

Microservices and Security

Memilavi
www.memilavi.com



Security Challenges with Microservices

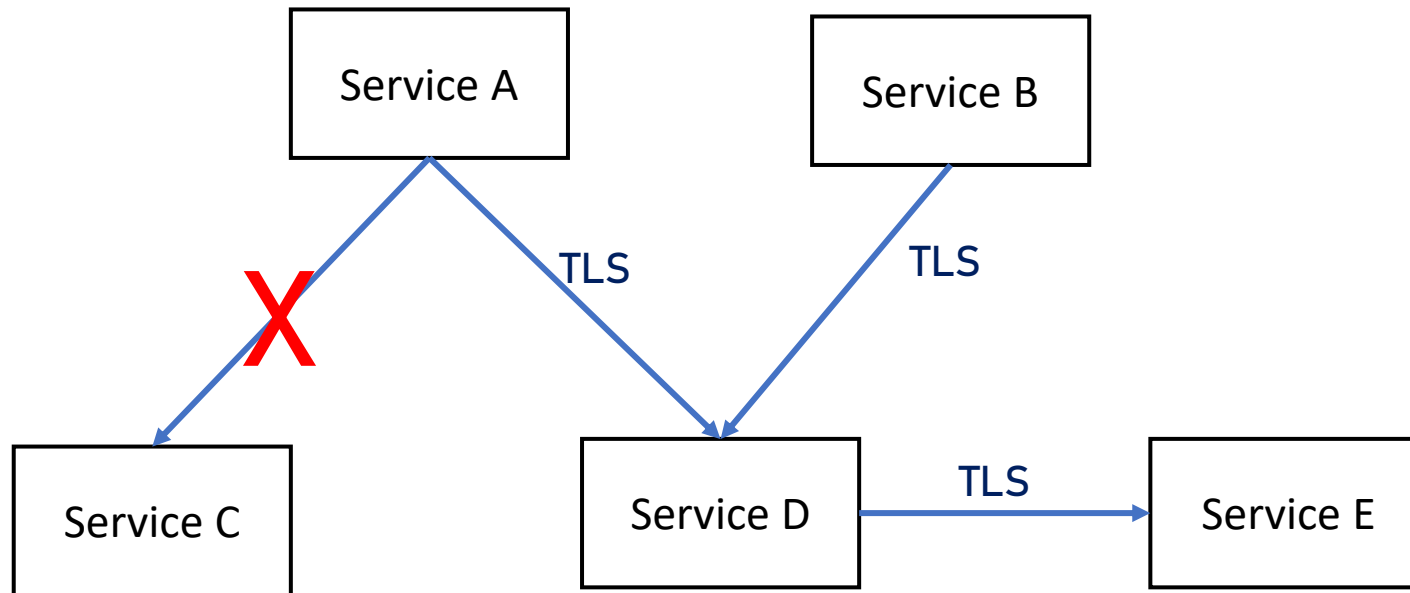
- Microservices system can have a lot of moving parts
 - All of them need to be secured
- The basic principles are the same but need to be adapted to microservices

Securing Microservices Network

- Communication between microservices is the most important element in microservices' systems
- Must be as secure as possible

