To: Professor Krasso

From: Chris Bohnet

Date: 06/17/2020

Subject: Securing Microservices

With all the advantages that the outbreak of microservices has to offer, it could come at a cost for security if not carefully considered.

There are various reasons why security can become an issue:

1. Communication and data exchanges are exposed over a network relative to past

legacy systems of in-memory on local servers.

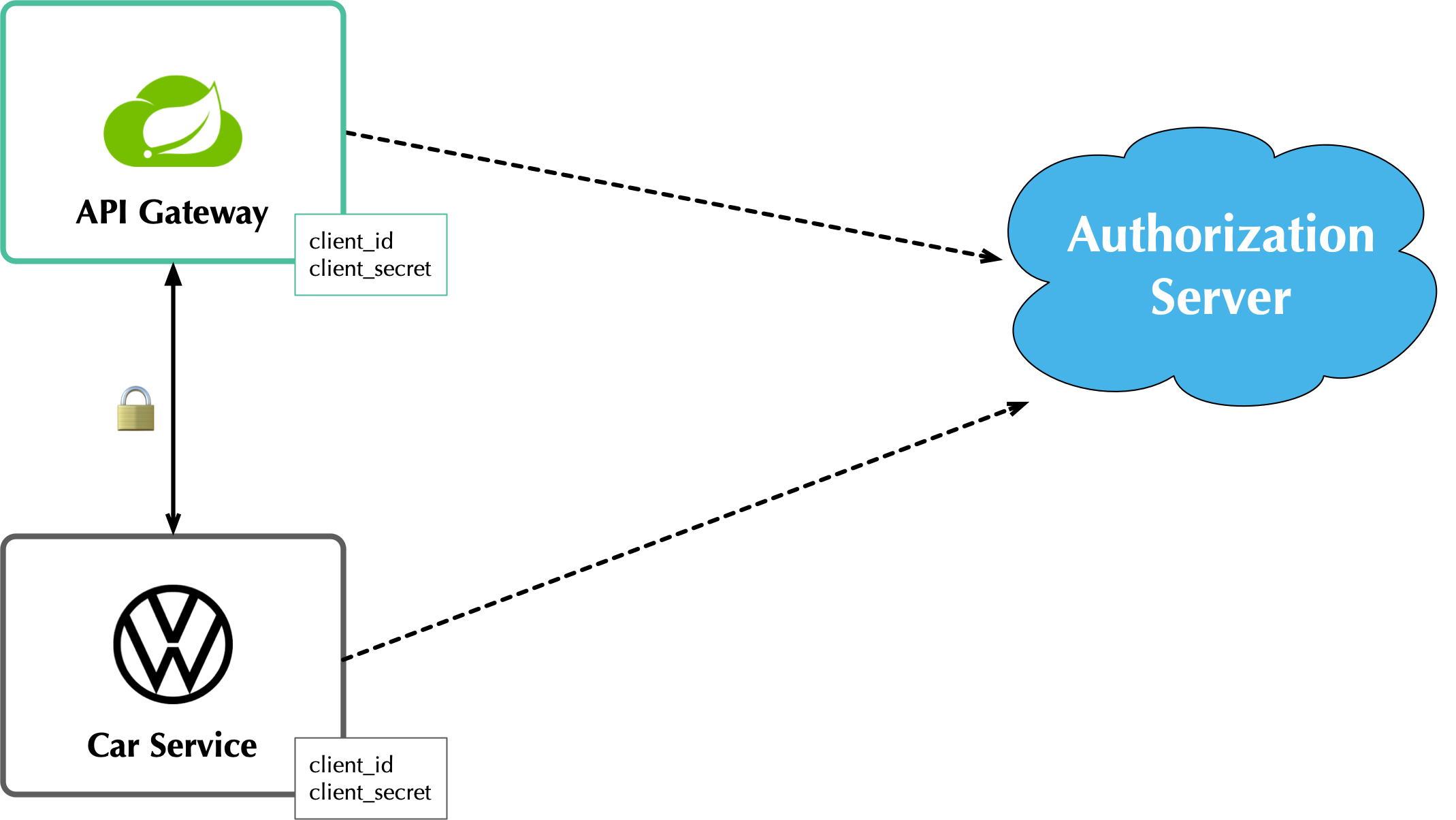
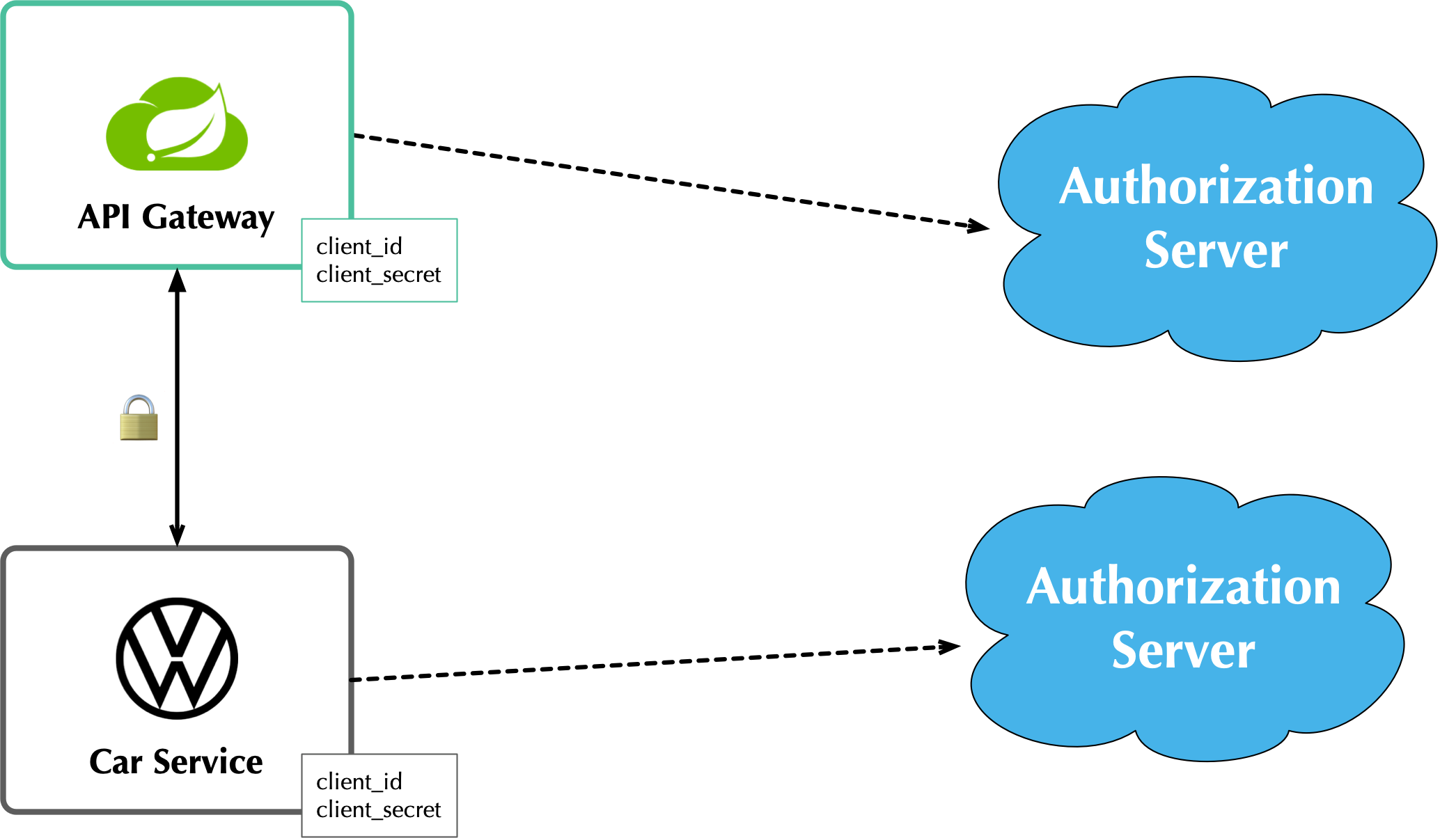
2. Tightly coupled contracts between services and processes of numerous databases and threatened endpoints.

