To: Professor Krasso

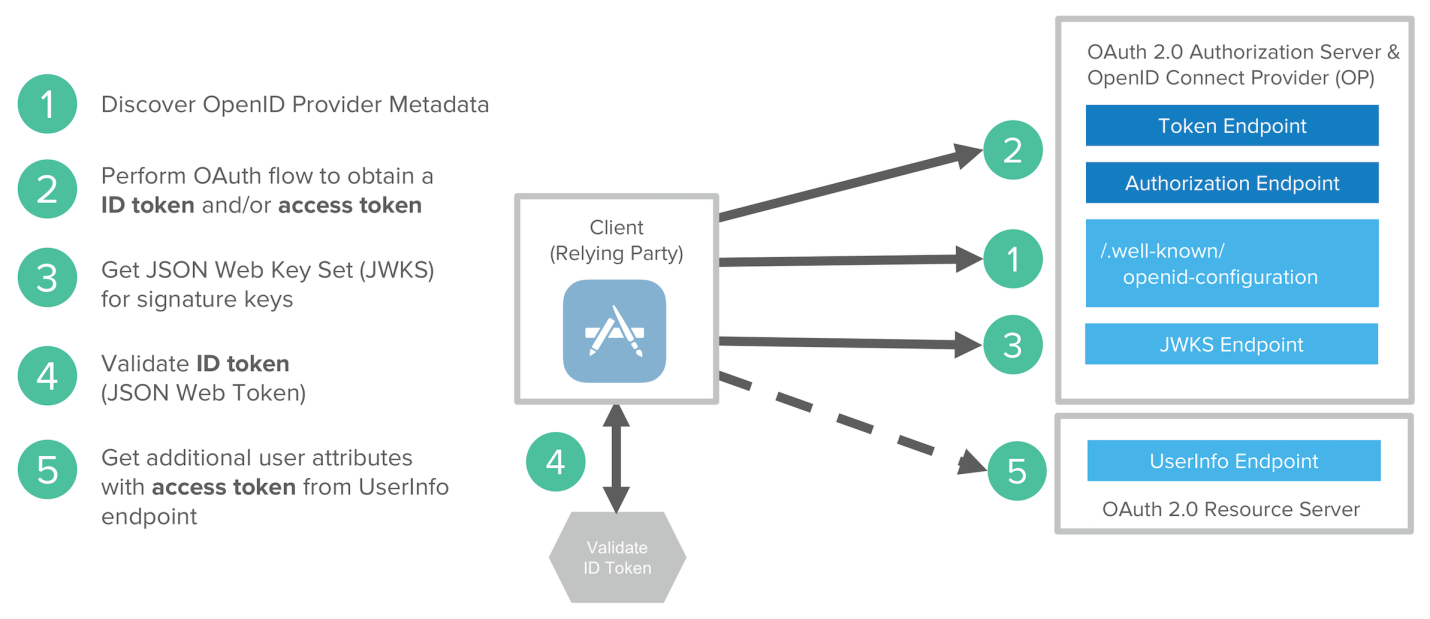
From: Nicole Forke

Date: 21 June 2020

Subject: Security in Microservice Architecture

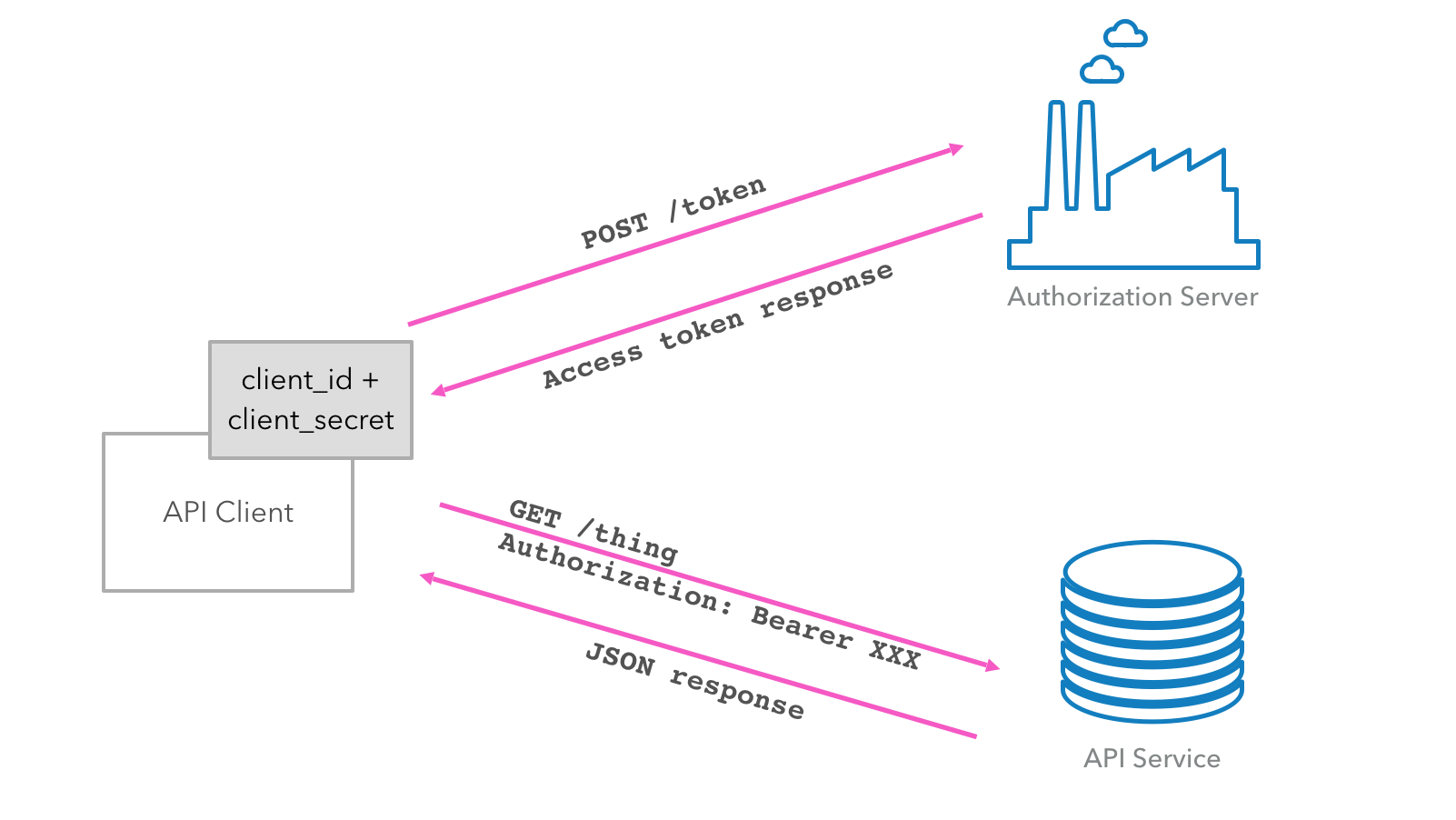
Security in a microservice architecture is not an easy task. A lot of companies still use a monolithic approach for their applications because the security has been around longer and well tested. Before moving your applications over to a microservice architecture you need to make sure a microservice will benefit you in the long run. Ask yourself, will a microservice help us achieve agility and scale? Then weigh the benefits before making any final decisions. If you decide microservices is right for your application and deploy them, then it's time to think about security.

When securing a microservice you'll want to use OAuth for user identity and access control. Your application is going to need to perform access control and authorization handling. The majority of them do. If you do your research, you'll see that OAuth is the standard for application authorization so there's no need to create your own protocol. With OAuth you can rely on libraries and platforms that will accelerate your development while improving security. Communicating between microservices uses OAuth 2.0's client credentials to implement secure server-to-server communication. You can reference the chart below, it's an example of how the identity of a user can be looked up by sending an access token to the /userinfo endpoint.



<https://developer.okta.com/blog/2020/03/23/microservice-security-patterns>

When developing for microservices you need to think about security vulnerabilities from the design up. You need to be able to think about every potential weakness that might occur such as cross site scripting (XSS). So when creating a username class it needs to encapsulate all of the security concerns. You should also use HTTPS everywhere, even