Professionele Bachelor Toegepaste Informatica

MULTIFACTORAUTHENTICATIE IN REACT

JAN VINKENROYE

Promotoren:

Jeroen Moors	Level27 BV
Luc Doumen	Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2021-2022

Professionele Bachelor Toegepaste Informatica

MULTIFACTORAUTHENTICATIE IN REACT

JAN VINKENROYE

Promotoren:

Jeroen Moors	Level27 BV
Luc Doumen	Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2021-2022

DANKWOORD

Na twaalf weken hard werken zit mijn stageproject erop. Ik heb in die twaalf weken veel geleerd en ik ben best wel trots op het resultaat.

Dit resultaat zou echter onmogelijk geweest zijn zonder de hulp en steun van een aantal personen.

Allereerst wil ik mijn bedrijfspromotor Jeroen Moors en mijn hogeschoolpromotor Luc Doumen bedanken voor alle ondersteuning, bijsturing en hulp.

Verder wil ik ook al de collega's bij Level27 bedanken. Ik wil in het bijzonder Joran Dirkzwager en Roald Lenaerts bedanken voor de fijne samenwerking, de inspirerende ideeën en de goede begeleiding. Alle andere collega's bedank ik voor de goede ontvangst en de uitstekende werksfeer.

Tenslotte wil ik mijn vriendin Sunisa bedanken voor de steun tijdens mijn stageperiode en meer algemeen tijdens de hele bacheloropleiding. Zij stond altijd klaar voor mij en sprak mij moed in op momenten dat ik het wat minder zag zitten.

ABSTRACT

Level27 BV is een Limburgse hostingprovider die klanten bijstaat bij het gebruiken van complexe technologieën.

Level27 heeft een eigen controlepaneel ontwikkeld van waaruit klanten hun domeinnamen, gedeelde hosting, servers en clusters kunnen beheren. De backend van dit controlepaneel is ontwikkeld met behulp van Symfony. De frontend is een *single page application* gemaakt met behulp van React.

De bachelorproef spitst zich toe op het frontendgedeelte van het controlepaneel. Ze bestaat uit een onderzoeksopdracht en een stageopdracht.

De onderzoeksopdracht onderzoekt op welke manier multifactorauthenticatie het best kan worden geïmplementeerd. De verschillende mogelijke verificatiemethodes worden vergeleken op het vlak van veiligheid, gebruiksgemak en implementatie.

De stageopdracht bestaat op zijn beurt uit twee delen.

Het eerste deel omvat de inbouw van multifactorauthenticatie in het controlepaneel.

Het tweede deel bestaat erin om een aantal diverse kleine features, verbeteringen en bugreparaties te implementeren.

INHOUDSOPGAVE

LIJST VAN GEBRUIKTE FIGUREN

LIJST VAN GEBRUIKTE TABELLEN

LIJST VAN GEBRUIKTE AFKORTINGEN

api	Application programming interface
CSS	Cascading Style Sheets
CTAP	Client to Authenticator Protocol
DOM	Document Object Model
FIDO	Fast Identity Online
HMAC	Hash-Based Message Authentication Codes
HOTP	HMAC-based One-Time Password
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDaaS	Identity as a Service
IP	Internet Protocol
IT	Information technology
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript XML
JWT	JSON Web Token
NFC	Near-field communication
OATH	Initiative for Open Authentication
OAuth	Open Authorization
OCRA	OATH Challenge-Response Algorithm
OTP	One-Time Password
PKI	Public key infrastructure
QR	Quick Response
RSA	Rivest–Shamir–Adleman
SAML	Security Assertion Markup Language
sms	Short message service
SSO	Single sign-on
TAN	Transaction authentication number
TOTP	Time-based One-Time Password
U2F	Universal 2nd Factor
UAF	Universal Authentication Framework
URL	Uniform Resource Locator
USB	Universal Serial Bus
W3C	World Wide Web Consortium
WebAuthn	Web Authentication
XML	Extensible Markup Language

INLEIDING

Dit eindwerk maakt deel uit van de bachelorproef van de bachelor toegepaste informatica aan Hogeschool PXL.

Het eerste deel van het eindwerk is een verslag van de stage van 12 weken die ik in het kader van deze bachelorproef doorlopen heb. De stage vond plaats bij Level27 in Hasselt. De stageopdracht was om te bekijken op welke manier tweefactorauthenticatie kon ingebouwd worden in het controlepaneel van Level27. Daarnaast was het de bedoeling om een aantal kleinere features, verbeteringen en bugreparaties te implementeren in de frontend van het controlepaneel.