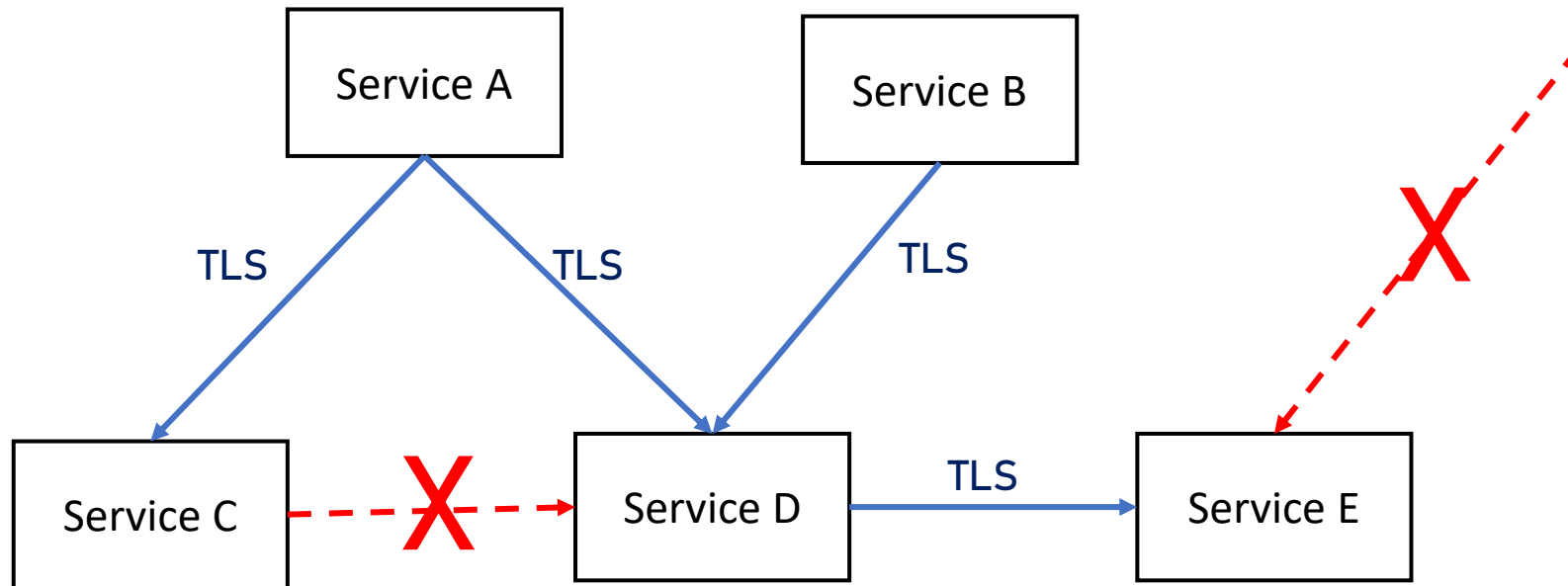
Securing Microservices Network

- Implement TLS



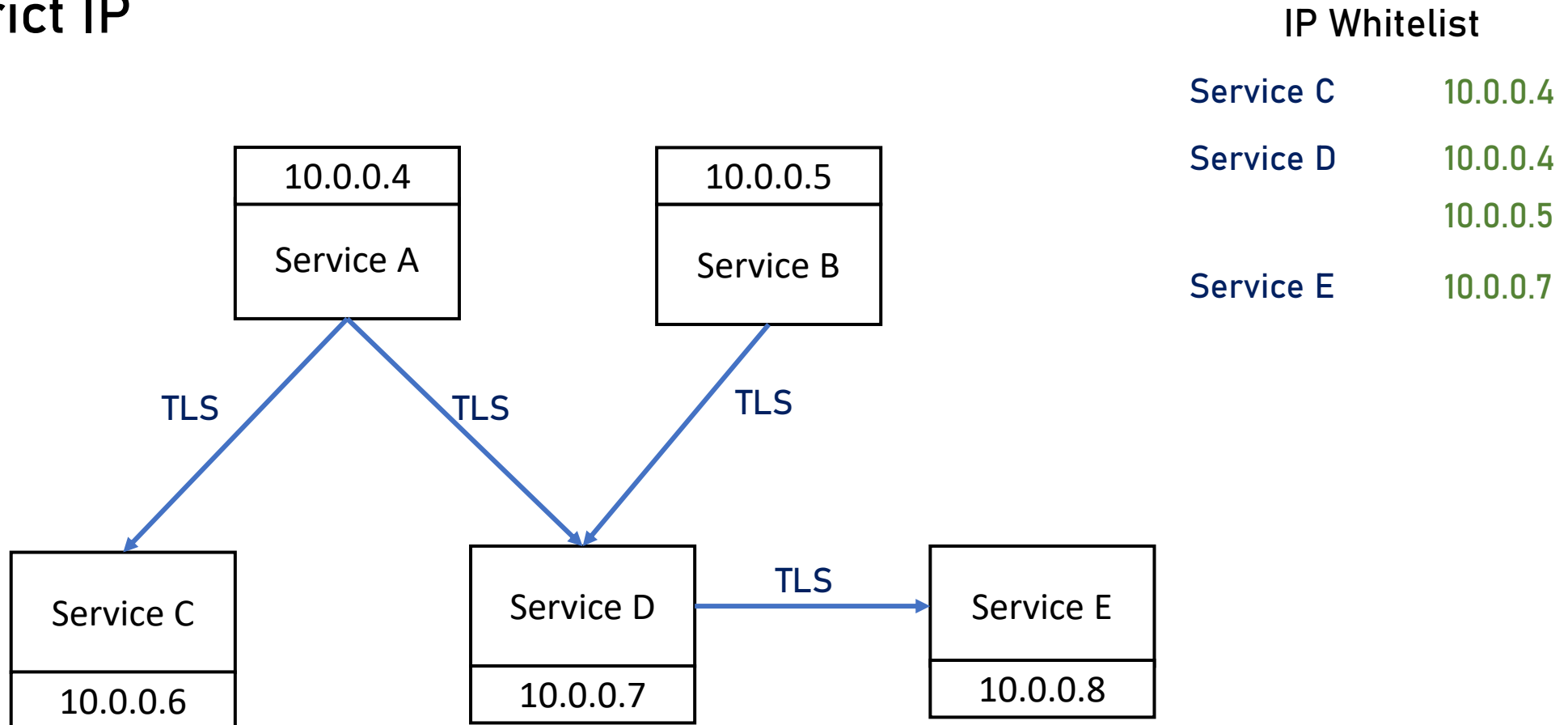
Securing Microservices Network

- Restrict IP



Securing Microservices Network

- Restrict IP



Securing Microservices Network

- IP restriction implementation
- Using code:

```
var whitelist = [ '10.0.0.4', '10.0.0.5' ];  
var http = require('http');  
var server = http.createServer(function(req, res) {  
    var ip = req.ip ||  
        req.connection.remoteAddress ||  
        req.socket.remoteAddress ||  
        req.connection.socket.remoteAddress;  
    if(whitelist.indexOf(ip) == -1) {  
        res.end();  
    }  
}).listen(80, '127.0.0.1');
```



Securing Microservices Network

- IP restriction implementation
- Using code:

```
public static bool IsIPAllowed(this HttpRequestMessage request) {  
    var whiteListedIPs = ConfigurationManager.AppSettings["AllowedIPs"];  
    if (!string.IsNullOrEmpty(whiteListedIPs)) {  
        var whiteListIPList = whiteListedIPs.Split(',').ToList();  
        var ipAddressString = request.GetIP();  
        var isInwhiteListIPList = whiteListIPList .Where(a => a.Trim()  
            .Equals(ipAddressString, StringComparison.InvariantCultureIgnoreCase))  
            .Any();  
        return isInwhiteListIPList; }  
    return true;  
}
```



Securing Microservices Ne

- IP restriction implementation
- Using infrastructure:



