

# HOLD THE DOOR!

GUARDING BESTSECRET'S SECRETS  
WHILE ADOPTING THE POWER OF  
A MODERN IDENTITY PROVIDER



BESTSECRET



# Introduction

## Who are we?



Francisco Javier Roa López

Principal Engineer @ BestSecret GmbH

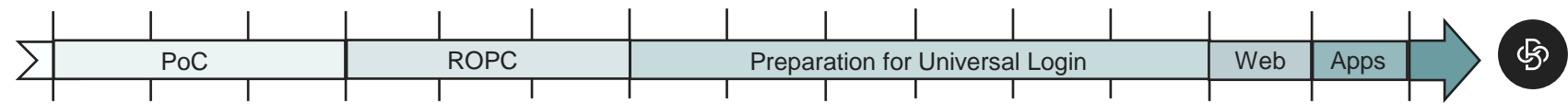
francisco.roa@bestsecret.com



Maximilian Kiesel

Software Consultant @ TNG Technology Consulting

maximilian.kiesel@tngtech.com



## What is this story about?

Current system is working well and performant  
Users can log in securely and attackers are blocked  
Success story with more and more users

To scale, business implement a model of distributed systems.  
Need to shift to a more scalable architecture **without any disturbance for the users**

How to make the communication secure?

- between the user and the system
- between the components of the system

# THE MISSION

S M O O T H   T R A N S I T I O N   T O   A   S C A L A B L E  
A U T H E N T I C A T I O N   M O D E L



BESTSECRET



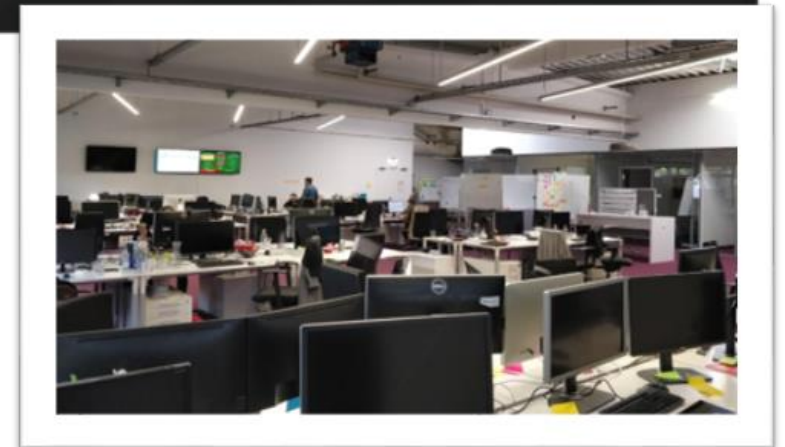
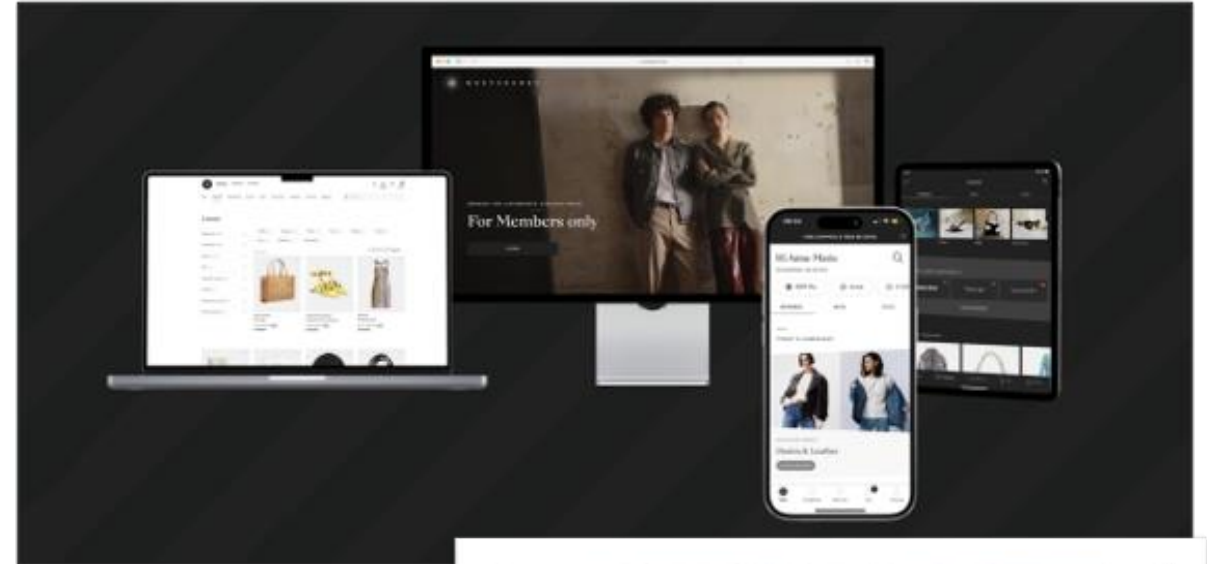
# The first steps

## A short history of BestSecret

- We have nearly 100 years' heritage in the textile industry.
- We reinvented the **off-price market** creating membership-only outlets with a customer referral model in the 80's.
- The online platform BestSecret was launched in 2007

The company was built like many start-ups with focus on:

- Quick launch, Lean Approach: Agility and Flexibility
- Brand and Culture Building, Business Growth, Innovative Spirit
- Based on a monolithic commercial e-commerce platform hosted on-premise





# BestSecret at a glance

**Our core KPIs**  
In 2023

**27 countries**

GEOGRAPHICAL  
REACH

**~2,300**

EMPLOYEES

**90+ nations**

IN OUR TEAM



**Organizational challenges for webshop development:**

Multiple feature teams



A lot of coordination needed for maintenance and deployment

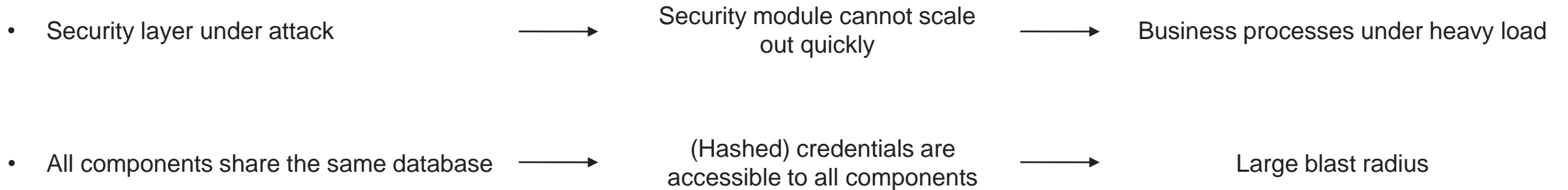


# Security: Becoming a Visible Target

## General security threats for internet services

- Brute-force attacks
- Credential stuffing
- Bot-network attacks
- Distributed denial-of-service attacks (DDoS)

