Securing Microservices: The API gateway, authentication and authorization

Latest News (https://sdtimes.com/category/latest-news/) **Published: September 20th, 2017 - Mostafa Siraj (https://sdtimes.com/author/m-siraj/)**

Recently I was building a thousand-piece puzzle with my girlfriend. Experienced puzzle builders have some techniques to finish the task successfully. They follow what we call in algorithms "Divide and Conquer."

For example, in the puzzle I built above with my girlfriend, we started by building the frame, then we gathered the pieces of trees, ground, castle and sky separately. We built each block separately, then at the end we collected all the bigger blocks together to have our full puzzle. Possibly, if we didn't follow this approach and we tried to build line by line, we could have done it, but it would have taken a lot more effort and energy. We also wouldn't have benefitted from being a team because we would have been looking at the same line, instead of doing working collaboratively and efficiently.

This is the same for software! Building software is quite complex, and that's why software architects used to design the solution into separate modules, built by separate teams, then integrated into a full solution. However, a lot of the teams were still struggling with this approach. A single error in a small module could bring the entire solution down. Also, any update to any of the components meant that you would have to build the entire solution again. If a certain module has performance issues, you'd likely need to upgrade all of the servers supporting your application.

For these reasons (and many others), software firms such as Netflix, Google, and Amazon have started to adopt a "Microservices Architecture."

There is no widely accepted definition for Microservices, but there is consensus about the characteristics. Microservices:

- are usually autonomously developed
- are independently deployable
- use messaging to communicate
- each deliver a certain business capability.

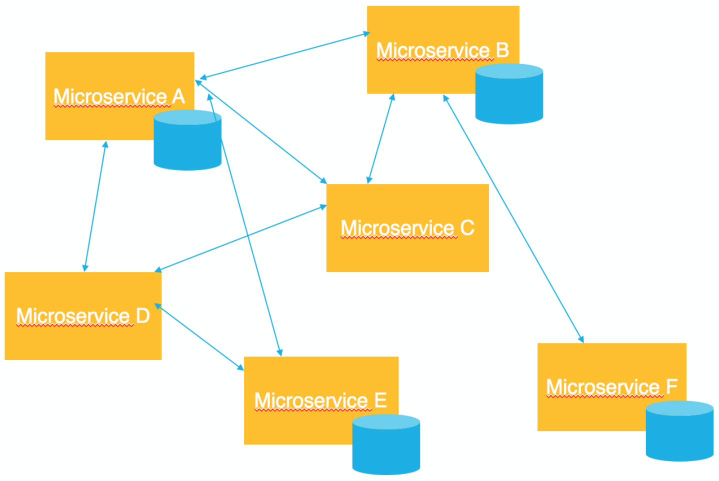
TRENDING STORIES

1 **Unlocking the blockchain potential (http://sdtimes.com/webdev/ur blockchain-potential/)**

2 **For job-seeking developers, skills outweigh resume (http://sdtimes.com/devexec/jo seeking-developers-skills-outweigh-resume/)**

3 **Securing Microservices: The API gateway, authentication and authorization (http://sdtimes.com/micro/secu microservices-the-api-gateway-authentication-and-authorization/)**

4 **Developers want more from platform providers, report says (http://sdtimes.com/webdev/de want-more-from-platform-providers/)**



(https://sdtimes.com/wp-content/uploads/2017/09/image1.png)

The graph above illustrates the structure of a typical microservice. You'll notice that some microservices will have their own data stores, while others only process information. All of the microservices communicate through messaging.

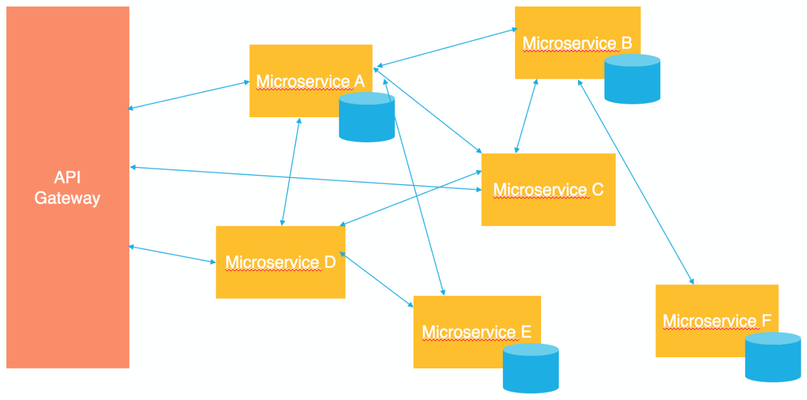
While a microservices architecture makes building software easier, managing the security of microservices has become a challenge. Managing the security of the traditional monolithic applications is quite different than managing the security of a microservices application. For example, in the monolithic application, it is easy to implement a centralized security module that manages authentication, authorization, and other security operations; with the distributed nature of microservices, having a centralized security module could impact efficiency and defeat the main purpose of the architecture. The same issues hold true for data sharing: monolithic applications share data between the different modules of the app "in-memory" or in a "centralized database," which is a challenge with the distributed microservices.