Add Access Restriction

General settings

Name ⓘ

Traffic from service A ✓

Action

Allow

Deny

Priority *

100 ✓

Description

Traffic from services A ✓

Source settings

Type

IPv4 ▾

IP Address Block *

10.0.0.4 ✓

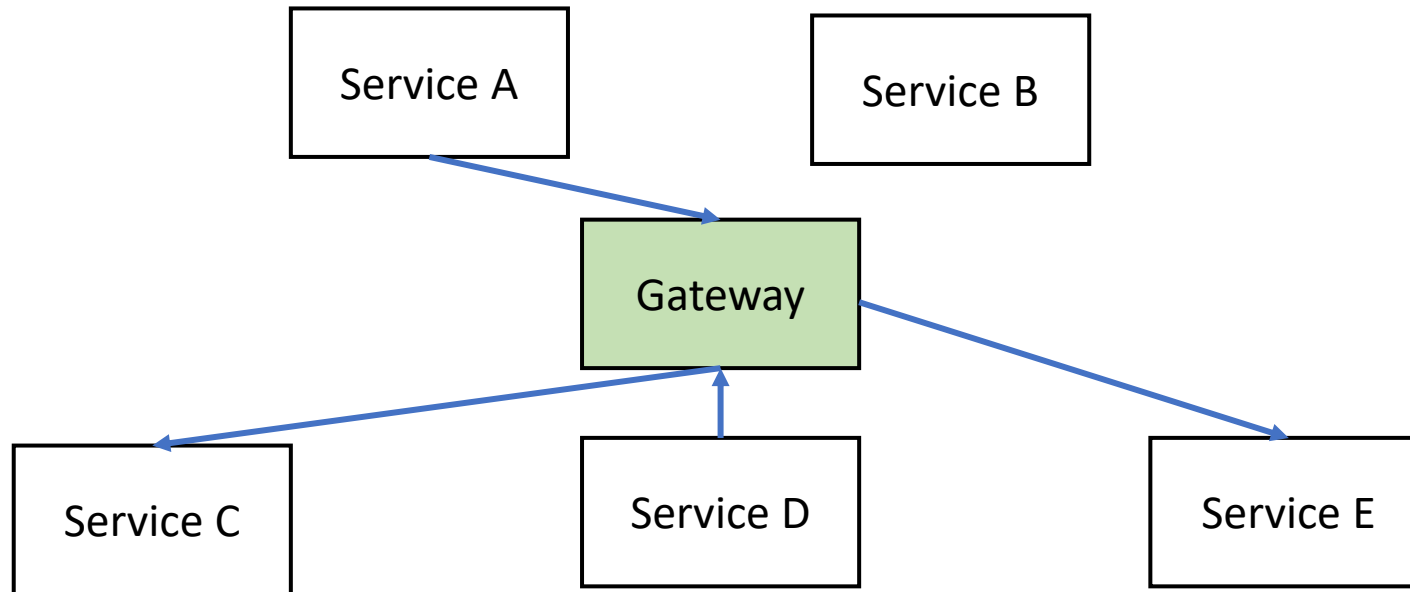
HTTP headers filter settings

X-Forwarded-Host ⓘ

Add rule

Securing Microservices Network

- Use Gateway



Securing Microservices Network

- Gateway provides:
 - Authentication / Authorization
 - Monitoring
 - Rate-limit
 - Header validation
 - ...and much more...

Securing Microservices Network

- Implementing API Gateway



Don't develop your own
API Gateway!

Securing Microservices Identity

- When calling microservice, the called service should know who calls it and whether the call is allowed
- Or: Authentication and Authorization
- Need to decide between two approaches

Securing Microservices Identity

Service Identity

The called service receives the identity of the service calling it, and decides whether this call is allowed

User Identity

The called service receives the identity of the end user calling it, and decides whether this call is allowed

Securing Microservices Identity

Service Identity

- Useful for managing service-to-service communication
- Limits access based on calling service

User Identity

- Useful for audit user actions in the system
- Limits access based on end user permissions

Can be combined, rarely done

Service Identity

- Used for non-interactive scenarios
 - No login screen for the user
- The calling service must have a token that can be used for authentication
- Usually API Key or access token

Service Identity

- API Key:
 - Provided by the called service or by central identity management
 - Sent along the request
 - The called service can manually validate it and decide whether it's allowed access
 - Quite easy to implement, no special libraries required

API Key

The screenshot shows the Google Maps Platform documentation page for the JavaScript API. A modal titled "Adding the API key to your request" is displayed in the center. The modal contains the following text: "You must include an API key with every Maps JavaScript API request. In the following example, replace `YOUR_API_KEY` with your API key." Below this text is a code block with the following code:

```
<script async defer src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"></script>
```

 The code block has a dark background and syntax highlighting. The text "YOUR_API_KEY" is highlighted with a red dashed box. To the right of the code block are icons for settings and a copy button. Below the code block is a "Send feedback" button. In the background, the documentation page is visible, showing the "Web > Maps JavaScript API" breadcrumb, a "Get Started" button, and a sidebar with "Guides" and "Tutorials" sections. The "Use API Keys" link is highlighted in the "Guides" section. The "Tutorials" section lists "All tutorials", "Add a marker to your map", "Cluster markers", "Real-time collaborative mapping", "Show current location", and "Use data with your map". The main content area of the page shows the beginning of a section titled "Using API keys" with the text: "Google Maps Platform products are secured from unauthorized use by restricting API calls to those that provide proper authentication credentials. These credentials are in the form of an API key a unique alphanumeric string that associates your Google billing account with your project, and with the specific API or SDK." The text "API key" is highlighted with a green box.