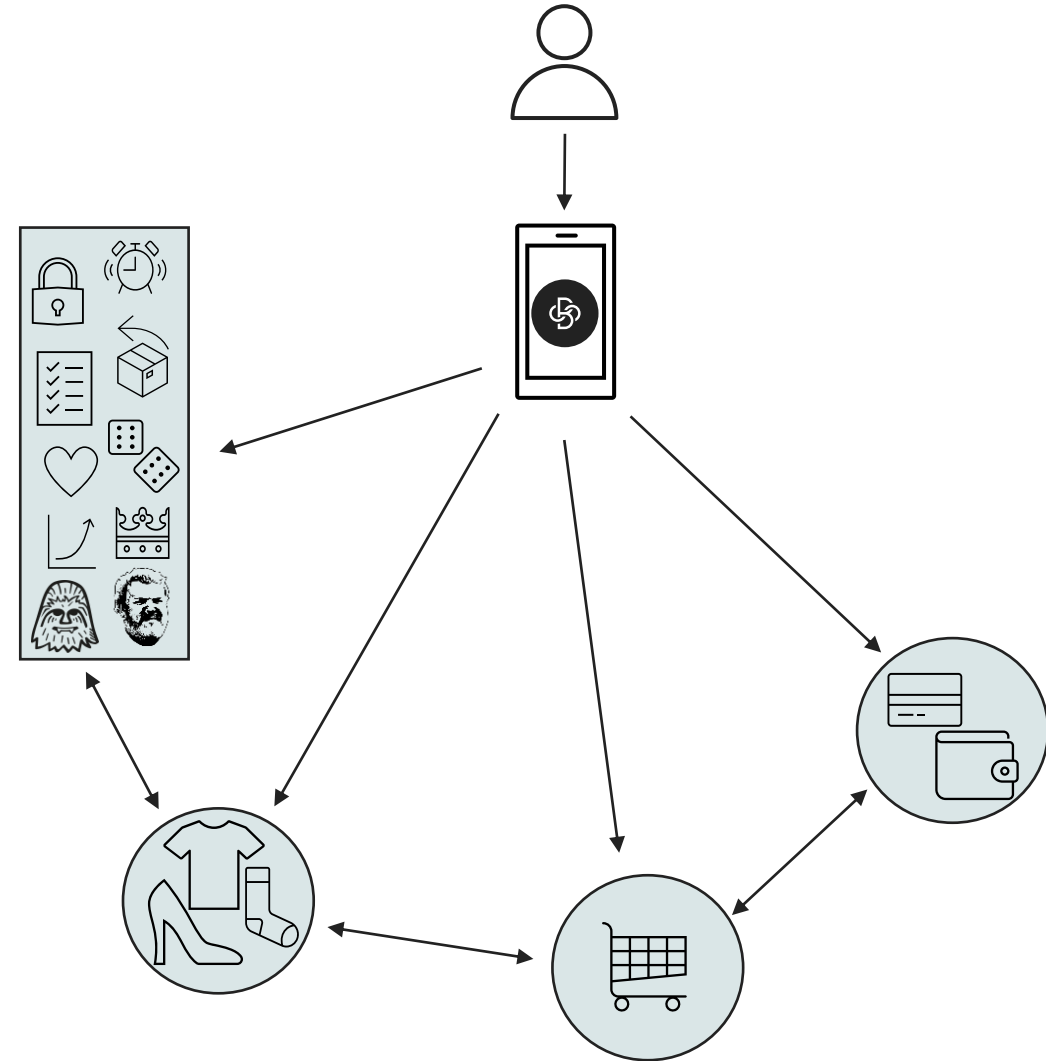
## Disadvantages of a monolithic system



# Distributed Authentication: Breaking the Monolith

## Adopting cloud-hosted domain services

- Move to smaller, more manageable services
  - Identify boundary contexts for service decomposition
  - Design each service around a specific business capability
  - Ensure that each service is loosely-coupled
- Shift incrementally
  - Start with implementation of new features
  - Then slice out other parts of the monolith
  - Good opportunity to streamline business processes



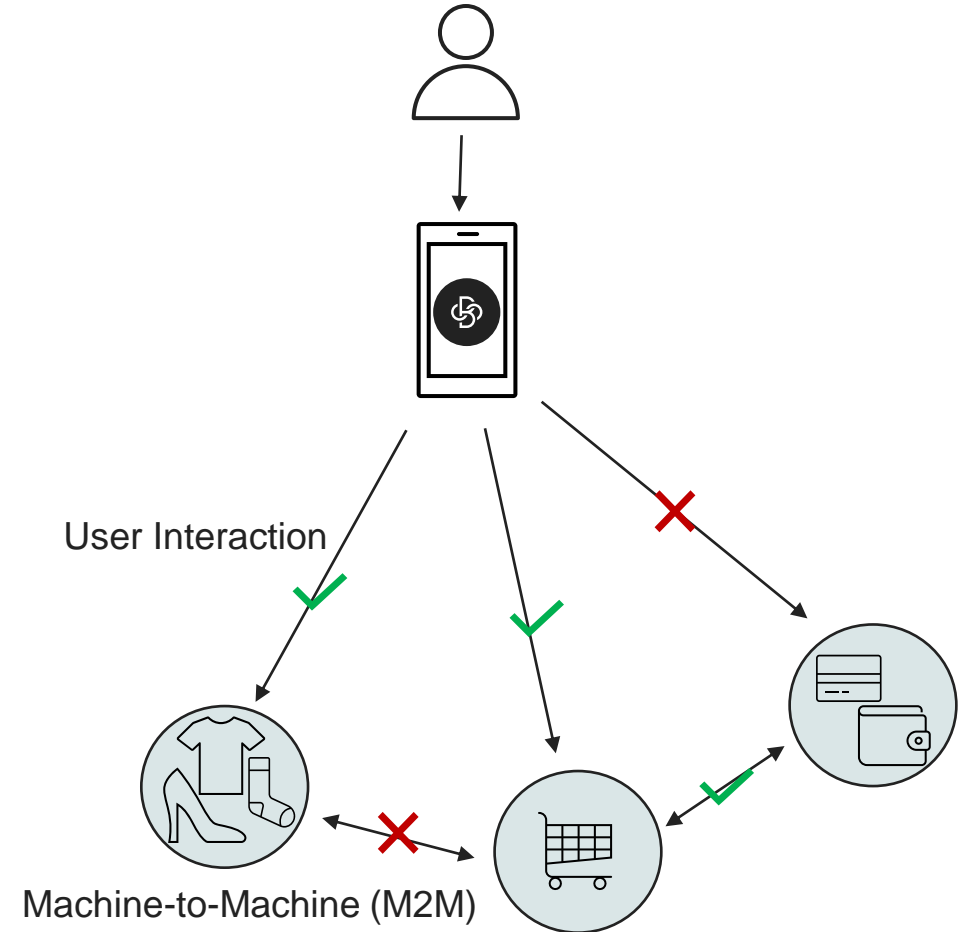




# Distributed Authentication: Breaking the Monolith

## Authentication is crucial in a distributed service architecture

- **Distributed Nature:**  
Secure communication between services hosted on different servers
- **Security:**  
Protecting against unauthorized access and potential breaches
- **Identity Verification:**  
Maintaining trust and security *within the system*
- **Access Control:**  
Determine what actions an authenticated *user or service* is allowed to perform



# KEEPING THE BAD GUYS OUT

ADOPTING INDUSTRY STANDARD PROTOCOLS FOR  
AUTHENTICATION



BESTSECRET



# Authentication vs. Authorization

## *Authentication*

The act of identifying a user.

## *Authorization*

The act of allowing or denying a user's access rights.

"I know you!"



"YOU SHALL NOT PASS!"