3. Microservices add a layer of complexity that requires technical experience that are cautiously mindful of security gaps.

Not to fear. There are precautions and guidelines that can be adhered to in order to protect against any of these security gaps.

First, follow general good coding practices. Establish coding standards that can be reused across microservices that are repeatedly tested. This practice decreases possible security vulnerabilities which can be introduced with new designs. Also, less is better when determining the microservice access. Identify the risk of each microservice if compromised and determine precautions prior to deployment. Conducting peer reviews and keeping documentation current is imperative to safeguard your microservices, and is generally a good development practice as a whole as well.

Secondly, OAuth/OAth2 are generally thought of as the industry standard for authentication. They come with libraries and platforms that can be used to greatly increase development productivity. OAuth2 uses an authorization server and can support both many-to-one and one-to-one authorization. It's recommended to use the many-to-one since the one-to-one can become very busy and requires a good design to be effective.

*Figure 1. Many-to-one relationship provided by Figure 2. One-to-one relationship provided by*

[*https://developer.okta.com/blog/2020/03/23/*](https://developer.okta.com/blog/2020/03/23/microservice-security-patterns)*.* [*https://developer.okta.com/blog/2020/03/23/*](https://developer.okta.com/blog/2020/03/23/microservice-security-patterns)*.*

Additionally, utilize the defense in depth theory by Identifying your most sensitive microservices and create multiple layers of protection for them. You should utilize the encryption/decryption adapters and libraries that are out there already. Don't attempt to create your own. The complexity involved and the security risk associated if not properly built is not worth the risk and why reinvent the wheel?

Also, choose to use HTTPS over HTTP. Everywhere. HTTPS is built to provide privacy and data integrity between services. HTTPS uses certificates to grant permissions to use encrypted communication by a Public Key Infrastructure (PKI), and also authenticates the identity of the certificate’s holder. Also using an API gateway for a single request entry point provides an interface to handle the external requests and talk to your microservices behind a secure firewall.

Containers are another good practice to use for automated security and software updates. This involves creating a wrapper around your service for the updates so testing is just a matter of wrapping a different container around it for each new update. It would also be useful to perform automated periodic security scanning of the containers.

To conclude, use microservices. They are a great way to architect a complex system into smaller narrow-defined services for quick development and scalability. Just be sure to keep security precautions at the fore-front to your design. Thinking ahead and following industry recommendations will only benefit your solution for the long term.

References

TechRepublic, "10 Tips for securing microservice architecture", as referenced at <https://www.techrepublic.com/article/10-tips-for-securing-microservice-architecture/>

TechBeacon, "8 Best practices for microservices", as referenced at <https://techbeacon.com/app-dev-testing/8-best-practices-microservices-app-sec>.

Okta, “Security Patterns for Microservices”, as referenced at <https://developer.okta.com/blog/2020/03/23/microservice-security-patterns>