Het tweede deel bevat de onderzoeksopdracht. In de onderzoeksopdracht heb ik proberen een overzicht te geven van het huidige authenticatielandschap. Ik heb ook de verschillende verificatiemethoden onderzocht die gebruikt kunnen worden als tweede factor.

I. STAGEVERSLAG

1 BEDRIJFSVOORSTELLING

Figuur 1: Het logo van Level27

Level27 is een Limburgse hostingprovider die klanten ondersteunt om complexe technologieën te gebruiken.

De diensten die Level27 aanbiedt vallen grofweg onder te verdelen in drie categorieën.

Allereerst is er de **webhosting voor websites en applicaties**. Daartoe biedt Level27 een controlepaneel aan om domeinnamen, *shared hosting*, servers en clusters te beheren. Level27 streeft daarbij naar innovatie door gebruik te maken van een zelf ontwikkeld controlepaneel en door vernieuwende technologieën zoals Docker en Kubernetes te implementeren.

Daarnaast voert Level27 ook **technologieprojecten** uit voor *agencies*, start-ups en developer-georiënteerde organisaties. Level 27 biedt zijn *operations*ervaring aan om samen met de klant een *proof of concept* op te zetten.

Tenslotte kan Level27 zorgen voor het volledige **beheer van publieke en private clouddiensten** van een klant in samenwerking met de lokale IT-partner.

De bedrijfscultuur van Level27 vertrekt vanuit een passie voor technologie en esthetiek. Door complexe technologie dagelijks bruikbaar te maken, wil Level27 haar klanten helpen groeien.

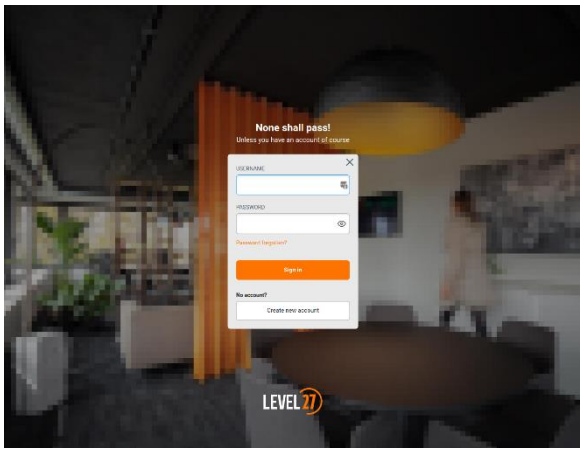
Level27 heeft vier kernwaarden:

- Kwaliteit en consistentie
- Betrokken
- No-bullshit
- Klantgericht

2 VOORSTELLING STAGEOPDRACHT

Level27 heeft een controlepaneel ontwikkeld van waaruit klanten hun account en hun systemen, applicaties, mailgroepen en domeinen kunnen beheren. Dit controlepaneel is ontwikkeld met behulp van Symfony en React.js. De stageopdracht spitst zich toe op het frontendgedeelte van dit controlepaneel.

Het controlepaneel maakt momenteel gebruik authenticatie met behulp van een wachtwoord.



Figuur 2: Het inlogscherm van het controlepaneel

Om de veiligheid te verhogen, wil Level27 een extra authenticatiefactor aan het authenticatiesysteem toevoegen. Het eerste deel van de stageopdracht omvat de implementatie hiervan.

Het tweede deel van de stageopdracht omvat een aantal kleinere taken in het controlepaneel. Het gaat om diverse bugreparaties, verbeteringen en kleinere features.

3 UITWERKING STAGEOPDRACHT: TWEEFACTORAUTHENTICATIE

3.1 IDENTITY AS A SERVICE (IDAAS) OF EIGEN ONTWIKKELING

De eerste stap was de keuze om al dan niet een cloudoplossing te gebruiken.

Platformen als Auth0, Okta, AWS Cognito, Microsoft Azure AD en Firebase Auth laten toe om het hele identiteitsbeheer, daarbij inbegrepen het authenticatieproces, uit te besteden.

Onderstaande tabel geeft een overzicht van de belangrijkste voor- en nadelen van het gebruik van deze platformen ten opzichte van het zelf ontwikkelen van tweefactorauthenticatie.