# Authentication

Some examples for authentication mechanisms

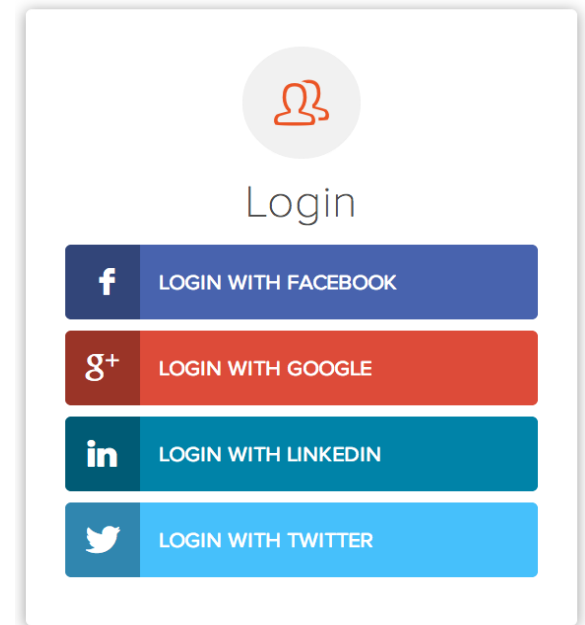


Biometrics

Username: president  
Password: 12345



Hardware Token



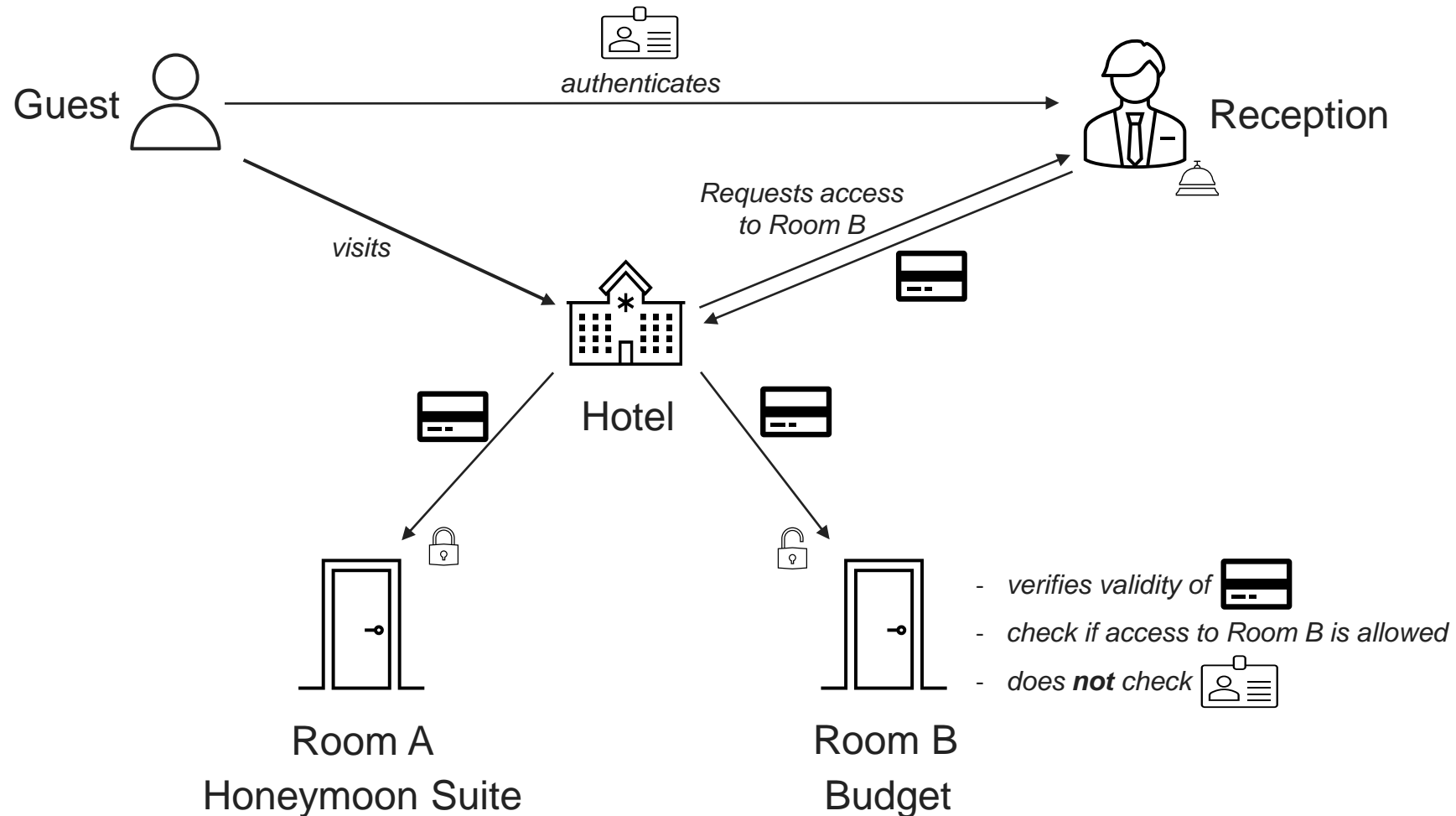
Social ID Provider

*Multi-Factor Authentication* for improved security!



# Authorization

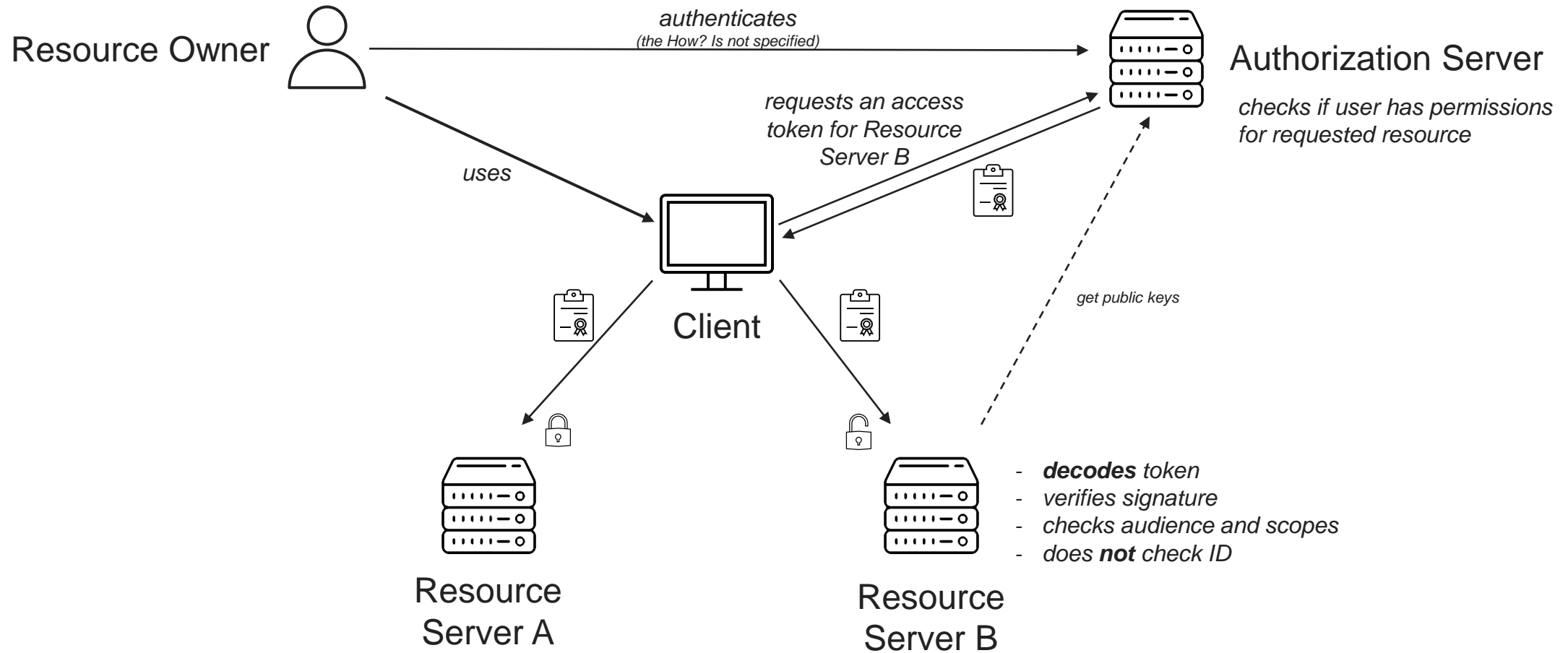
## OAuth2 in a Nutshell: Hotel Check-In





# Authorization

*OAuth2 = "Open Authorization"*





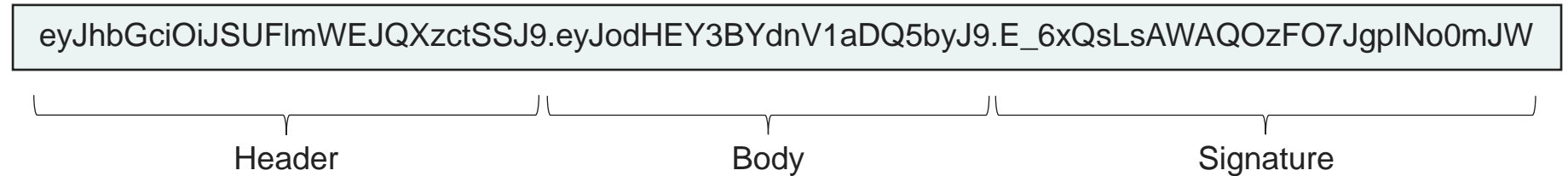
# Adopting modern Authentication Protocols

## Token-based authentication

- Secure and flexible authentication mechanisms suitable for distributed services
- Scalability

### Access Token

- Json Web Token (JWT)
- Base64 encoded
- Short-living



### Refresh Token

- Random unique string
- Long-living
- Exchange for new access token

VDYs8N0Ail6TdTIYwRp3zDY

```
{
  "iss": "https://login.bestsecret.com/",
  "sub": "12345-6789",
  "aud": "https://www.bestsecret.com/v1",
  "iat": 1718862478,
  "exp": 1718863078,
  "scope": "shop_access"
}
```

# THE JOURNEY

T H E R E   A N D   N O T   B A C K   A G A I N



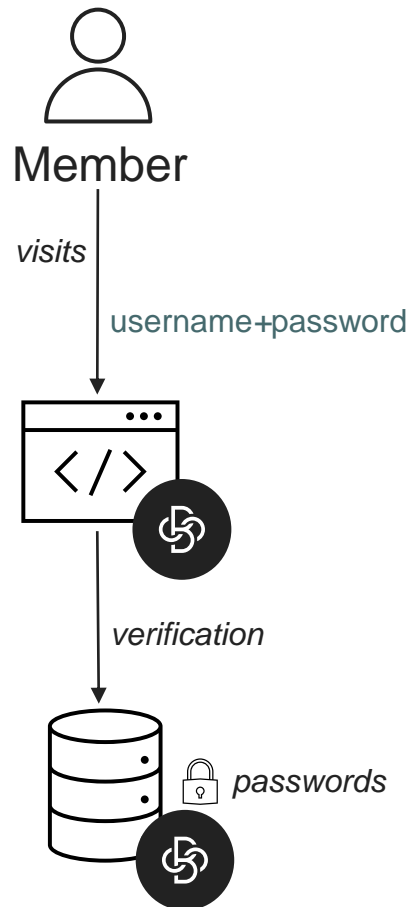
BESTSECRET





# Initial Situation

## Authentication at BestSecret before 3rd party identity provider



The monolith had a simple but effective authentication, based on Spring Security.

With:

- Credentials are stored in DB  
(as salted hashes)
- Self-implemented attack protection  
(supported by CDN provider)



# Act I

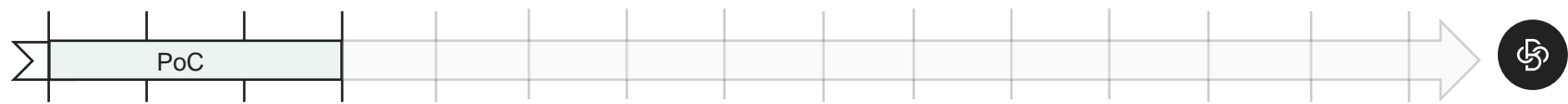
## Proof-of-Concept Phase

Questions:

- Authentication system inside our infrastructure or provided by a commercial solution?
- Can we refactor our monolith to integrate an identity provider?
- How does the final system should look and be maintained?

