Aalto University
School of Science
Bachelor's Programme in Science and Technology

Security in Microservice Architecture

- Impact of a Switch from Monolith to Microservices

Bachelor's Thesis

xx. xxxxxkuuta 2020

Tommi Jäske

Aalto-yliopisto
Perustieteiden korkeakoulu
Teknistieteellinen kandidaattiohjelma

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä:	Tommi Jäske
Työn nimi:	Turvallisuus mikropalveluarkkitehtuurissa - Monoliitisesta arkkitehtuurista siirtyminen mikropalveluarkkitehtuuriin ja sen vaikutukset.
Päiväys:	xx. xxxxxkuuta 2020
Sivumäärä:	?
Pääaine:	Computer Science
Koodi:	SCI3027
Vastuuopettaja:	Professori Eero Hyvönen
Työn ohjaaja(t):	Professori Tuomas Aura (Tietotekniikan laitos)
Kirjoitetaan myöhemmin.	
Avainsanat:	avain, sanoja, niitäkin, tähän, vielä, useampi, vaikkei, niitä, niin, montaa, oikeasti, tarvitse
Kieli:	Suomi

Aalto University
School of Science
Bachelor's Programme in Science and Technology

ABSTRACT OF
BACHELOR'S THESIS

Author:	Tommi Jäske
Title of thesis:	Security in Microservice Architecture - Impact of a Switch from Monolith to Microservices
Date:	MonthName 31, 2020
Pages:	?
Major:	Computer Science
Code:	SCI3027
Supervisor:	Professor Eero Hyvönen
Instructor:	Professor Tuomas Aura (Department of Computer Science)
Will be written.	
Keywords:	key, words, the same as in FIN/SWE
Language:	English

Contents

1 Ideas

In MA it is possible to implement features in such ways that a session can carry user information. This information can consist of granted roles and rights for the user. This session can be queried when e.g. access control is needed to execute an action or operation.

Sessions can be used in MSA when the architecture has an API Gateway. The session is stored in the Gateway and a session key (?) is carried in the requests made by the client application. The API Gateway then verifies the request and can grant access to specific service. Without the Gateway the session would have to be either centrally maintained at a session service or be handeld by the client and passed along in the requests.

Client handled session is easy to tamper with and has risks involved. The service would either have to trust the client offered session or verify a signature. In both of these cases the client application would have to be aware of the service and would be able to communicate with it directly. This would bring considerable overhead on all the services and a security risk.

2 Comments

authentication, credential, authorization, access control, policy decision point (PDP), policy enforcement point (PEP), session, token (JWT, OAuth 2.0). (API gateway, IdP) vs distribution

Identity and Access Management (IAM), Identity federation, Azure Active Directory, Security Assertion Markup Language (SAML), SAML identity provider, OpenId provider, OAuth, OpenId Connect (OIDC),

Identity Management: Identification, Authentication, Authorization. ...

3 Possible new structure

3.1 Identity management

4 Introduction

In recent years the mobile app has revolutionized our daily lives. These services have infiltrated social life, shopping and almost every aspect of our existence. The services and their apps compete of our time and markets are reinventing themselves constantly. The rapid expansion and at times even faster decline of these web services need a matching

architecture to meet these very specific needs.

There are many web services already in use which have been designed and implemented before the onslaugh of microservices. Some of these services need to evolve to be of use in the future. In many cases the monolith services have already started to use certain aspects from the microservice world, such as access tokens and REST API:s. The pressure from new competitors adopting new technologies right from the start and the fact that the industry and its developer base are extremely young dictates that the old and established services have to address the situation someway or the other. Monoliths have served us well but the time has come to evolve with the customer needs.

Stackoverflow annual survey (**?**) conducted on developers finds that half of the respondents identified as full-stack or backend developers. The professional developers had very little experience and about 40% of them had less than five years of professional experience.

The new developers entering the work force have very different mindset than the older more seasoned professionals. Thus, it is very clear that the ways of working and paradigms to be used are in constant change. The old and established have to embrace the change and refactor their architecture before it is too late. Microservices are not the proper choice for all needs (**?**) but in many cases there simply is no other valid choice. This change needs to happen in an orderly and safe way and the security aspects need to be addressed.

