Security Issues in Microservices

INFS605

Name: Ieuan Davies

Student ID: 17981714

Introduction

The adoption of microservices architecture has brought about remarkable improvements in scalability and agility, yet it has also exposed a complex web of security challenges that demand our attention. In this essay, we will explore a portion of the world of microservice security. Our exploration will begin with pointing out why there are security issues when using microservices. Then we will move on to the task of identifying the security issues that trouble individuals who develop microservices. Later, we will look into an in-depth analysis of these issues, pointing out their potential consequences. Eventually, we will provide educated justifications for the proposed solutions, offering insight into the potential security-enhancing benefits that microservice developers gain from adopting the suggested techniques. Throughout the essay, we aim to encourage the importance of addressing these security concerns to ensure the continued success and sustainability of microservices in our ever-evolving digital landscape.

Problem Statement

Why are there security issues in microservices?

In comparison to microservices, a monolithic application's communication that happens between internal components occurs within a single process- a Java application [1]. However, with microservices, the same internal components would be designed as independent, separate microservices. Rather than having a couple of observable entry points of attack by using a monolith, the usage of microservices results in many more entry points. As more points of entry start to exist, the surface area of attack expands [1]. Therefore, anyone planning on using microservices must face the challenge of deploying ways to protect each entry point from attack. The security of a system is no stronger than the strength of the weakest link [1].

What are the issues that will be covered?

Security issues in microservices are a very broad topic to cover, as there are many factors to consider when implementing security features. Therefore, for this essay three major security issues will be discussed:

- How is a system able to identify who is accessing the system to make sure individuals who should not have access are kept out?
- How to stop people from being able to access information and functions within the system that they should not have access to?

- How to protect microservices from vulnerability exploits that allow unintended access to private information.

The problems these issues cause to the overall safety of a microservices architecture will be discussed and solutions to these issues will be suggested to greatly mitigate the ability of attackers to breach the system.

Analysis and Justification

How is a system able to identify who is accessing the system to make sure individuals who should not have access are kept out?

A key aspect of security is protecting the data of a system's users. Protection becomes an issue when someone who is supposed to have access manages to find a way in, as they would possibly gain access to private data [2]. Therefore, it is important to add layers of protection to microservices to make sure only the people who are supposed to have access gain access.

Authentication

A core concept when allowing users to interact with a system is Authentication, this is the process that allows the system to confirm whether or not a person accessing the system is who they say they are. A very common way to authenticate a human user is to implement a username and password requirement to access the system. The idea behind this authentication is that the individual entering this information is the only one with access to this information [3]. This is a simple single layer of protection that would stop anyone without a set of these credentials from accessing the system.

Multi-Factor Authentication (MFA)

Unfortunately, in modern times data hacks can be quite common and can result in attackers gaining access to one set of online credentials or thousands. Data stolen could result in users of a system being used to send fraudulent emails and experience money being taken out of their bank account. In reality, a username and password are static and once someone has access to both they have access to the account until either is changed. This is where MFA is a useful solution to this problem, this is the concept of needing extra layers of verification on top of just your username and password. Simply adding a non-static variable that only the user could have access to increases the security massively. Companies such as Google recommend that users secure their online accounts using MFA [4]. Google offers its users various forms of MFA. Alongside a basic password, users can be sent a one-time security code to their mobile phone, or a code that is constantly regenerated using the Google Authentication app [4].

Justification

Successfully implementing some form of authentication greatly improves the security of any microservice architecture as it forms a base layer of protection against individuals who should not have access to the system. For example, using password protection to keep email accounts secure. Rather than having each individual have access to everyone else's emails including their own, password authentication creates a boundary between every email account. Then again there will be individuals that steal these credentials so it is highly recommended that MFA is introduced into the system to guarantee a high-level authentication security, as it is very difficult to steal credentials that can only be generated by a user's device or an authentication code that regenerates its format every few seconds.

How to stop people from being able to access information and functions within the system that they should not have access to?

Once a user is authenticated and given access to a given microservice, they should not be able to access everything. This is dangerous as a user who has access to everything ends up having access to all the private data that is on the system.

Authorisation

Authorisation is another core concept when letting users interact with a system, this concept runs close with authentication. Once a user is identified as the right person, information provided to the system will decide what they are allowed to do [3]. For example, if the system was told that the user is a customer, that user would have a set of functions they can access but they would not have access to the higher level management tools of the system. This makes sure that people only have access to what they need and nothing more.