Idea: **Create a new service outside of the monolith as proxy**

- Hosted in Azure Kubernetes Service
- Fast and regular deployments
- Goal: Explore several services and migrate quickly from one to another



# Act I

## Outcome

- We don't want to maintain a self-hosted solution

- Decision to go with Auth0:



- Well-established in the market, top support
- GDPR Compliance: Servers within EU
- SLA of 99,99%
- Security Features

- Main goal: Migrate our customer database to Auth0 to enable the Attack Protections they provide for our customers

- Focus on seamless migration, no big-bang releases

How to get the members' credentials into Auth0's database?





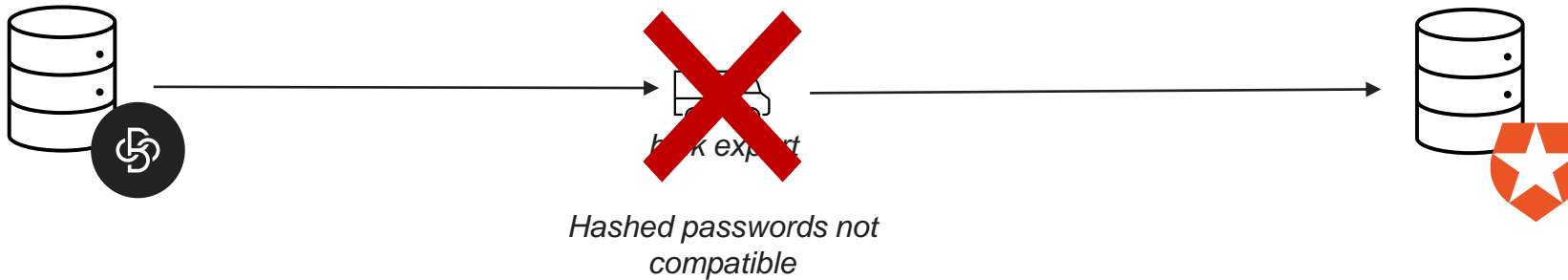
# Act II

## How to get the members' credentials into Auth0's database?

Important requirements:

- Zero Downtime, keep the business running
- Switch to the old system anytime : both databases need to be in sync
- Do not leave any customer behind, support legacy authentication system for web and apps

Bulk export and import of hashed passwords?

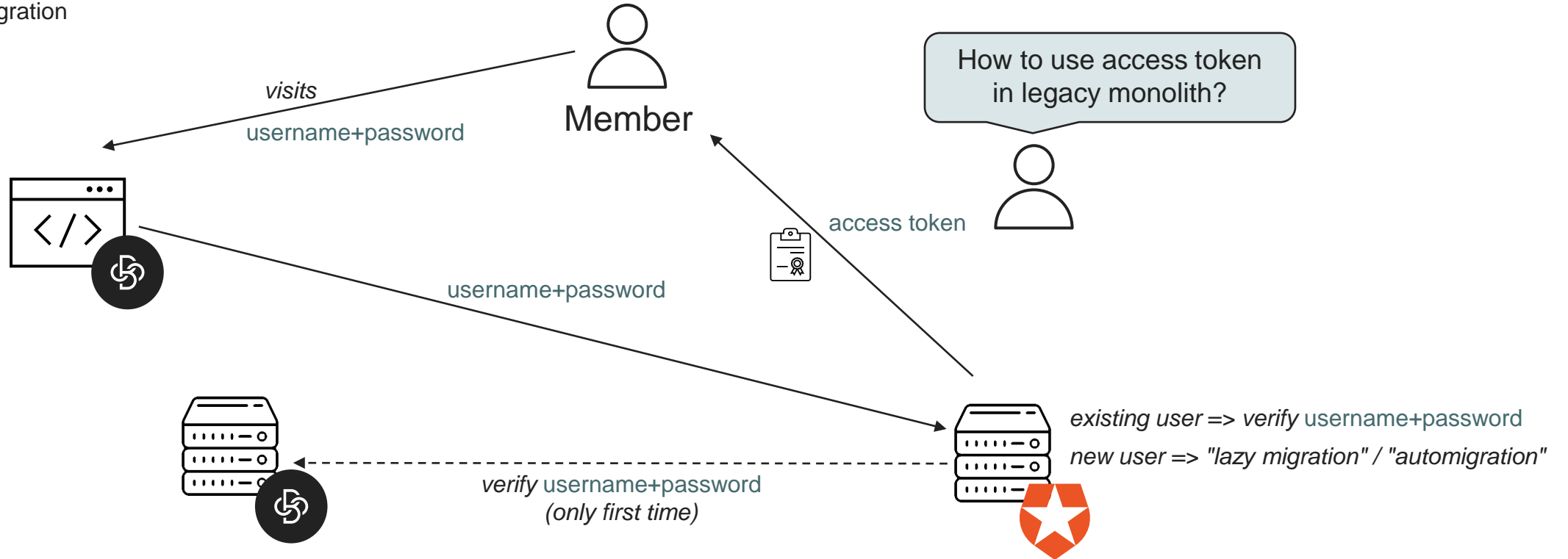


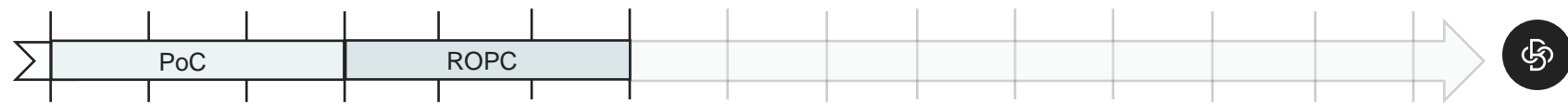


# Customer Data Migration Challenge

## Resource Owner Password Credential Flow (ROPC)

with lazy migration

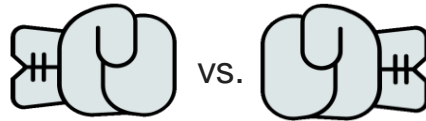




# Session Management Challenge

How to integrate session handling of monolith?

Domain Services



Legacy Monolith

12-factor apps

"VI. Processes

Execute the app as one or more **stateless** processes"

**Stateful** session



ACCESS\_TOKEN

JWT-to-Session-Mapping



JSESSIONID



REFRESH\_TOKEN

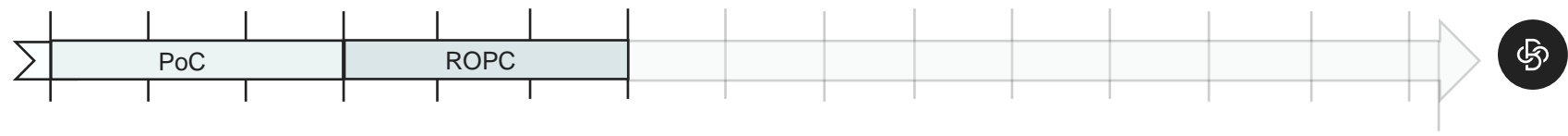
Delegation



REMEMBER\_ME

Sessions of monolith can be used with access tokens





# Rollout Challenge

How to deploy changes to production?

Rollout Plan:

- Controlled by feature toggles
- Starting with a small country
- Adding more countries step-by-step
- Intense monitoring
- Detecting and fixing of edge cases