Microservice Architecture (MSA) differs in many ways from the more tradition Monolith Architecture (MA). This shift entails very specific security issues.

In this thesis the MSA and security literature is evaluated and the main differences between MA and MSA on back end security aspects are found.

Chapter a. presents the definitions used in this thesis. Chapter b. discusses the Confidentiality aspect of a switch from MA to MSA. In chapter c. Integrity of the information is discussed in the context of MA and MSA. Chapter 5. presents Availability when changing from MA to MSA. In Chapter 6. other relevat security aspects are presented. Chapter 7. contains the conclusions and presents further research topics.

5 Definitions

This thesis uses the following definitions.

5.1 Microservice

MSA can be viewed as an extension of the service oriented architecture (SOA)(**??**). It's guiding principles are stated in the SOA manifesto (**?**) and one is to prioritize:

- *Business value over technical strategy*
- *Strategic goals over project-specific benefits*
- *Intrinsic interoperability over custom integration*
- *Shared services over specific-purpose implementations*
- *Flexibility over optimization*
- *Evolutionary refinement over pursuit of initial perfection*

A microservice is a service that: is independently deployable, is modeled around business domain, that owns the data that they need to operate, that communicates via network, is technology agnostic, that encapsulates data storage and retrieval and that has stable interface (**?**).

5.2 Security

Security can be defined in multiple ways but in this thesis security and more specifically information security is defined as consisting of Confidentiality, Integrity, and Availability (CIA) as is stated in the pocket book on ISO/IEC 27001 -standard for information security (**?**).

The ISO/IEC 27001 standard defines confidentiality as such that information or property is available to the authorized user only. The authorized users can consist of persons, processes or entities to whom the information or property can be disclosed. Integrity means that the data or property is safeguarded for accuracy and completeness. Availability in this web service context is defined as such that the property or information is available when it is needed.

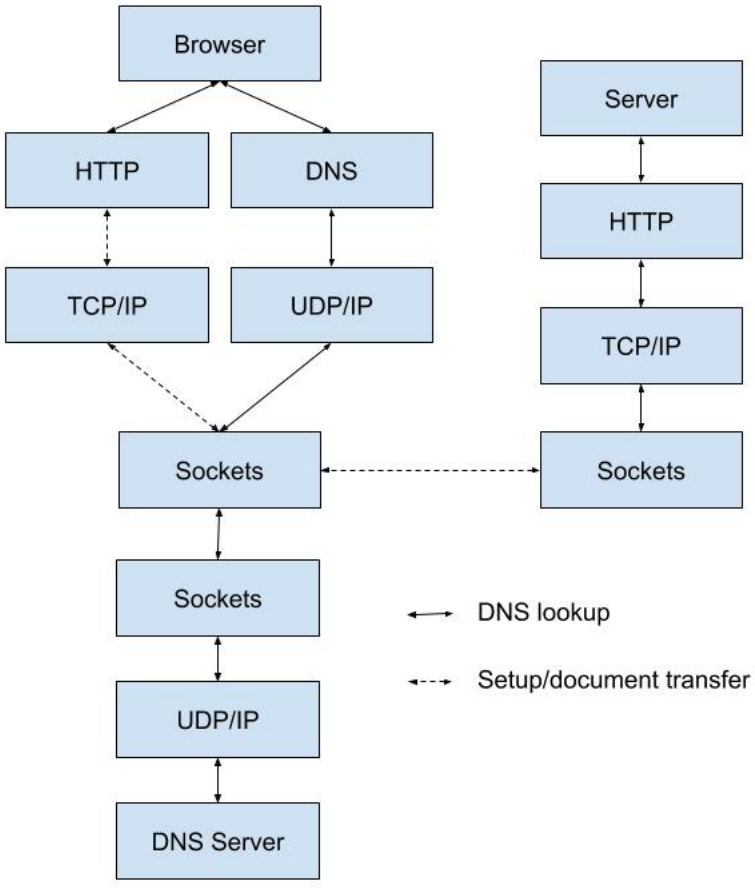
6 The Switch of Architectures

To change the architecture from MA to MSA should be a gradual process. The MA is or at least should be split to modules with separation of concerns (**?**). The actual splitting of the monolith can be carried out in various ways. One of whic is DDD (SOURCE FOR THIS).