Google Maps Platform

Overview Products Pricing Documentation ▾ Blog Search Eng

Web > Maps JavaScript API [Get Started](#)

Guides [Refer](#)

Overview
Set up in Cloud Cons

[Use API Keys](#)

Tutorials
All tutorials
Add a marker to your map
Cluster markers
Real-time collaborative mapping
Show current location
▶ Use data with your map

Adding the API key to your request

You must include an API key with every Maps JavaScript API request. In the following example, replace `YOUR_API_KEY` with your API key.

```
<script async defer src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"></script>
```

Using API keys

Google Maps Platform products are secured from unauthorized use by restricting API calls to those that provide proper authentication credentials. These credentials are in the form of an **API key** a unique alphanumeric string that associates your Google billing account with your project, and with the specific API or SDK.

[Send feedback](#)

Service Identity

- Access token
 - Provided as part of OAuth2 authentication flow
 - Time limited
 - Requires deep integration with the authentication server
 - Not so easy to implement
 - More secure

Access Token



Documentation

 Search the docs

Getting started

Fundamentals



Developer Apps

Projects

Developer portal

Authentication

Counting characters

OAuth 2.0 Bearer Token

OAuth 2.0 Bearer Token authenticates requests on behalf of your [developer App](#). As this method is specific to the App, it does not involve any users. This method is typically for developers that need read-only access to public information.

This authentication method requires for you to pass a Bearer Token with your request, which you can generate within the Keys and tokens section of your developer Apps. Here is an example of what a request looks like with a fake bearer token plugged in:

```
1 curl "https://api.twitter.com/2/tweets?ids=1261326399320715264,127834746869091"
```



API calls using app-only authentication are rate limited per endpoint at the App level.

User Identity

- Uses the end user identity
- Interactive experience
 - Login page required
- Result of authentication is usually JWT
- The JWT is passed to the called service which validates it

Authentication vs Authorization

- Various approaches affect the authentication only
- Authorization is always part of the service itself

Securing Microservices Data

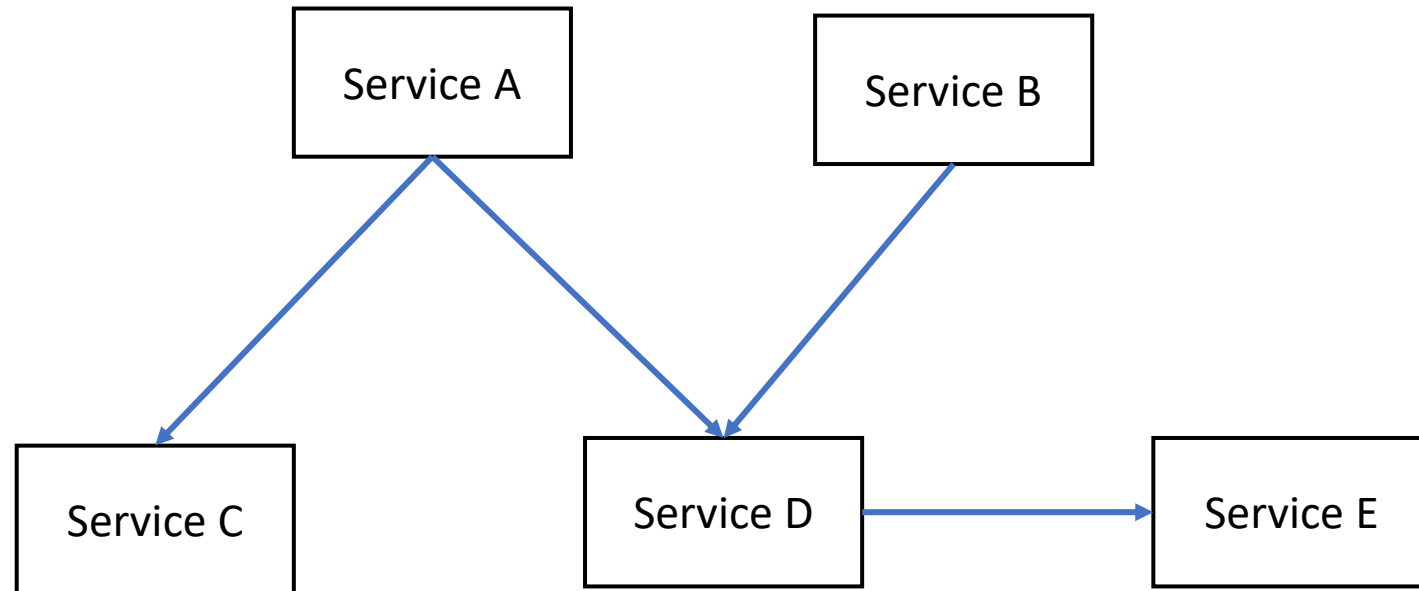
- Not really different from regular architecture
- No matter how many databases are in the system – all data should be encrypted
- All major databases support data encryption