# Act II

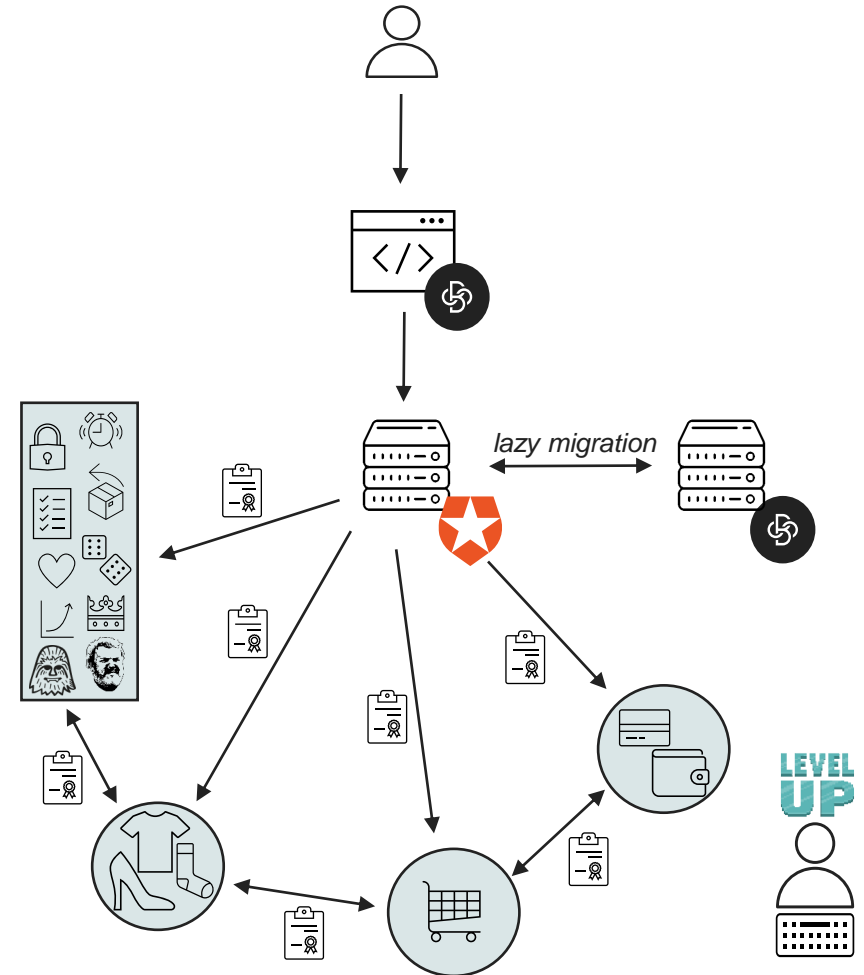
## Intermediate status

### Achievements:

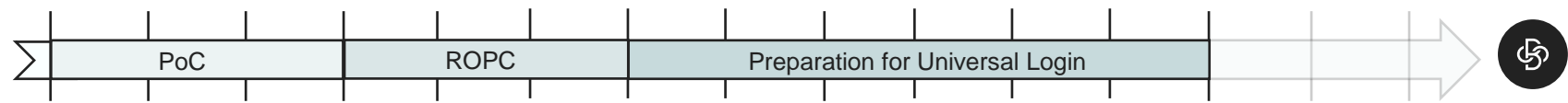
- We enabled authentication to distributed services using access tokens
- We created a mechanism so that members migrate into Auth0 passively
- We gained some experience in how to integrate with Auth0

### Open steps:

- Simplify token handling for domain services
  - Members to login at Auth0 directly via browser and native apps
- Complete isolation from the monolith







# Act III

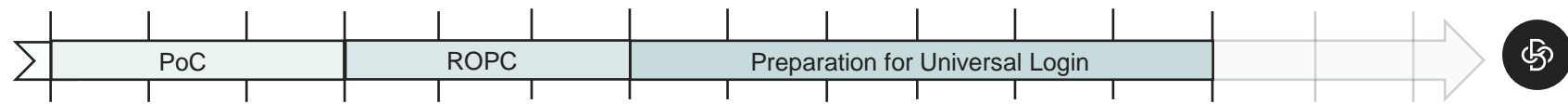
## Goal: Universal Login hosted at Auth0

For enabling all security features offered by Auth0, the login page must be hosted by Auth0



- Hosted at custom domain: *www.login.bestsecret.com*
- Serving emails from Auth0 via Azure Communication Services
- Connect monitoring and logging to DataDog
- Setting up test infrastructure





# Data Alignment Challenge

Which property can be used as username at Auth0?

- must be a valid **email format**
- must be **unique**



## Login name

- legacy members with non-email format
- unique



Login: Jane Doe  
 Email: jane.doe@example.com

Can only be changed by customer service!

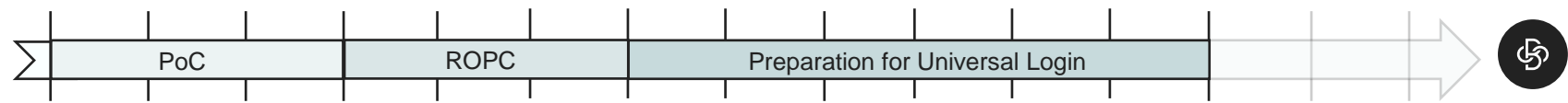
## Contact Email Address

- always valid email format
- not unique



Login: john.doe@example.com  
 Email: jane.doe@example.com

*"What does not fit is made to fit"*



# Data Alignment Challenge

## How can we use the login name as username at Auth0?

- must be a valid **email format**
- must be **unique**



Implement self-service to change *login name*

- only validly-formatted email addresses allowed
- use it also as *contact email address*



Deactivate self-service to change *contact email address* separately



Inform all members with distinct *login name* and *contact email address* to change it to one unified value



Give members some time to react



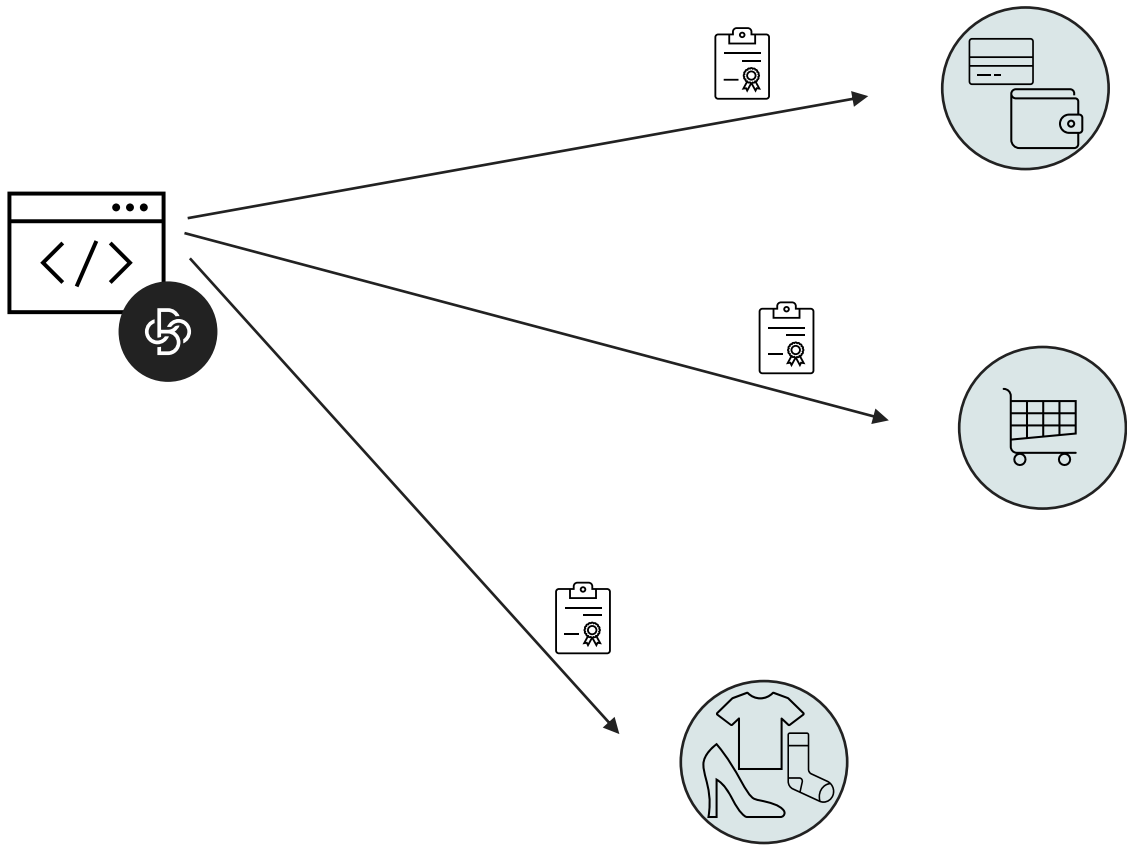
**Unique and validly-formatted *login name*, also used as *contact email address***





# Decentralized Identity Challenge

## How to authenticate at domain services?



Does every service team have to implement an authentication layer itself?