The MSA differs from a MA in fundamental ways. According to **?** one of which is the communication between its components. In a monolith application the processes can send function calls or method invocations amongst them selves. In MSA the messaging is based on sending messages or HTTP requests.

Function calls entail a stackframe creation in the call stack, execution of the function code and finally popping the stackframe and returning the result. The actual overhead depends on the language and systems used to run the application (SOURCE). Compilers can optimize the code further and inline the function calls to eliminate the stackframe creation and following procedures to be carried out.

Communication using the network is extremely slow. In a paper **?** studied the response times of web sites offered to the public. The websites response times where measured in seconds. In the paper a simplified model presented of the layered model of the communication (Figure 1).



The requests sent to other microservices throught the network are extremely slow when compared to operation within one computer as the function calls would be. Therefore, the communication patterns should be changed to take into account the change in communication path.

If the architecture is changed in such a way that the previous communication model amongs the components is preserved, there would be an excessive amount of communication and the resulting system is not as performant as it could be (**?**).

7 Confidentiality

Confidentiality in a web service is usually critical security feature. There are specific services which do not nead information confidentiality regarding some of the data such as public weather services and other similar data. Some of the information is still regarded sensitive and must be kept confidential. These data can consists of personal, user, logs or other similar content. Users should not be able to use or view content not authorized to him/her. To verify this a form of access control has to be performed. Access control consists of user authentication and authorization. A user has to authenticate him/her self and authorization is acquired to access to information or property.

In an MA access control can be implemented using sessions. A user authenticates using appropriate channels and a session with a session key is created. The session can have an expiration time and the messages originating from the user interface (UI) carry this key. Sessions and session keys can be used in a distributed system which MSA is but the implementation is more difficult (**?**).

7.1 Authentication

In these cases where the user has to be authenticated the web service needs a way to do this securily. Usually authentication is done using a tuple containing user credentials i.e. a username and a password for the user. The user is authenticated and a key or token is transmitted to the user via the network. This communication should in both MA and MSA be encrypted in a way that none of the actors in the transfer path can intercept the message and be able to use the credentials.

The credential counterparts i.e. shared secret by the server and the user have to be available for the web service for verification. When using MSA the service should own it's own data. When ever such information is available it is a target for thieves and hackers. The services in MSA are to be individually deployable and the service scalable. Authentication service implementation has to take this into account. The service has to adhere to practices that minimize the risks of data breaches.

7.1.1 Attacks

Authentication can be attacked by a multitude of methods.

- Cracking
- Impersonation attacks
- Hacking the system
- Malware
- Social engineering
- Cracking the encryption on the communication channel exchanging credentials and keys or tokens.

From 2013 onwards malware and data breaches performed by hackers have increased and the scale of the damage is massive. The user data containing also the user passwords or hash thereof is valuable commodity which can be traded in the black markets. The damage of the dataloss can be susbtantial. The estimated value from the Yahoo data breach is over $440 billion. The attacks seem to have been targeted to entities with valuable data and also to such targets that are lacking secure infrastructure. The least likely target to be hacked where non profit organisations and the most likely were medical related organizations (**?**).

The hacked account credentials have to some extent been available for download from the web. **?** created a service where everyone can verify whether any of their accounts are amogst the ones added to the service. The service named as "';– have i been pwned ?" allows users to enter their username or password to the site and see a result.

7.2 Authorization

Authorization of the user rights can be implemented in various ways. One of which is an authorization service which can contain the access control matrix. Services being accessed verify from the authorization service that a particular user or the role that the user has can access the requested service or functionality.

In a MA the access rights to a functionality can be implemented using annotations within the source code. This can be effective since the verification can be done in memory or atleast without network communication. If a session is used it can contain the information needed to verify access rights.

In contrast to the MA in the MSA the access control matrix or matrices can't be as easily accessed. In order to verify that a specific right exists the service would have communicate with the authorization service every time a user tries to access a functionality with access restrictions. This could potentially lead to an extremely lively communication from all the services a formation of a bottleneck to the service.

https://techbeacon.com/security/microservices-apps-do-identity-access-management-without-overhead

7.3 Interaction Paradigms

As already discussed in an MA the service components can communicate using events, procedure calls or other methods available within a single server machine. Usually all this communication stays within a single computer and thus does not necessarily compromise confidentiality.

