket**, which is de facto proof that you have the right to be where you're trying to get into.

Consider for a moment a driver's license. In many countries, a driver's license proves both that you are who you say you are via a picture or other certified element, and then goes further to prove that you have a right to drive the vehicle class you're driving. In such a case, we have authentication and authorization – and in many API solutions, we have systems that give a piece of code that both authenticates the user and proves their authorization. In such a case, we have **hybrid solutions**.

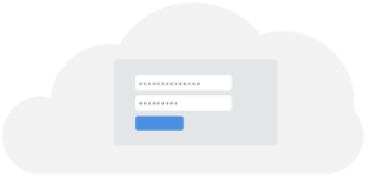
Therefore, moving forward, it's important to remember that what we're actually talking about here is a system that proves your **identity** – nothing more, nothing less.

Related: How To Control User Identity Within Microservices (https://nordicapis.com/how-to-control-user-identity-within-microservices/)

Common Methods of API Authentication

While there are as many proprietary authentication methods as there are systems which utilize them, they are largely variations of a few major approaches. These approaches almost always were developed to solve limitations in early communications and internet systems, and as such, typically use broad existent architectural approaches with novel implementations in order to allow authentication to occur.

HTTP Basic Authentication



HTTP Basic Auth is rarely recommended due to its inherent security vulnerabilities.

One solution is that of **HTTP Basic Authentication**. In this approach, an HTTP user agent simply provides a **username** and **password** to prove their authentication. This approach does not require cookies, session IDs, login pages, and other such specialty solutions, and because it uses the **HTTP header** itself, there's no need to handshakes or other complex response systems.

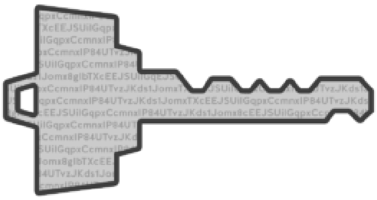
The problem is that, unless the process is strictly enforced throughout the entire data cycle to **SSL** for security, the authentication is transmitted in open on insecure lines. This lends itself to man in the middle attacks (https://nordicapis.com/fostering-an-internal-culture-of-security/), where a user can simply capture the login data and authenticate via a copy-cat HTTP header attached to a malicious packet.

Additionally, even if SSL is enforced, this results in a **slowing** of the response time. And even ignoring that, in its base form, HTTP is not encrypted (https://nordicapis.com/securing-your-datastream-with-p2p-encryption/) in any way. It is encapsulated in base64, and is often erroneously proclaimed as encrypted due to this.

HTTP Basic Authentication does have its place. In an **internal network**, especially in **IoT** situations where speed is of no essence, having an HTTP Basic Authentication system is acceptable as a balance between cost of implementation and actual function. As a general authentication solution, however, **HTTP Basic Authentication should be seldom used in its base form**.

Read more: Maintaining Security In A Continuous Delivery Environment (https://nordicapis.com/maintaining-api-security-in-a-continuous-delivery-environment/)

API Keys



API keys are an industry standard, but shouldn't be considered a holistic security measure.

API Keys were created as somewhat of a fix to the early authentication issues of HTTP Basic Authentication and other such systems. In this approach, a **unique generated value** is assigned to each first time user, signifying that the user is known. When the user attempts to re-enter the system, their unique key (sometimes generated from their hardware combination and IP data, and other times **randomly generated** by the server which knows them) is used to prove that they're the same user as before.

On one hand, this is very **fast**. The ability to prove identity once and move on is very agile, and is why it has been used for many years now as a **default approach** for many API providers. Additionally, setting up the system itself is quite easy, and controlling these keys once generated is even easier. This also allows systems to purge keys, thereby removing authentication after the fact and denying entry to any system attempting to use a removed key.

The problem, however, is that API keys are often used for what they're not – **an API key is not a method of authorization**, it's a method of authentication. Because anyone who makes a request of a service transmits their key, in theory, this key can be picked up just as easy as any network transmission, and if any point in the entire network is insecure, the entire network is exposed. This makes API keys a hard thing to recommend – often misused and **fundamentally insecure**, they nonetheless do have their place when properly secured and hemmed in by authorization systems.