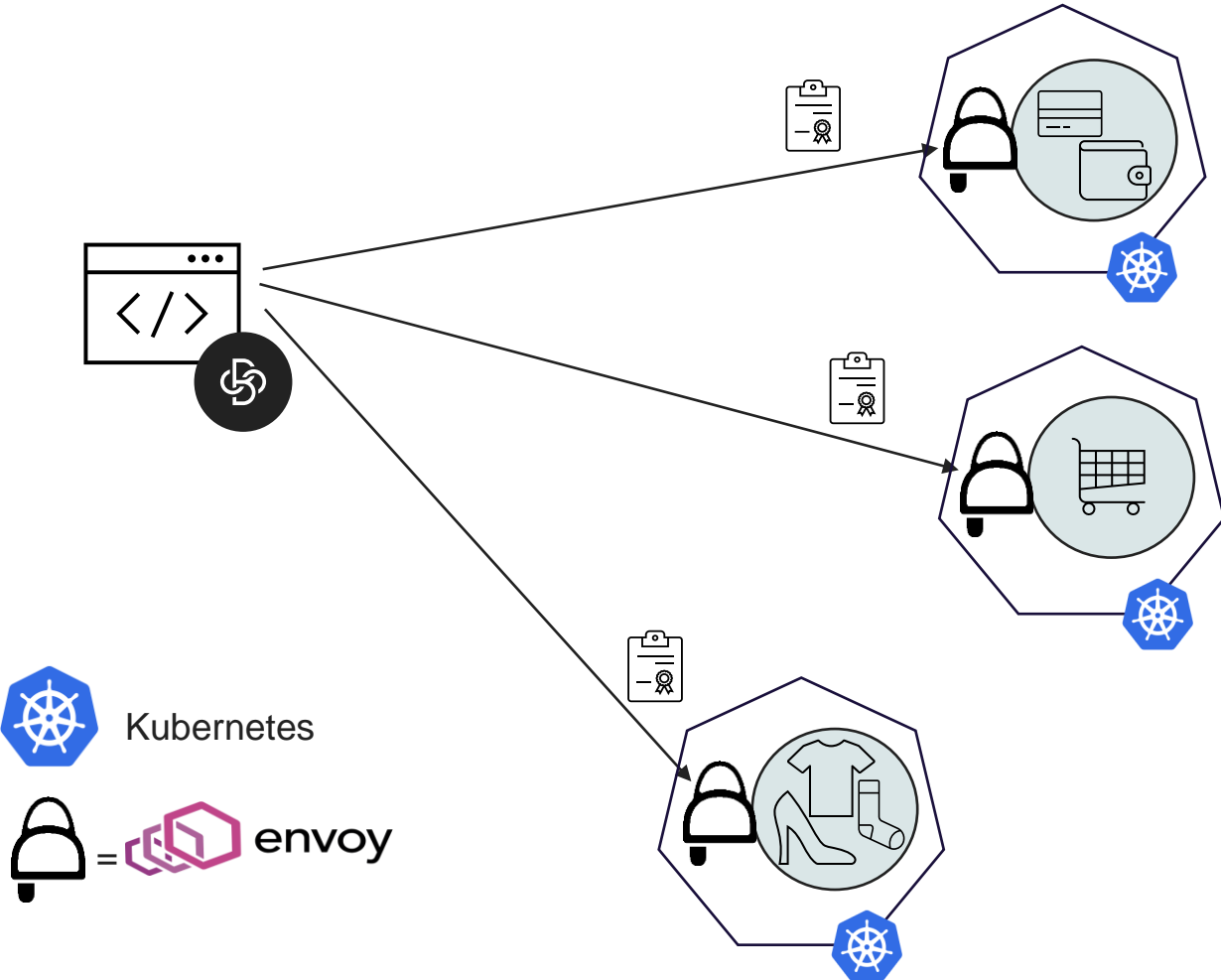
=>  envoy proxy:

- Handles authentication
- Forwards permitted requests to service

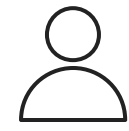


# Decentralized Identity Challenge

## Authentication at domain services with sidecars



How to enable sidecar for new domain services?

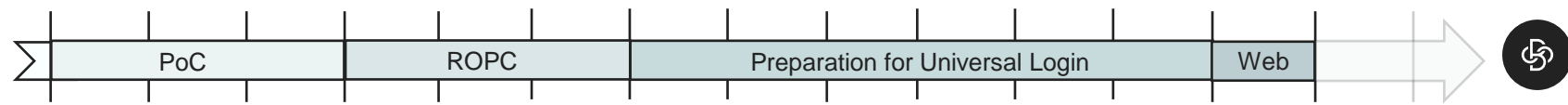


1. HELM bestsecret-base-chart:


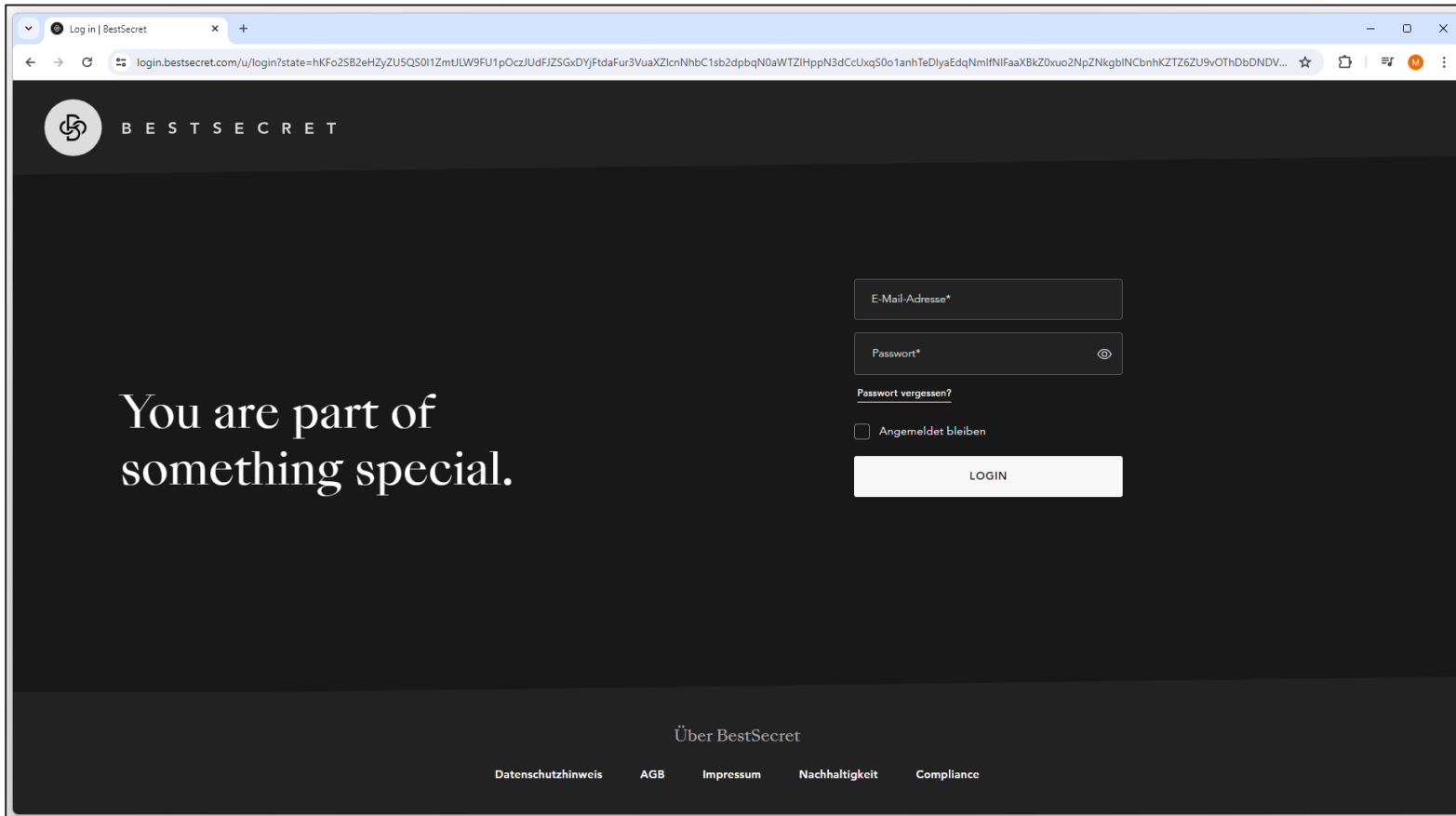
```
authSidecar:  
  enabled: true
```

2. Done ✓

# Act IV



## Goal: Web login



Rollout Plan:

- Controlled by feature toggles
- Starting with a small country
- Adding more countries step-by-step
- Intense monitoring
- Detecting and fixing of edge cases

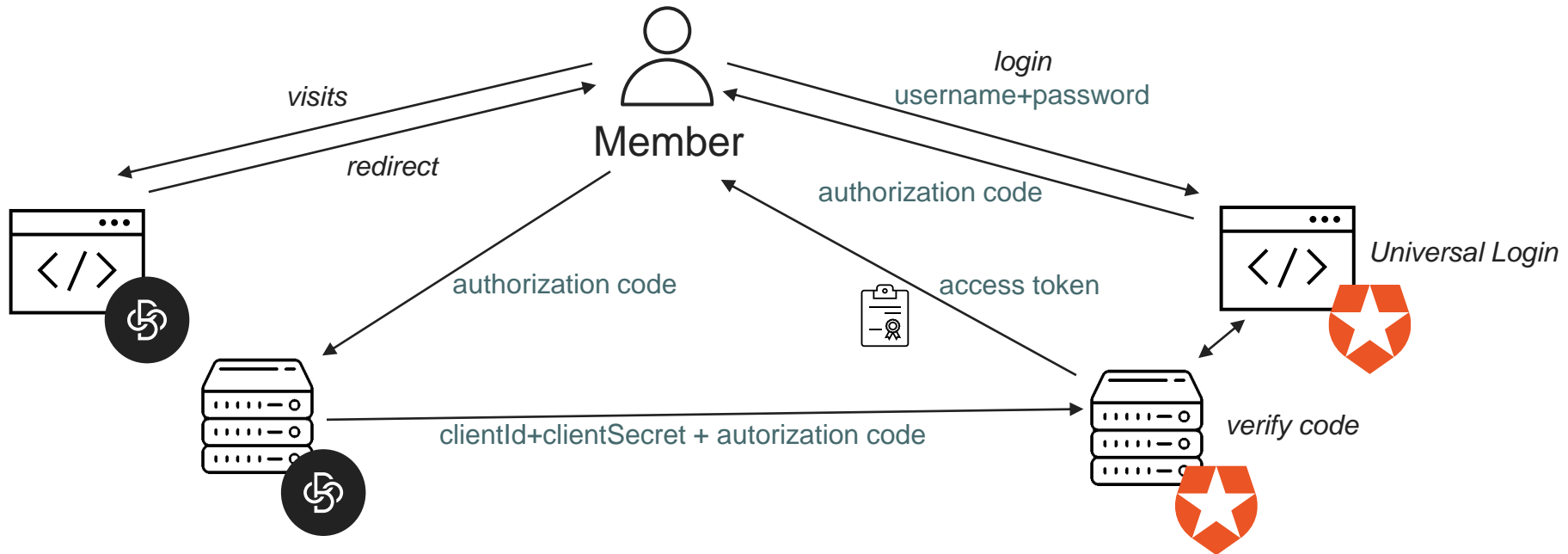
Login site hosted by Auth0!?