The API Gateway

Probably the most obvious approach to communicating with microservices from the external world is having an API Gateway.

Postman

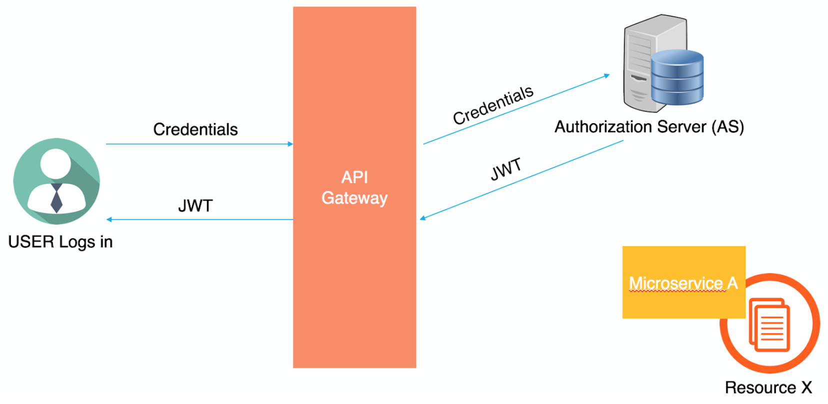
(https://cta-redirect.hubspot.com/cta/redirect/3475429/bedccda7-f86f-4600-aaf8-b73b4e93ab90)



(https://sdtimes.com/wp-content/uploads/2017/09/image3.png)

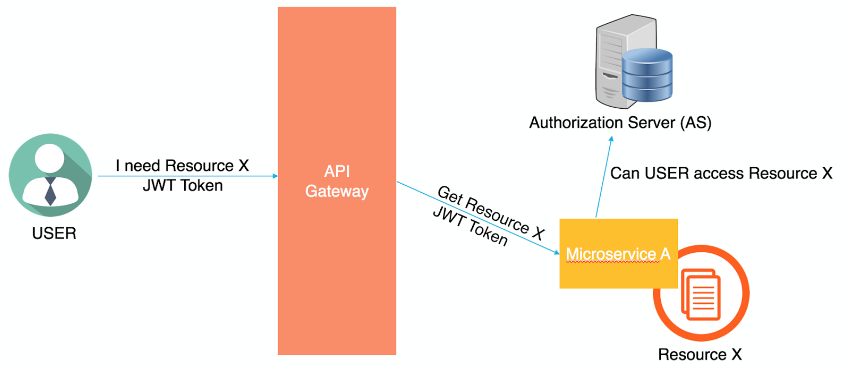
The API Gateway is the entry point to all the services that your application is providing. It's responsible for service discovery (from the client side), routing the requests coming from external callers to the right microservices, and fanning out to different microservices if different capabilities were requested by an external caller (imagine a web page with dashboards delivered by different microservices). If you take a deeper look at the API Gateways, you'll find them to be a manifestation of the famous ***façade design pattern*** (https://en.wikipedia.org/wiki/Facade_pattern).

From the security point of view, API Gateways usually handle the authentication and authorization from the external callers to the microservice level. While there is no one right approach to do this, I found using OAuth delegated authorization along with JSON Web Tokens (JWT) to be the most efficient and scalable solution for authentication and authorization for microservices.