TABEL 1: VOOR- EN NADELEN VAN IDAAS-PLATFORMEN EN EIGEN ONTWIKKELING

	EIGEN ONTWIKKELING	IDENTITY AS A SERVICE (IDAAS)
VOORDELEN	▪ Het beheer van de gebruikersdata blijft in eigen handen. ▪ Onafhankelijkheid van een externe partij.	▪ Steun op ervaren professionals. ▪ Minder werk om op te zetten. ▪ Makkelijk uitbreidbaar met andere *identity providers*, authenticatiemethoden en andere integraties. ▪ De flows zijn vrij eenvoudig.
NADELEN	▪ Complexe flows. ▪ De backendcode moet onderhouden en aangepast worden. ▪ Elke kleine fout vormt een veiligheidsrisico. ▪ Het is vereist om: - hetzij te steunen op externe bibliotheken waarvan de documentatie vaak ondermaats is en waarvan de *security patches* nauwgezet in het oog moeten gehouden worden, - hetzij helemaal alles zelf te ontwikkelen (meer werk en risico op fouten).	▪ Grote afhankelijkheid van externe provider voor cruciale bedrijfsprocessen. ▪ Groot vertrouwen nodig in externe provider. ▪ *Vendor lock-in.* Dit kan worden beperkt door een provider te kiezen die standaarden als OAuth2/OIDC volgt. ▪ Migratie van de gebruikersdata naar de IDaaS-provider. ▪ Vanaf een bepaald aantal gebruikers is er een prijs aan verbonden.

Uiteindelijk werd gekozen om alles zelf te ontwikkelen. Level27 vond het belangrijk om alles in eigen handen te kunnen houden. De vereisten van Level27 op het vlak van authenticatie waren bovendien relatief eenvoudig, waardoor de meerwaarde van een IDaaS-provider eerder beperkt bleef.

3.2 De verificatiemethode van de tweede factor

De tweede stap was de keuze van de methode die gebruikt zou worden als tweede factor. In de onderzoeksopdracht (zie het tweede deel van dit eindwerk) worden de voor- en nadelen van diverse mogelijke methoden onderzocht. Uiteindelijk werd er gekozen om een *time-based one-time password* (TOTP) te gebruiken.

3.3 Welke gebruikers

Het was van bij het begin af aan duidelijk dat tweefactorauthenticatie niet zal opgelegd worden aan alle gebruikers. Elke gebruiker moet zelf de keuze hebben om tweefactorauthenticatie al dan niet in te stellen. Op termijn is het wel de bedoeling om dit afdwingbaar te maken op organisatieniveau, maar dat valt buiten de scope van deze stageopdracht.

3.4 Frequentie van de tweede factor

Omwille van de gebruiksvriendelijkheid zal de tweede factor niet systematisch gevraagd worden bij elke login met een wachtwoord. Wanneer de klant inlogt met een *one-time password* kan hij een optie aanvinken om de tweede factor slechts na 14 dagen opnieuw te vragen. Voor deze functionaliteit wordt gebruik gemaakt van een permanente cookie.

3.5 Lost device policy

Gebruikers die hun toestel verliezen kunnen zich mogelijk niet meer authenticeren. Er bestaan diverse mogelijke manieren om hiermee om te gaan:

- De gebruiker kan gevraagd worden het gedeeld geheim op te slaan zodat hij later een ander toestel kan instellen om dezelfde TOTP's te genereren. Dat kan in de vorm van een schermopname van de QR-code of in de vorm van de tekst zelf van het gedeeld geheim.
- Er kan gebruik gemaakt worden van recoverycodes. Dit zijn codes die eenmalig als tweede factor gebruikt kunnen worden.
- De gebruiker kan gevraagd worden om een back-uptoestel te registreren. Bij verlies van het hoofdtoestel wordt dan een verificatiecode naar dit back-uptoestel gestuurd, bijvoorbeeld via sms.
- De gebruiker kan gevraagd worden om zelf een tweede toestel als back-up in te stellen.
- De mogelijkheid kan voorzien worden om tweefactorauthenticatie via e-mail uit te schakelen. Er wordt dan een e-mail gestuurd naar het geregistreerde e-mailadres van de gebruiker. De gebruiker kan op een link in de e-mail klikken om tweefactorauthenticatie uit te schakelen.

Maar ook deze manieren zijn niet waterdicht. Gebruikers kunnen hun recoverycodes of back-uptoestellen verliezen, geen toegang meer hebben tot hun e-mailadres … Het laatste redmiddel blijft altijd het contact met de klantendienst om de volledige account te resetten. Er werd daarom gekozen om geen enkele van bovengenoemde methoden op te nemen in de scope van de stageopdracht.