if your microservice is static. HTTPS is designed to ensure privacy and data integrity between computer applications. This is very important when using microservices because to use HTTPS you need a certificate. Which sorts and serves two functions. It grants permission to use encrypted communication and authenticates the identity of the certificate holder. Here is an example of server to server communication where the API client is one server and the API server is the other.



<https://developer.okta.com/blog/2020/03/23/microservice-security-patterns>

Another very important part of securing microservices is to encrypt and protect secrets. Your microservices will likely have secrets they use when communicating with other servers or services. In order to protect these secrets you want to make sure you don't check them into source control. Instead you need to store them in environment variables and make sure you encrypt them. There are many tools out there that you can use to encrypt these secrets, you just need to do some research and find the one that's right for your application.

Overall, securing a microservice architecture application over a monolithic application takes a lot of time and a great team. When using microservices make sure you are secure by design, use HTTPS everywhere while encrypting secrets that are being passed from server to server/service. Remember not to reinvent the wheel, there are plenty of services and techniques already out there that you may utilize in order to secure your applications. If you follow these guidelines when developing your microservices you'll have a security.

References

Deciding if Microservices are Right for You. (2017).

<https://devblog.axway.com/apis/deciding-if-microservices-are-right-for-you/>

Okta. (2020). *Security Patterns for Microservice Architectures.*

<https://developer.okta.com/blog/2020/03/23/microservice-security-patterns>