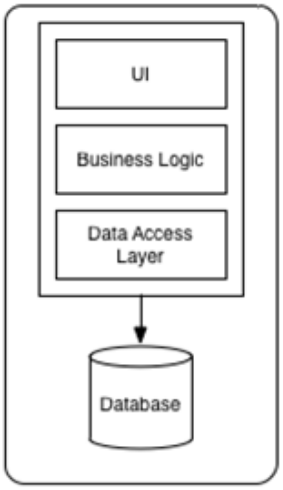
LSQ.IO

SECURITY ANALYSIS

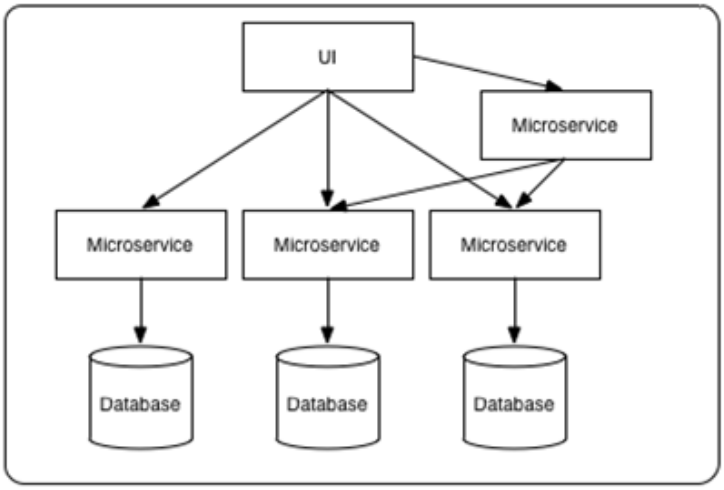
August Tan, Piyush Jadhav
Application Security
October 6, 2015

Introduction

- Microservices has become an emerging architecture that many companies have evolved from using in place of a monolithic architecture. Big names like Netflix, Ebay, and Amazon have begun using it.
- One of the drawback with this system is the amount of complexity that is necessary to code in that environment. Another drawback is the amount of memory consumption that is required for running the multiple microservices.
- The purpose of LSQ is to remedy these problems by making it PaaS to manage all your microservices. However, with this new service, comes many security concerns?



Monolithic Architecture



Microservices Architecture

- This is an image of the microservice architecture vs. the monolithic architecture. In this image, you can see that each microservice is a self-container application that the UI can access, compared to the Monolithic Architecture which does not utilize separate microservices and has all the components of the application working in the same container.

Architecture Overview

- This startup runs a Microservices Platform which allows developers to divide their application into many separate smaller services that all run their own processes.
- Microservices serves as a framework for different aspects of your application such as email, server, and user authentication.
- This allows each component to be a self-contained unit that manages its own resources and interacts with other microservices via standard http connections.

- Http has the advantage of being easy to use thus allowing us to connect to the application itself and any other third party API's it may be using.

- One thing to consider is that the microservices can be numerous when used for large scale applications so to make it easier for one microservice to find another each one will "announce" their presence when they are searching for other microservices. This is done in a process called, "service discovery" which is a new field of research that is still being looked into.

- Something unique about this architecture is the fact that each microservice will manage its own database. One of the problems with developing in such an environment is the fact that LSQ is only offering an architecture for its basis and it contains only a handful is microservices that need adding to.

Security Concerns

Deployment Security

- LSQ is developing a platform and developing environment for microservices and as such, they should give their clients an architecture that is easy to code in while minimizing security risks.

- One of the main issues with this microservice framework would be an attacker posing as a client. This attacker could then initiate an internal DDos attack. They could abuse LSQ's cloud infrastructure by spawning a large amount of microservices, say millions, which would then overload their servers and possibly crash them. This could be remedied by LSQ throttling the amount of containers that they allow on their cloud service per user.

- In addition to the concerns with the infrastructure of their cloud service, LSQ should be doing their best to mitigate any causes developers may have for making vulnerable containers. Meaning, that it is LSQ's responsibility to develop a base microservice that would ideally handle as many of the security vulnerabilities as possible. This includes things like what level of access would an attacker have if they are able to establish a connection to a microservice.

- Is it just that single microservice that is compromised or can the attacker then use that microservice to communicate false information to other microservices? To remedy these issues, LSQ should only allow a microservice to be used if it is called or pinged by another microservice. That way a compromised microservice container cannot mess with other containers by transmitting them faulty http requests. For LSQ, it is important for them provide a base platform that manages security for its users as well as keep their cloud service secure.

Unauthorized access to the services

- Every call to each service should be authenticated. Many systems have an authentication gateway that authenticates the initial API call. The Internal calls are not authenticated. Thus if anyone has access to the network, they can access any service inside including the one that contains the sensitive data.
- There should be a secure way for the services to authenticate each other. One way to do that is to user SSL certificates for authentication
- The Services should always pass the identity of person who originated each call while making calls to internal services. Even if the information is not directly required by the service. This ensures all calls in the systems are accounted for.
- Services should NOT let their callers access all the APIs that a service offers. They should provide access to just the ones it needs to fulfil its function.
- If an attacker owned a service, there should be some mechanism to stop them from requesting anything from its downstream services.
- The request received, should be checked for tampering.
- Services should be protected from replay attacks. HTTPS and TLS/SSL can be used to protect against replay attack.

Password Security

- Keep up to speed with the state of the art in password storage
- Using just salted password is not enough these days. The attackers, with today's hardware can calculate ridiculously high number of hashes quickly. Below are the numbers that demonstrate how many hashes can be generated in a second on a 25-GPU rig.

Scheme	**Tries/sec**
NTLM	350,000,000,000
MD5	180,000,000,000
SHA1	63,000,000,000
SHA512Crypt	364,000
Bcrypt	71,000

- Therefore it advised that you use modern techniques like scrypt ,bcrypt or PBKDF2
- When a password is entered wrong, what feedback do you give? Is there a difference in response time if the user does not exist in the database? This could be used for username enumeration.

Security against DDOS Attacks

- With the sheer volume of today DDOS attacks, it is very difficult to keep them out. Akamai Technologies shared new details recently of an existing botnet that is now capable of launching 150+ gigabit-per-second (Gbps) DDoS attacks from Linux systems infected by the XOR DDoS Trojan.

- There should be a clear process defined in event of a Denial of service attack.

Script Insertion

- Various Script injection attacks are possible. Therefore it is very important to ensure that the input data is properly sanitized.