Single Sign On (SSO)

A useful tool for implementing secure authorisation across all microservices is SSO. SSO is a tool that allows users to log in to multiple applications and websites only having to authorise themselves once [n]. Amazon uses SSO across their services by generating an SSO token and checking whether the user was already authenticated into the system, if the check is successful the user gains access to the system. Otherwise, the user will be directed to a service to authenticate their credentials. If the user provides valid credentials they gain access to the system, if they do not they will be required to re-enter the credentials until they are locked out if too many attempts are made [5]. The SSO token

generated at the start is loaded with user-identifying information that is then passed around to different services to help the system identify what kind of privileges to give the user. It is also used to reduce the amount of times a user needs to authenticate themselves using credentials across multiple services [5].

Justification

Authorization is a key component of microservice security. Setting rules for individuals or groups of users on what they are allowed to access, alone makes sure a majority of a system's users do not end up with access to the wrong data. The extra layer of implementing SSO is a benefit to any microservices architecture. It allows the system's users to navigate the services provided seamlessly and securely, which as a result increases productivity as users don't need to re-enter credentials repeatedly and results in a better user experience as users get to manage fewer passwords across multiple services [5].

How to protect microservices from vulnerability exploits that allow unintended access to private information.

Even with proper authentication in place, without proper security built into microservices information can still be stolen. Attackers will always try to probe for vulnerabilities in a system, so it is important to protect as many possible entry points of attack as possible.

Confused Deputy Problem

As mentioned before SSO is useful to implement solid authentication and authorization. However, SSO is designed as a barrier to protect microservices from the user environment. This means user requests are sent through an SSO Gateway to the required service, where the responses would be sent back the same way. Here lies the Confused Deputy Problem, when an upstream service tricks downstream services into sharing data that it shouldn't [3]. This could be done by altering the request details sent through the SSO Gateway to the target service. An example of a normal interaction between the user and a service would be to ask for order details about a purchase made through a microservice. But what if the user were to alter the request details to ask for another person's order details? This dangerous vulnerability would allow attackers to extract a substantial amount of private data from the system.

JSON Web Tokens (JWTs)

A solution to this problem would be to use JWTs, these tokens allow the system to store information about a user in the system as a string that can be passed around [3]. For added security can be signed to signify that they have not been tampered with, and also the option for encryption further improves security. The key benefit of using JWTs other than being able to exchange information, is the guarantee the data being transferred hasn't been tampered with. Companies such as Microsoft use JWTs for authentication and secure access to company resources [6]. A type of JWT Microsoft uses is ID tokens. These are commonly used to give the system user account information to help decide on what level of authorisation they should be given, these ID tokens are not encrypted but do require a valid signature that is used to prove the token is authentic [6]. These tokens then can be passed from the user through the SSO Gateway to the required microservice, then if data is needed from one or more downstream microservices these tokens guarantee that the data brought back to the user isn't data they shouldn't have access to in the first place.

Justification

Developers need to be aware of possible vulnerabilities that occur when implementing microservices, as attackers are willing to break through many layers of security to get what they want. That is why using JWTs is a good way to secure the communication channels between the user and the system, as they remove the benefits of tampering with requests sent to the microservices. They are also a good solution to manage authorisation through microservices as the tokens can identify the user, which enables the system to delegate the required privileges to that specific user.

Conclusion

In conclusion, our extensive examination of security issues within the domain of microservices architecture has not only successfully identified some of the potential threats but also provided analyses and justifications for potential solutions. It is evident that actively being aware of the potential security issues and integrating the suggested solutions into future microservices, is a great step towards developing secure software systems for users and organisations.

Reference list:

[1] A. N. Dias and P. Siriwardena, "Microservices Security in Action," Manning Publications, 2020.

[2] P. R. Brandão, "The Importance of Authentication and Encryption in Cloud Computing Framework Security," Int. J. Data Sci. Technol., vol. 4, no. 1, pp. 1-5, 2018. doi: 10.11648/j.ijdst.20180401.11

[3] S. Newman, “Building microservices: Designing fine-grained systems,” Second Edition. Sebastopol, CA: O’Reilly Media, 2021

[4] "Alongside the traditional password, users can enable two-step verification for added security on devices within their Google Account." Google, [No publication date]. [Online]. Available: https://about.google/stories/passwort/#:~:text=Alongside%20the%20traditional%20password%2C%20users,devices%20within%20their%20Google%20Account. Accessed on: Oct. 16, 2023.

[5] "Single Sign-On (SSO) - Amazon Web Services (AWS)." AWS, [Online]. Available: https://aws.amazon.com/what-is/sso/. Accessed on: Oct. 16, 2023.

[6] "Tokens overview - Azure Active Directory B2C | Microsoft Docs," Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory-b2c/tokens-overview. Accessed on: Oct. 17, 2023.