Read more: Why API Keys ≠ API Security (https://nordicapis.com/why-api-keys-are-not-enough/)

OAuth

OAuth combines Authentication and Authorization to allow more sophisticated scope and validity control.

OAuth (https://nordicapis.com/why-oauth-2-0-is-vital-to-iot-security/) is a bit of a strange beast. OAuth is not technically an authentication method, but a method of **both authentication and authorization**. When OAuth is used solely for authentication, it is what is referred to as "pseudo-authentication."

In this approach, the user logs into a system. That system will then request authentication, usually in the form of a token (https://nordicapis.com/why-cant-i-just-send-jwts-without-oauth/). The user will then forward this request to an authentication server, which will either reject or allow this authentication. From here, the token is provided to the user, and then to the requester. Such a token can then be checked at any time independently of the user by the requester for validation, and can be used over time with strictly limited scope and age of validity.

This is **fundamentally a much more secure and powerful system** than the other approaches, largely because it allows for the soft establishment of **scope** (that is, what systems the key allows the user to authenticate to) and **validity** (meaning the key doesn't have to be purposely revoked by the system, it will automatically become deprecated in time).

As with anything, there are some major pros and cons to this approach. On the one hand, it's clearly superior when it comes to the level of security it can offer, and for this reason, OAuth is quickly becoming the **de facto choice** for anyone choosing to eschew API keys (https://nordicapis.com/why-api-keys-are-not-enough/). On the other hand, using OAuth for authentication alone is ignoring everything else that OAuth has to offer – it would be like driving a Ferrari as an everyday driver, and never exceeding the residential speed limits.

Those caveats in mind, OAuth is easy to set up, and it is incredibly fast.

Read more: Deep Dive Into OAuth and OpenID Connect (https://nordicapis.com/api-security-oauth-openid-connect-depth/)

The Best Option

So of these three approaches, two more general and one more specific, what is the best? That's a hard question to answer, and the answer itself largely depends on your situations. While the clear winner of the three approaches is **OAuth**, there are some use cases in which API keys or HTTP Basic Authentication might be appropriate.

That being said, these use cases are few and far in-between, and accordingly, it's very hard to argue against OAuth at the end of the day. OAuth delivers a ton of benefits (https://nordicapis.com/how-to-handle-batch-processing-with-oauth-2-0/), from ease of use to a federated system (https://nordicapis.com/api-security-the-4-defenses-of-the-api-stronghold/) module, and most importantly offers scalability of security – providers may only be seeking authentication at this time, but having a system that natively supports strong authorization in addition to the baked-in authentication methods is very valuable, and decreases cost of implementation over the long run.

What do you think? What's the best way to authenticate a user? More to the point, what do you think are the most clear use cases for using something like an API key over OAuth? Let us know in the comments below.

13

Facebook (https://www.facebook.com/sharer/sharer.php?u=https://nordicapis.com/3-common-methods-api-authentication-explained/&t=3+Common+Methods+of+API+Authentication+Explained)

27 Twitter

29 Google+ (https://plus.google.com/share?url=https://nordicapis.com/3-common-methods-api-authentication-explained/)

19

LinkedIn (https://www.linkedin.com/shareArticle?mini=true&ro=true&trk=EasySocialShareButtons&title=3+Common+Methods+of+API+Authentication+Explained&url=https://nordicapis.com/3-common-methods-api-authentication-explained/)

Reddit (http://reddit.com/submit?url=https://nordicapis.com/3-common-methods-api-authentication-explained/&title=3+Common+Methods+of+API+Authentication+Explained)

HackerNews (https://news.ycombinator.com/submitlink?u=https://nordicapis.com/3-common-methods-api-authentication-explained/&t=3+Common+Methods+of+API+Authentication+Explained)