=> Authentication Code Flow

# OAuth2 Flows

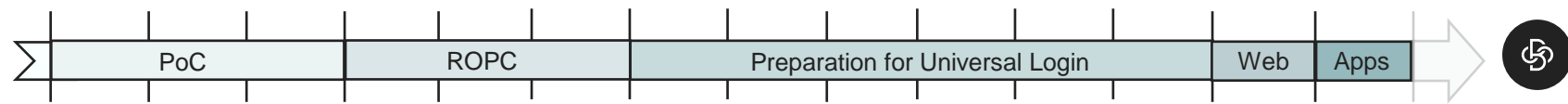
## Authorization Code Flow



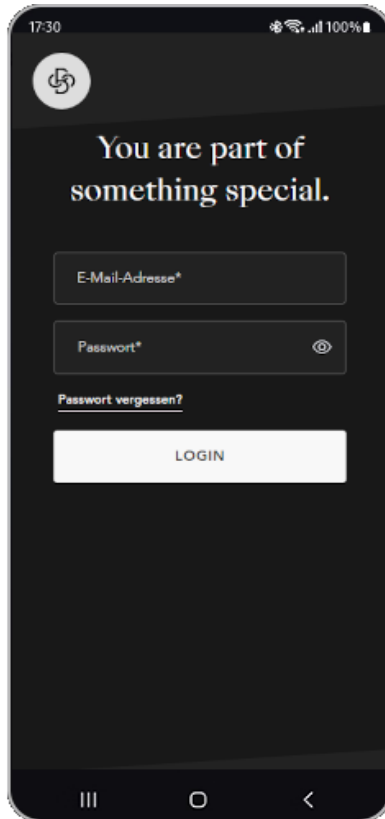
- Login page hosted and protected by Auth0
- Credentials entered at their site



# Act V




## Goal: Native apps login



Rollout Plan:

- Controlled by feature toggles
- Starting with a small country
- Adding more countries step-by-step
- Intense monitoring
- Detecting and fixing of edge cases

 Old login flow still maintained to support old app versions!

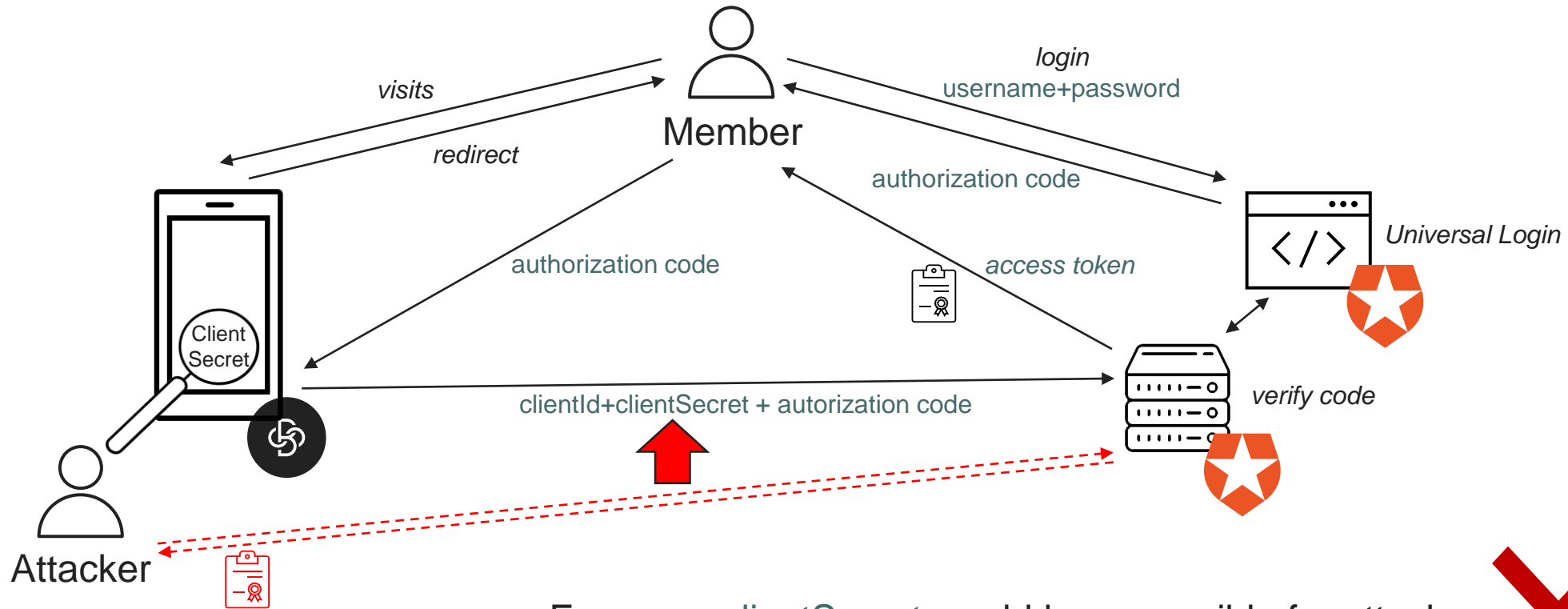
Can we use authorization code flow with apps?





# OAuth2 Flows

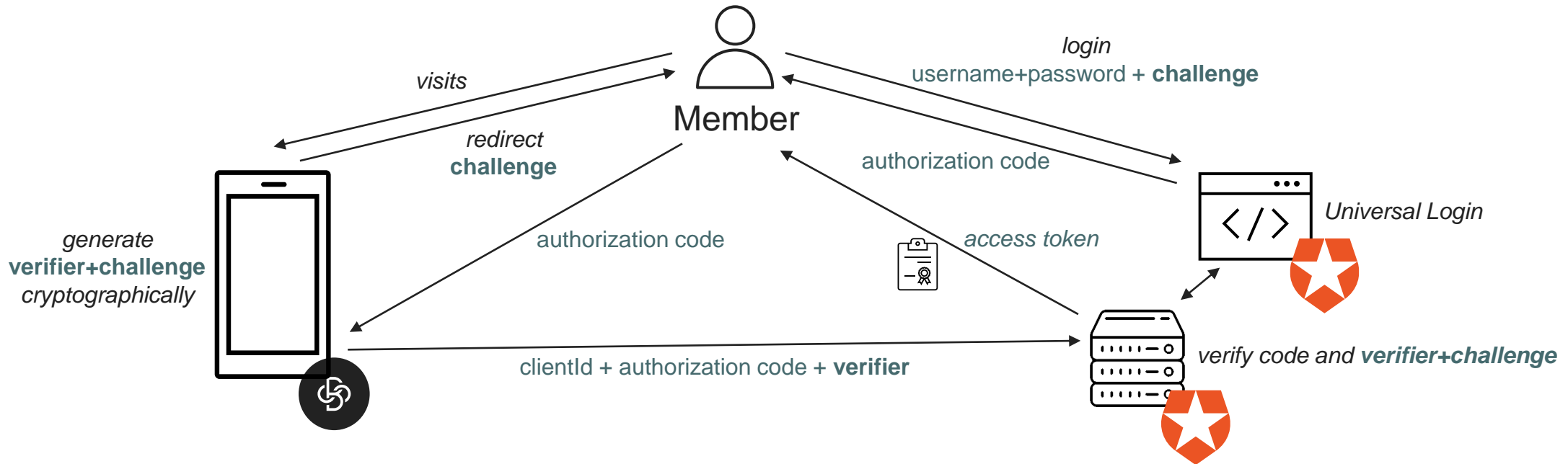
## Authorization Code Flow for the apps?