External vs Internal Services

- Sometimes it's a good idea to make a distinction between two types of services

External
Called from outside

Internal
Called only from other services

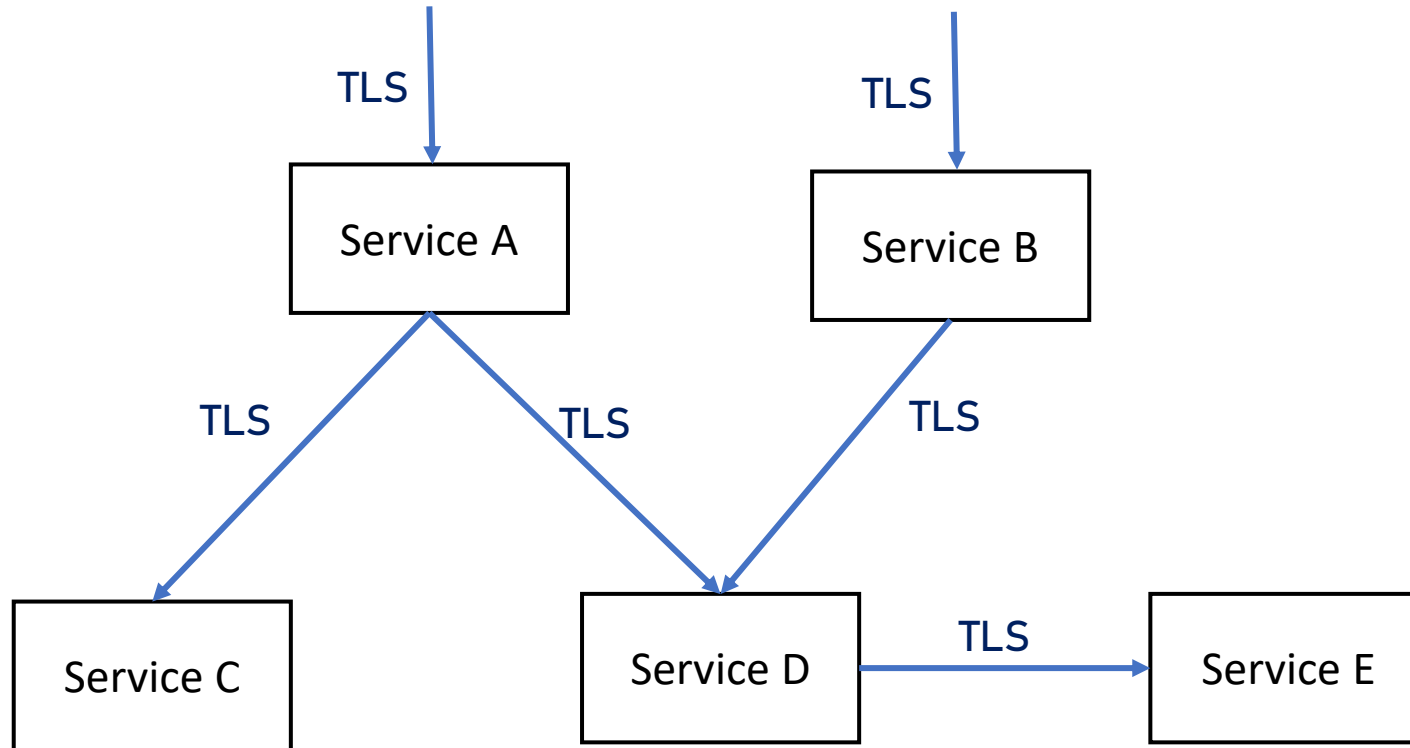


External vs Internal

- Questions to ask:
 - Do we need TLS all the way down?
 - Do we need end-user authentication for internal services?
 - How do we limit access to internal services?
 - Usually there's no point in limiting traffic to external services