In MSA single services communicate via a network. TODO

Next messaging systems list is from (**?**): lightweight - REST API, Sync RPC, GraphQl - Async REST, gRPC - Apache Kafka, ZeroMQ - Java Message Service: 1 ActiveMQ, 2 JBOSS messaging, 3 Glassfish - AMQP: 1 RabbitMQ, 2 Qpid, 3 HornetQ - MuleESB, Apache ServiceMix, JBossESB - heavyweight WebSocket

7.3.1 Representational State Transfer (REST)

? presented REST in 2000 and it has become very successfull. The architectural style is was derived using various constraints one of which is the demand of stateless communication. The communication i.e. the request must contain all information for the server to fullfil the request. All session state is stored in the client of which the server has no prior knowledge before a request.

In her doctoral thesis **?** critiques the REST paradigm from the security perspective. She states that the design of the architecture does not meet the security requirements for web applications. The statelessness of REST does not allow for any server side sessions and thus making e.g. token repudiation impossible due to not being able to verify tokens other than the correct issuer by signature and the validity. As such tokens are more compatible with REST but there still has to be the private keys in the server for signature verification.

7.3.2 Event-Driven Communication

7.3.3 Effects on Confidentiality

7.3.4 Example case

8 Integrity

Information integrity in an MA web service is usually left to a single database and sound architectural choices (REALLY? SOURCE). Transactions can be used when updating database constents to make sure that atomicity, consistency, isolation, and durability (ACID) (**?**) is followed. When using MSA according to the definition each of the micro services should contain or have access to it's own data i.e. database. This leads to extreme difficulties in information integrity. TODO

8.1 Introduction

8.2 Effects on Integrity

8.3 Example case

8.4 Threats

9 Availability

Availability in this web service context is defined as such that the property or information is available when it is needed.

9.1 Possible Attacks

D-o-S

9.2 Comparison

9.3 Introduction

9.4 Effects on Availability

9.5 Example case

10 Other MSA specific security matters

10.1 Platforms

Docker Swarm Kubernetes (K8s) Azure

sandbox virtualization

10.2 Monitoring and logging

10.3 Software Development

10.4 Deployment and Operation

Developing software using the MA the structure the whole application or service is usually deployed as a whole and the program code can be compiled, tested and used as a single unit or multiple modules. In contrast to this a service implemented by using a MSA can be deployed in single microservice units and thus a single service can be worked upon individually and deployed once ready.

The immediacy in the deployment of the microservices entail a very specific security risk. In a paper **?** present threats from malicious insiders working on the services as developers or other positions with access to sensitive information. In microservice development the finished implementations are to be immediately released to production. There are few steps in the CD pipeline prior to this but once tests pass in the test environments the pipeline is supposed to publish the changes to the actual production environment. The paper presents four specific threats. The first one is that the knowledge of sensitive information is spread among the developers more widely than in MA. The developers need access to be able to produce working solutions. The second threat is that the insiders monitoring and operating the running system intentionally harm the system by making malicious changes. The third threat is the developers knowing the configurations and their ability to make almost instat changes to them or the microservices them selves.

The last presented threat in the paper is the non-repudiation. The system is not able to dis-allow malicious requests when the developers have had access to the keys and other configurations. They can effectively implement services or requests that emit malicious requests or responds.

Malicious attempts in a MA are more easily screened by performing security audits and by peer reviewing the code. In a MSA the knowledge of a single service and it's inner workings are shared by a more limited number of people. Finding the compromised actions from the interoperability of the distinct microservices is a daunting task.

10.5 Service discovery

The MSA can have a service discovery service into which all available services can register them selves. https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/ https://www.consul.io

10.6 Externalized configuration

To allow for easy configuration change management there should exist a configuration orchestration service. This service should have an API from which services in their startup can load their appropriate configuration. The configuration of the whole system can be easily maintained through the API.

The contents of the configuration is highly sensitive information. It consists of addressess, credentials and other information that alter the behaviour of the system. Therefore, the content must be stored safely and not allowed to be read or altered by unauthorized users.

11 Conclusion