# EXPO AND BETTER-AUTH

Auth That Does’t suck

# Auth Components

- Passwords
- 2FA
- Oauth (Google Sign In, Apple Sign In)
- Sessions and Cookies
- Roles and Permissions
- Passkeys
- Rate Limiting

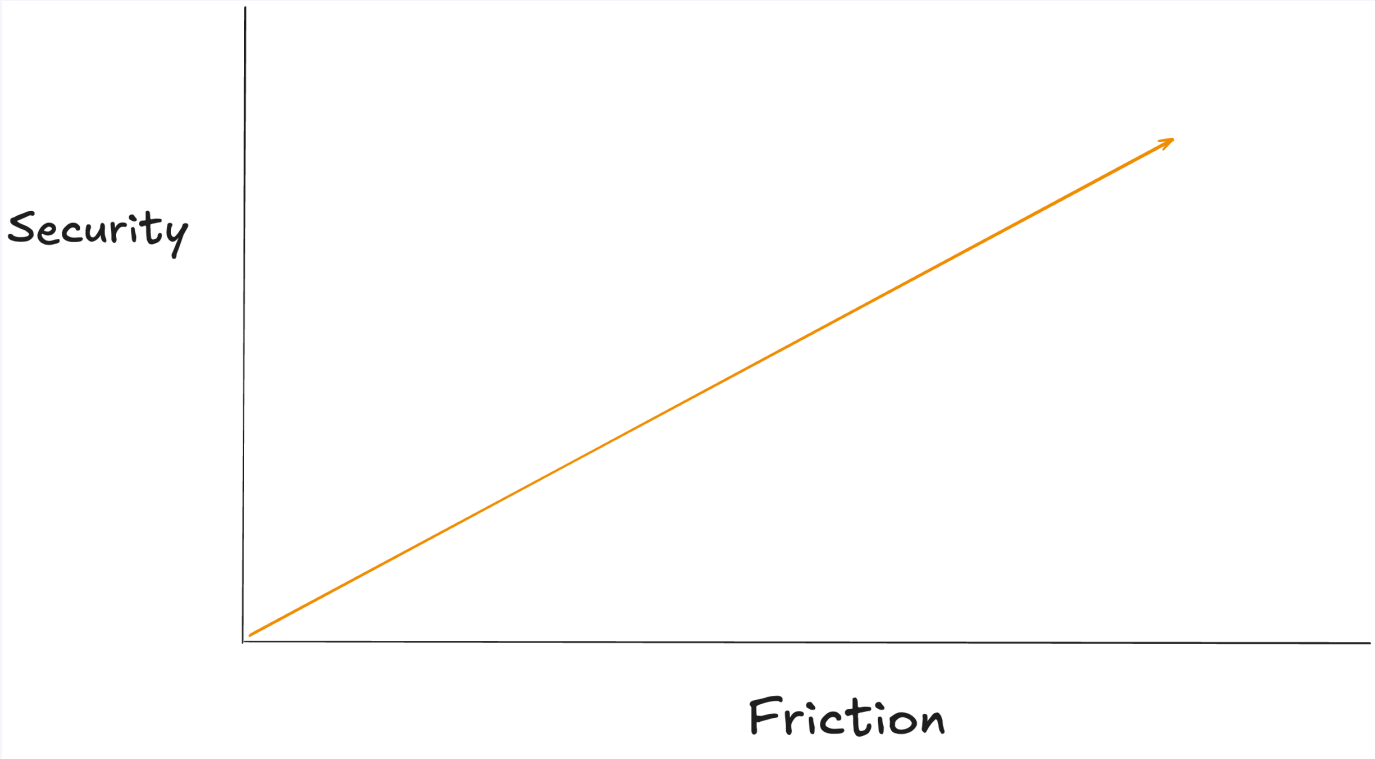
# Auth Companies

- Clerk
- Work OS
- Auth0 e.t.c

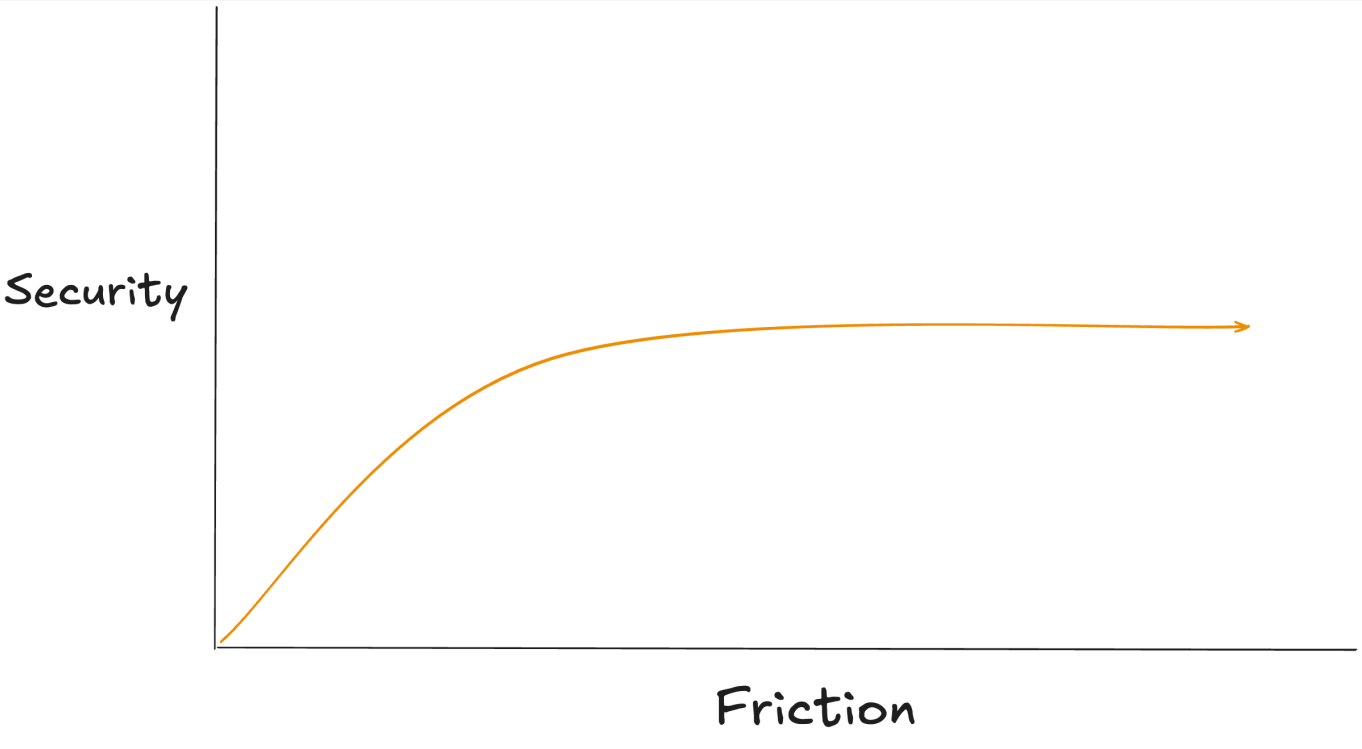
# In Conclusion

**Auth Sucks!!!**

# Friction vs Security



# Friction vs Security



# Auth

Auth should be the following:

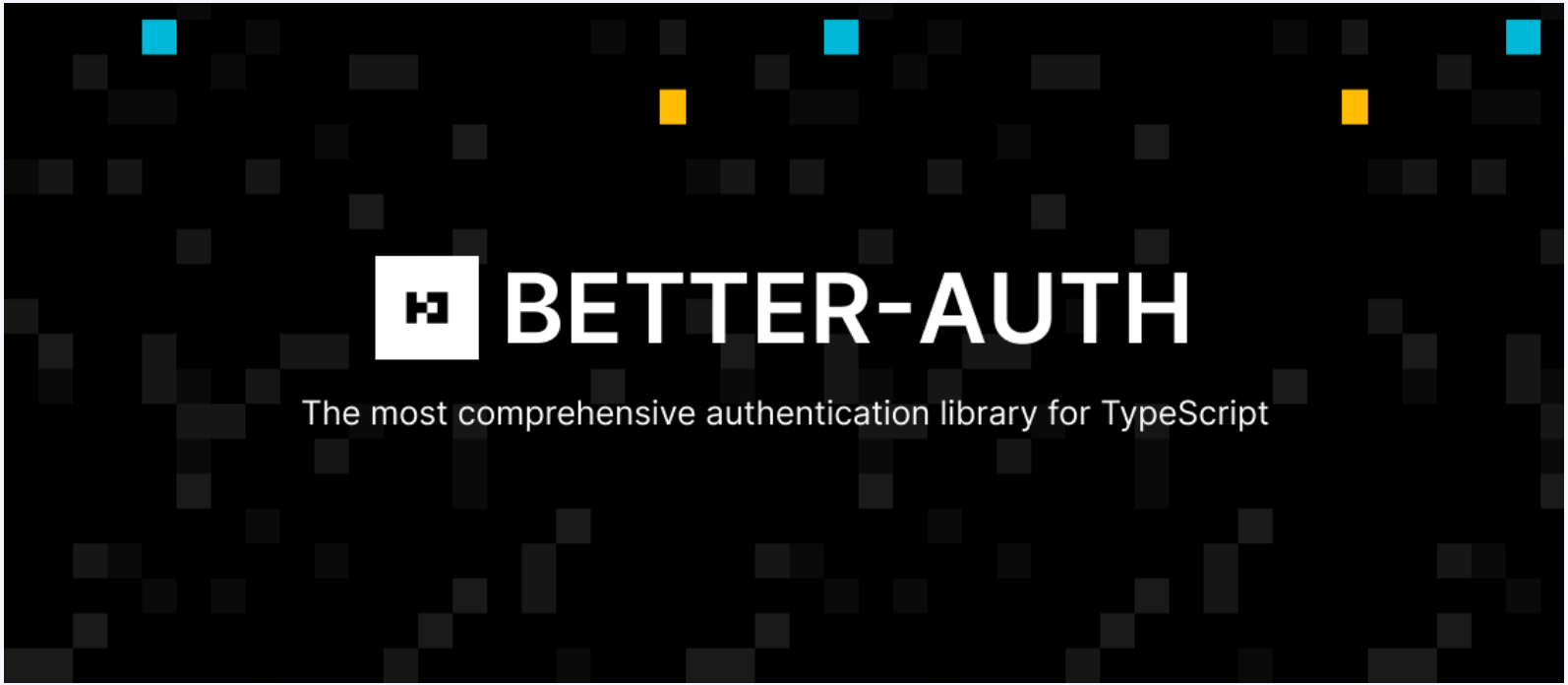
- **Easy** for the developer and the user and not solving a Rubiks cube.
- **Clever and dumb:** clever enough to spot a fake ID, dumb enough not to ask for your grandmother's blood type.
- **Invisible:** You barely notice it when it works, but it protects you constantly.
- **Trustworthy:** Secure enough that you don't worry about your data being stolen and not vibe-coded.