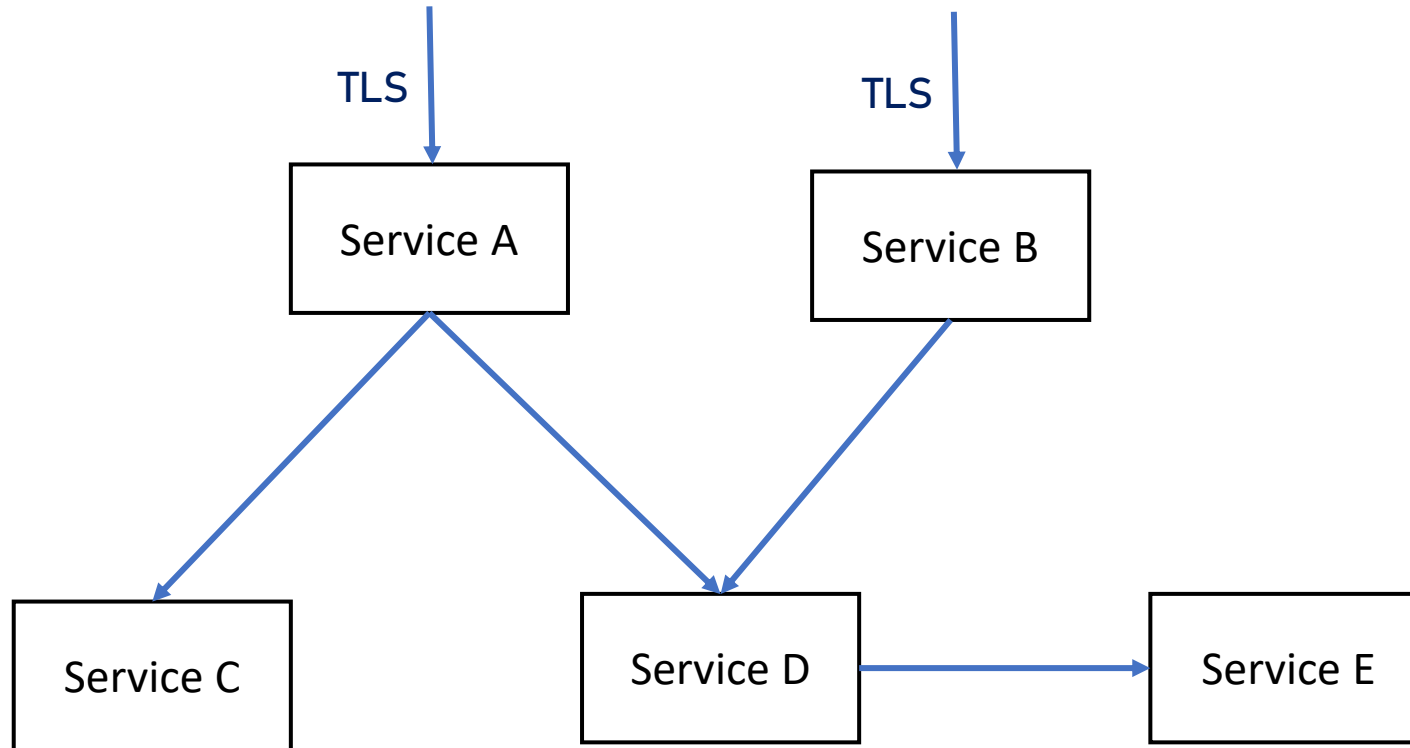
External vs Internal

- Scenario #1: TLS all the way down



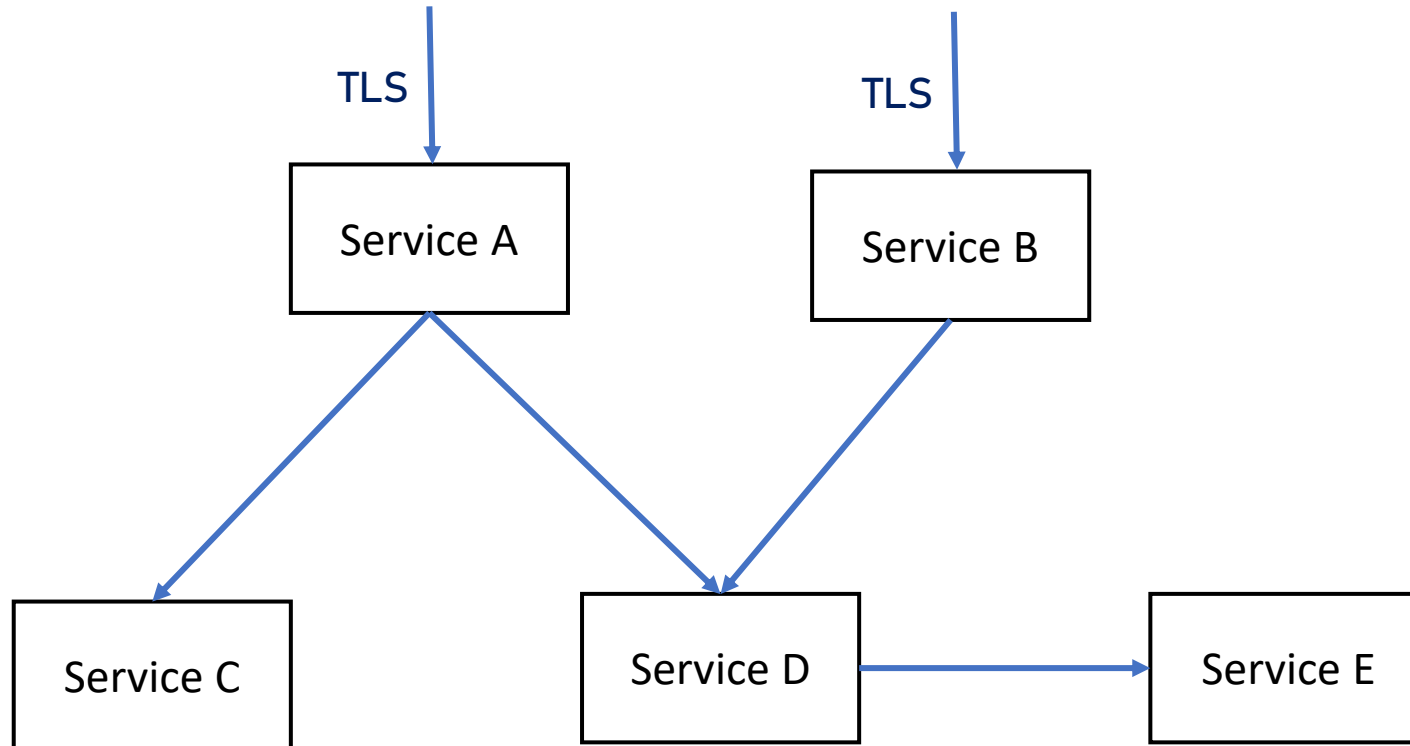
External vs Internal

- Scenario #2: TLS offloading



External vs Internal

- Scenario #3: TLS offloading with IP restrictions



IP Whitelist

Service C	10.0.0.4
Service D	10.0.0.4
	10.0.0.5
Service E	10.0.0.7

External vs Internal

- It's a mix-and-match
- Consult with the cyber security team in the organization