For apps, `clientSecret` would be accessible for attackers

# OAuth2 Flows

## Authorization Code Flow with Proof Key for Code Exchange



### Verifier and challenge

- one-time credentials
- sent over distinct channels





# Overview: OAuth2 Flows

## Which authorization flow do we need to implement?

Select the flow based on your use case!

no special requirements - as simple and secure as possible



Authorization Code Flow

Login in Web

Lazy Migration

client secret cannot be stored securely



Authorization Code Flow with Proof Key for Code Exchange

Login in Apps

forward user credentials



Resource Owner Password Credential Flow

Lazy Migration

I LOVE IT WHEN A PLAN COMES TOGETHER

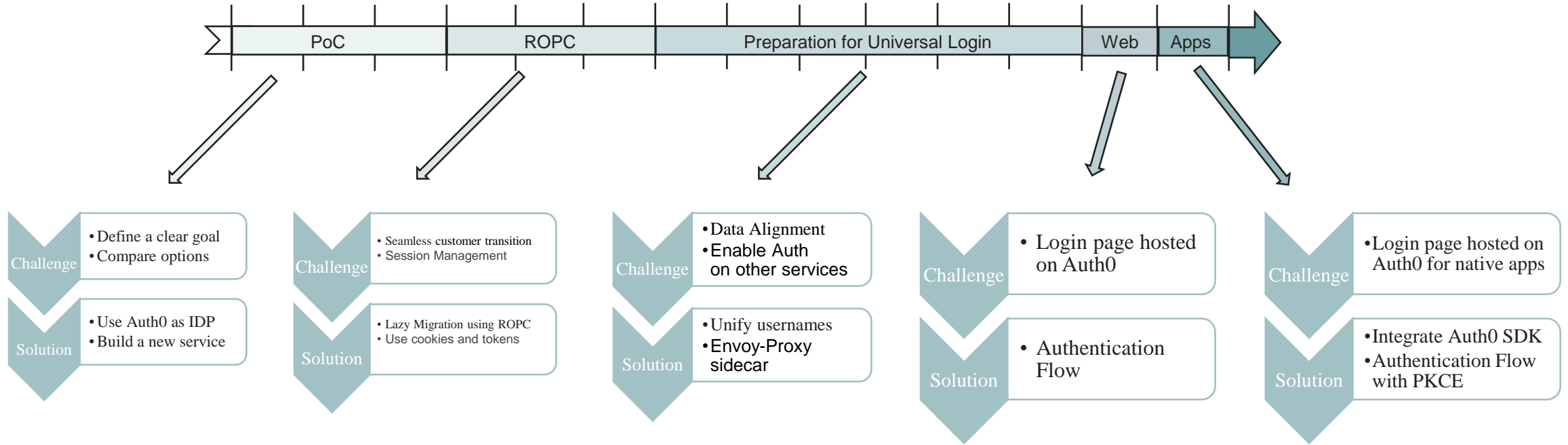
B E N E F I T S F O R E N D - U S E R S



BESTSECRET



# The end of our journey





# Benefits for end-users

## Seamless Integration:

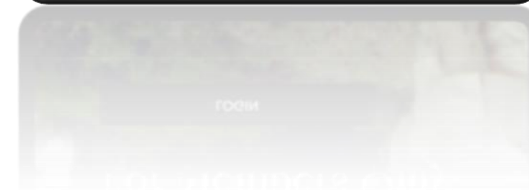
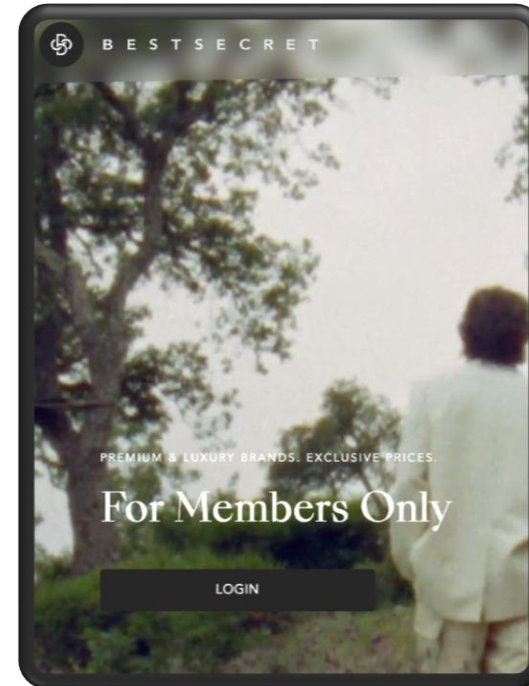
- Lazy Migration in the Background

## Security Features:

- Bot Protection
- Suspicious IP Throttling
- Brute-Force Detection
- Breached Password Protection

## New Options for Authentication possible:

- One-time passwords via email or SMS
- Biometric Authentication
- Multi-Factor Authentication
- Social Login



## Key takeaways

**CONCENTRATE** on your **CORE DOMAIN**: Leverage experts in the field to handle authentication, so you can focus on what you do best

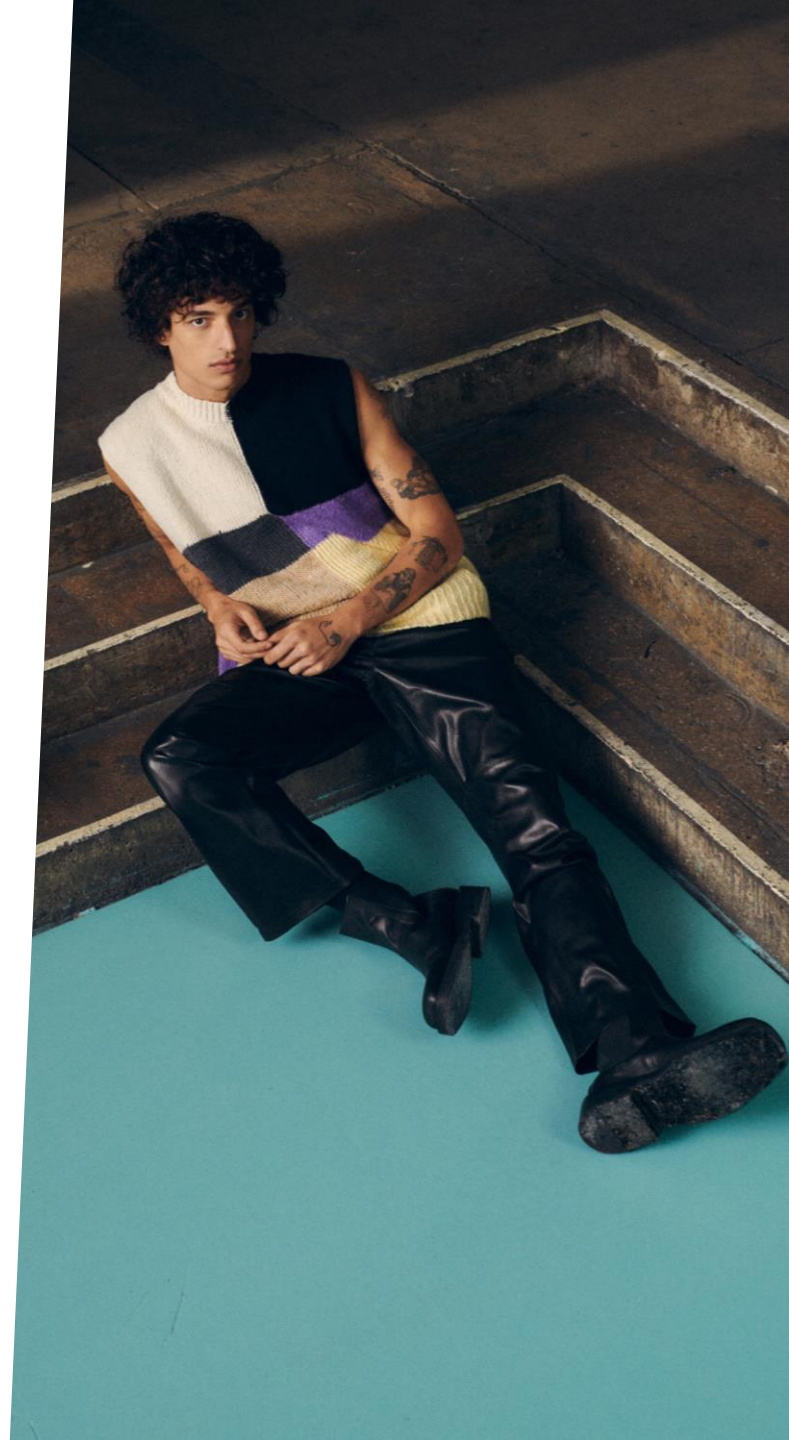
---

There are **SEVERAL OAUTH FLOWS**: Choose the one that aligns with your workflow and use case

---

**AVOID BIG-BANG tasks**: learn from small steps, always have a plan back, never take a step that *needs* to work

---





B E S T S E C R E T

Be part of  
something special.

T H A N K Y O U