Total: 88

access control (https://nordicapis.com/tag/access-control/), api (https://nordicapis.com/tag/api/), API key (https://nordicapis.com/tag/api-key/), API keys (https://nordicapis.com/tag/api-keys/), APIs (https://nordicapis.com/tag/apis/), authentication (https://nordicapis.com/tag/authentication/), authorization (https://nordicapis.com/tag/authorization/), Basic Authentication (https://nordicapis.com/tag/basic-authentication/), HTTP Basic Authentication (https://nordicapis.com/tag/http-basic-authentication/), HTTP header (https://nordicapis.com/tag/http-header/), identity (https://nordicapis.com/tag/identity/), identity control (https://nordicapis.com/tag/identity-control/), JWT (https://nordicapis.com/tag/jwt/), multi-factor (https://nordicapis.com/tag/multi-factor/), OAuth (https://nordicapis.com/tag/oauth/), OAuth 2.0 (https://nordicapis.com/tag/oauth-2-0/), password (https://nordicapis.com/tag/password/), resource (https://nordicapis.com/tag/resource/), Security (https://nordicapis.com/tag/security/), single-factor (https://nordicapis.com/tag/single-factor/), SSL (https://nordicapis.com/tag/ssl/), two-factor (https://nordicapis.com/tag/two-factor/), username (https://nordicapis.com/tag/username/)

11 Comments (https://nordicapis.com/3-common-methods-api-authentication-explained/#disqus_thread)

About Kristopher Sandoval

Kristopher is a web developer and author who writes on security and business. He has been writing articles for Nordic APIs since 2015.

(https://nordicapis.com/author/sandovaleffect/)

(https://www.linkedin.com/in/kristophersandoval/)

Tips On Monetizing APIs (https://nordicapis.com/tips-monetizing-apis/)

Tips On Building A Developer... (https://nordicapis.com/tips-building-developer-community/)

11 Comments **Nordic APIs** 1 **Login**

♡ **Recommend** 11 Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Jack tse • a year ago

Quite clear description. I just dived into the API security area, a bit curious why JWT is not mentioned here?

1 ^ | ˅ • Reply • Share ›

inambe → Jack tse • a year ago • edited

Cause JWT is not an authentication framework. JWT can be used with API Keys and Oauth2 frameworks.

2 ^ | ˅ • Reply • Share ›

PL RTZ • a year ago • edited

Thank you, well described. OAuth also has the advantage of cross system authentication capability...i.e. login with facebook uid/password and be authenticated. However, OAuth can be aggravatingly complex to get working initially with visual studio.

1 ⌃ | ⌄ • Reply • Share ›

Daniel Honrade • 2 days ago

I just noticed... should be identify and not identity...

We’ll identity the pros and cons... should be We’ll identify the pros and cons

⌃ | ⌄ • Reply • Share ›

marcc • 2 months ago

OAuth doesn't do user authentication properly speaking. It only provides authentication to the OAuth authorization server itself (in order to allow the OAuth process to take place). OAuth leverages user authentication done somewhere else, which could be a SAML assertion or an OpenID Connect (OIDC) token. OIDC is more commonly used in that case since OIDC is the authentication layer built on top of OAuth. OIDC uses an ID token, OAuth uses an access token.

⌃ | ⌄ • Reply • Share ›

shridhar patil • 6 months ago

understood , thanks

⌃ | ⌄ • Reply • Share ›

Kishore Garnayak • 8 months ago

Simple yet effective elaboration.. Thanks!

⌃ | ⌄ • Reply • Share ›

puspender • 10 months ago • edited

I developed a REST api for www.myownwebsite.com(just an example)
This API is not a public api, only the client application specific to this API can use this API. What would be the best choice? Some people say Basic Auth would work, some says Basic is not that secure.
Some says OAUTH would be the best, but some says OAUTH is for Public APIs who grant access to their resources to some third party clients.
Quite confused.

⌃ | ⌄ • Reply • Share ›

Hari Mothukuri ➜ puspender • 10 months ago

I guess if you are ssl and not worried about real time response basic auth should be enough. I would not use oauth in this case as its just for one client. you can use api key type of auth too based on if its needed to be like session based, and client does not have restriction(authorization) etc., just my opinion

⌃ | ⌄ • Reply • Share ›

Abbas Perçin • a year ago

Top notch article.