(https://sdtimes.com/wp-content/uploads/2017/09/image2.png)

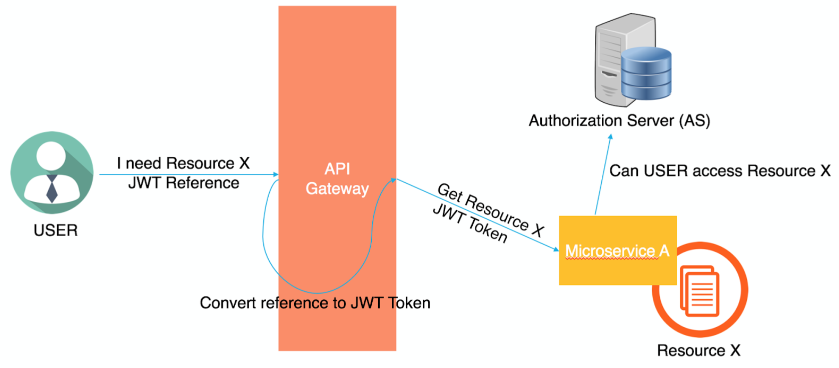
To illustrate further, a user starts by sending his credentials to the API gateway which will forward the credentials to the Authorization Server (AS) or the OAuth Server. The AS will generate a JASON Web Token (JWT) and will return it back to the user.



(https://sdtimes.com/wp-content/uploads/2017/09/image4.png)

Whenever the user wants to access a certain resource, he'll request it from the API Gateway and will send the JWT along with his request. The API Gateway will forward the request with the JWT to the microservice that owns this resource. The microservice will then decide to either grant the user the resource (if the user has the required permissions) or not. Based on the implementation, the microservice can make this decision by itself (if it knows the permissions of this user over this resource) or simply forward the request to one of the Authorization Servers within the environment to determine the user's permissions.

The approach above is much more scalable than the traditional centralized session management. It allows every individual microservice to handle the security of its own resources. If a centralized decision is needed, the OAuth server is there to share permissions with the different microservices.



(https://sdtimes.com/wp-content/uploads/2017/09/image5.png)

A challenge with this approach will be if you want to revoke the permissions of the user before the expiration time of the JWT. The microservices are distributed, possibly in several locations around the world, and the API Gateway is just forwarding the requests to the responsible microservice. That means that revoking the permissions requires communicating with every single microservice, which is not very practical. If this was a critical feature, then the API Gateway can play a pivotal role by sending a reference of the JWT to the user instead of the JWT value itself. Each time the user requests a resource from the API Gateway, the API Gateway will convert the reference to the JWT value and forward it as normal. If revoking the permissions is needed, then only a single request to the API Gateway will be provided, then the API Gateway can kill the session for that user. This solution is less "distributed" than the JWT value so it's up to the Software Architect and the application requirements to follow either approach.

ARTICLE TAGS

APIs (https://sdtimes.com/tag/apis/), micro (https://sdtimes.com/tag/micro/), microservices (https://sdtimes.com/tag/microservices/)

Subscribe to SDTimes (http://resources.sdtimes.com/sdtimes-subscription?__hstc=54983092.b1c3fc9f788d79895403cd04467299f2.1402689135055.1402689135055.1404158676313.2&__hssc=5498

Facebook Twitter Email More

About Mostafa Siraj

View all posts by Mostafa Siraj (https://sdtimes.com/author/m-siraj/)

RELATED ARTICLES

Analyst Watch: From monoliths to microservices (https://sdtimes.com/micro/ar watch-monoliths-microservices/)

Instana Joins OpenTracing Specification Council (https://sdtimes.com/apm/inst joins-opentracing-specification-council/)

The Twelve-Factor app methodology sets guidelines for modern apps (https://sdtimes.com/webdev/t factor-app-methodology-sets-guidelines-modern-apps/)

The difference between a miniservice and a microservice (https://sdtimes.com/micro/di miniservice-microservice/)

SD Times

- About Us (https://d2emerge.com/)
- Contact Us (https://d2emerge.com/about/)
- Privacy Policy (https://sdtimes.com/privacy-policy/)
- Site Map (https://sdtimes.com/site-map/)
- Advertise (http://assets.sdtimes.com/advertise-with-sd-times)
- Subscribe (http://assets.sdtimes.com/sdtimes-subscription)

Subscribe to SD Times

(http://assets.sdtimes.com/download-latest-issue)

Ready for your SD Times magazine? It's just a click away!

Subscribe (http://assets.sdtimes.com/sdtimes-subscription)

- Read Current Issue (http://assets.sdtimes.com/download-latest-issue)
- Read Back Issues (https://sdtimes.com/back-issues/)

Tweets by @sdtimes

SD Times @sdtimes

Developers should focus on finding bugs throughout the software development life cycle, not just during QA testing buff.ly/2EdR2rM

Guest View: Use this approach to cat...
A testing approach to finding the maxim...
sdtimes.com

20m

Embed View on Twitter