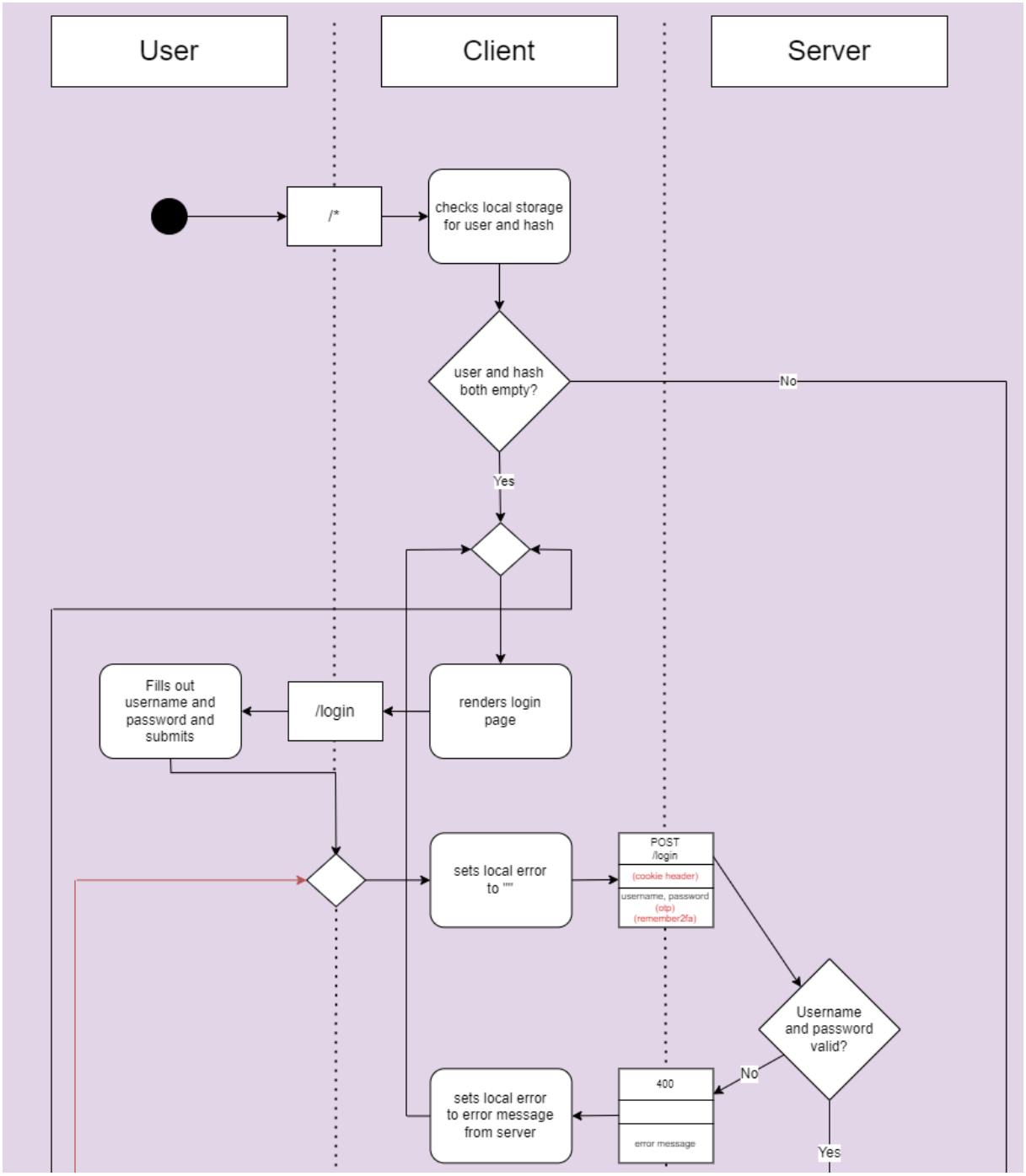
3.6 DE FLOWS

Momenteel wordt er in het controlepaneel gebruik gemaakt van *token based authentication*. De gebruiker logt in met zijn gebruikersnaam en wachtwoord en ontvangt dan een token van de server. De client slaat het token op in de *local storage* van de browser. Bij elke *request* wordt het token meegestuurd in de *authorization header*. De server beslist op basis van het token in de header om de client al dan niet toegang te geven tot de gevraagde gegevens. Het token blijft een aantal dagen geldig. Wanneer de expiratiedatum van het token verstreken is, moet de gebruiker opnieuw inloggen met zijn gebruikersnaam en wachtwoord om een nieuw token te ontvangen.