⌃ | ⌄ • Reply • Share ›

Soon Santos • a year ago

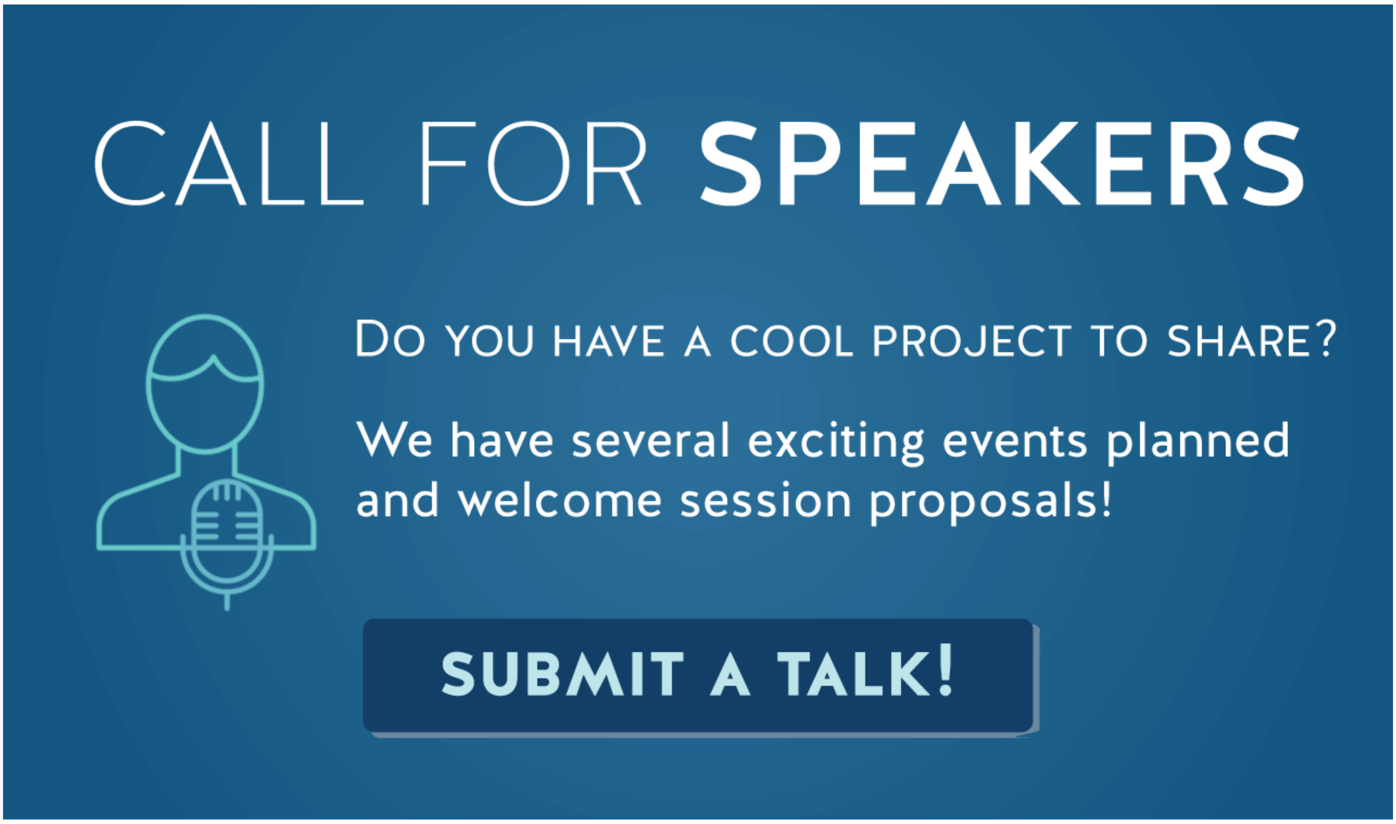
Thank you!