# We have a solution

“I’m going to implement my own authentication”

```
}

export default function makeLoginemployee () {
  return async (employeeInfo: LoginEmployeeInfo) => {
    const loggedInState: ILoggedInState = { status: false, user: null };
    const { username, password } = employeeInfo;

    if (!username || !password) {
      await slowDown(3500);
      return loggedInState;
    };

    const employee: any = await employeeRepository.findEmployeeByUsername(username);

    if (!employee) {
      await slowDown(3500);
      return loggedInState;
    };

    console.log(employeeInfo.deviceIp, employee.device_id);

    const savedIPAddress = employee.device_id.toString().split('.').join('');
    const currentIPAddress = employeeInfo.deviceIp.toString().replace('::ffff:',
'').split('.').join('');

    if (((savedIPAddress !== currentIPAddress) && (employeeInfo.deviceIp !== '::1')) && username !==
'test1@test.com') {
      await slowDown(3500);
      return loggedInState;
    }

    const hashedPassword = employee.password;
    const passwordMatch = await comparePasswords(password, hashedPassword);

    if (!passwordMatch) {
      await slowDown(3500);
```

# We have a solution



# Benefits

- Easy for the team to integrate (within minutes)
- Plugins for React and Expo React Native devs
- Open Source
- Data is self-hosted. You get to keep your data in your database. No third party integration needed.

# We have a solution

Demo with an Expo Application!!!

```ts
import { API_BASE_URL } from "@/src/constants";
import { expoClient } from "@better-auth/expo/client";
import { emailOTPClient, lastLoginMethodClient, multiSessionClient, phoneNumberClient, twoFactorClient } from "better-auth/client/plugins";
import { createAuthClient } from "better-auth/react";
import { expoPasskeyClient } from "expo-passkey/native";
import * as SecureStore from "expo-secure-store";

export const authClient = createAuthClient({
    baseURL: API_BASE_URL, // Base URL of your Better Auth backend.
    plugins: [
        expoClient({
            scheme: "rendercondemo2025",
            storagePrefix: "rendercondemo2025",
            storage: SecureStore,
        }),
        expoPasskeyClient({
            storagePrefix: "rendercondemo2025",
        }),
        twoFactorClient(),
        phoneNumberClient(),
        emailOTPClient(),
        lastLoginMethodClient(),
        multiSessionClient(),
    ]
});

export const {
    registerPasskey,
    authenticateWithPasskey,
    listPasskeys,
    revokePasskey,
    isPasskeySupported
  } = authClient;
```

```tsx
import { authClient } from "@/src/lib/auth-client";
import { useState } from "react";
import { Button, TextInput, View } from "react-native";
export default function SignIn() {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const handleLogin = async () => {
        await authClient.signIn.email({
            email,
            password,
        })
    };
    return (
        <View>
            <TextInput
                placeholder="Email"
                value={email}
                onChangeText={setEmail}
            />
            <TextInput
                placeholder="Password"
                value={password}
                onChangeText={setPassword}
            />
            <Button title="Login" onPress={handleLogin} />
        </View>
    );
}
```

```tsx
import { Button } from "react-native";
import { authClient } from "./auth-client";
export default function SocialSignIn() {
    const handleLogin = async () => {
        await authClient.signIn.social({
            provider: "google", // only google, apple and facebook are supported for idToken signIn
            idToken: {
                token: "...", // ID token from provider
                nonce: "...", // nonce from provider (optional)
            },
            callbackURL: "/dashboard" // this will be converted to a deep link (eg. `myapp://dashboard`) on native
        })
    };
    return <Button title="Login with Google" onPress={handleLogin} />;
}
```

# Who I am

**Timothy Mugo**

Full-stack developer

Building solutions

@timothygachengo

timothygachengo

Github Link: https://github.com/timothygachengo/rendercon-demo-2025