(https://nordicapis.com/events/the-2019-platform-summit/)

(https://nordicapis.com/sessions/gateway-to-graphql-protecting-and-managing-graphql-apis-in-an-open-world/)

(https://nordicapis.com/sessions/oauth-and-openid-connect-in-practice-2)

(https://nordicapis.com/events/livecast-developer-experience-for-api-products/#description)


CALL FOR SPEAKERS
DO YOU HAVE A COOL PROJECT TO SHARE?
We have several exciting events planned and welcome session proposals!
SUBMIT A TALK!

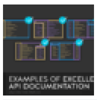
(https://nordicapis.com/call-speakers/)

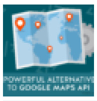
API INSIGHTS STRAIGHT TO YOUR INBOX!

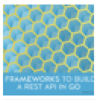
Subscribe to our API Digest

Subscribe

POPULAR POSTS


(https://nordicapis.com/5-examples-of-excellent-api-documentation/) 5 Examples of Excellent API Documentation (and Why We Think So) (https://nordicapis.com/5-examples-of-excellent-api-documentation/)
by Thomas Bush (https://nordicapis.com/author/thomas_bush/) | posted on May 16, 2019


(https://nordicapis.com/5-powerful-alternatives-to-google-maps-api/) 5 Powerful Alternatives to Google Maps API (https://nordicapis.com/5-powerful-alternatives-to-google-maps-api/)
by Thomas Bush (https://nordicapis.com/author/thomas_bush/) | posted on November 1, 2018


(https://nordicapis.com/7-frameworks-to-build-a-rest-api-in-go/) 7 Frameworks To Build A REST API In Go (https://nordicapis.com/7-frameworks-to-build-a-rest-api-in-go/)
by Kristopher Sandoval (https://nordicapis.com/author/sandovaleffect/) | posted on July 4, 2017

(https://nordicapis.com/best-practices-api-error-handling/) Best Practices for API Error Handling (https://nordicapis.com/best-practices-api-error-handling/)
by Kristopher Sandoval (https://nordicapis.com/author/sandovaleffect/) | posted on June 15, 2017

3 Common Methods of API Authentication Explained
by Kristopher Sandoval (https://nordicapis.com/author/sandovaleffect/) | posted on February 6, 2018

Search

Search

RECENT POSTS

Everything You Need to Know About API Pagination (https://nordicapis.com/everything-you-need-to-know-about-api-pagination/)

Simplify Identity Lifecycle Management with SCIM (https://nordicapis.com/simplify-identity-lifecycle-management-with-scim/)

The Need for an API Composition Layer (https://nordicapis.com/the-need-for-an-api-composition-layer/)

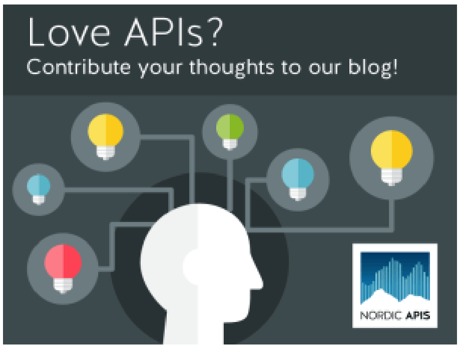
Recipes for API Ninjas (https://nordicapis.com/recipes-for-api-ninjas/)

Embedded Integration Frameworks: Tips from Shopify, Twitch, and More (https://nordicapis.com/embedded-integration-frameworks-tips-from-shopify-twitch-and-more/)

SUBSCRIBE TO OUR FEED

Nordic APIs RSS (http://nordicapis.com/feed/)

CREATE WITH US



(https://docs.google.com/a/twobotechnologies.com/forms/d/12Ng9A_QKUjmAHDgv8Pxb4uLIkECGJawV3vwAWJ4WxTs/viewform)

Follow us

Twitter (https://twitter.com/nordicapis)

Facebook (https://www.facebook.com/NordicAPIs)

SlideShare (http://www.slideshare.net/nordicapis)

LinkedIn (https://www.linkedin.com/company/nordic-apis)

YouTube (https://www.youtube.com/user/nordicapis)

RSS (http://nordicapis.com/feed/)

| Supported by CURITY (https://curity.io) | Website policies (/policies/)