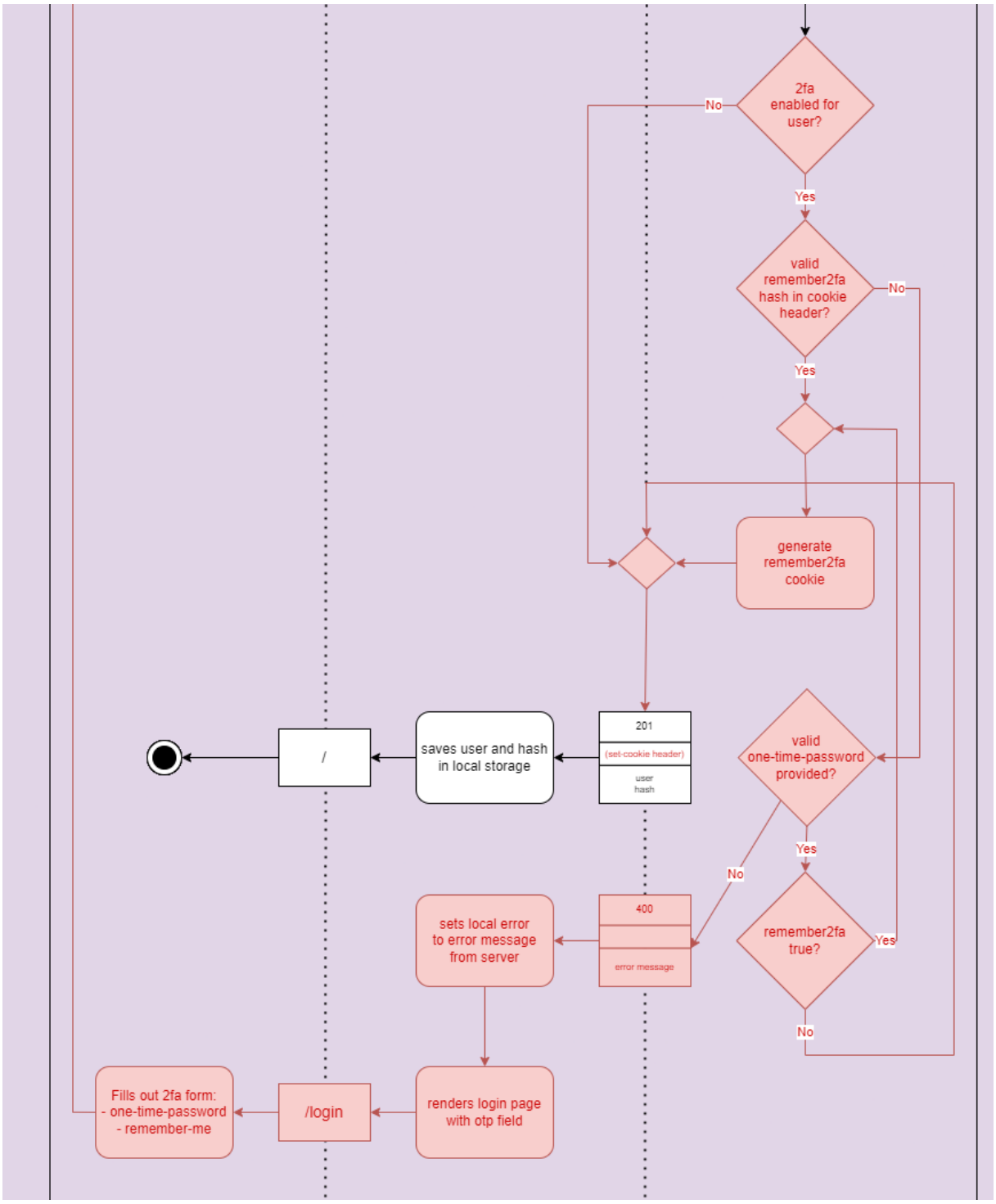
Deze authenticatieflow wordt aangevuld om tweefactorauthenticatie mogelijk te maken. Nadat de server de gebruikersnaam en het wachtwoord gevalideerd heeft, gaat de server na of de gebruiker tweefactorauthenticatie ingeschakeld heeft. Wanneer tweefactorauthenticatie uitgeschakeld is, blijft alles bij het oude en wordt een *response* met een *authorization hash* en de gebruikersgegevens teruggegeven.

Wanneer tweefactorauthenticatie wel ingeschakeld is, probeert de server de tweede factor te valideren. Is er een geldige *remember2fa hash* aanwezig in de *cookie header*, dan is de tweede factor gevalideerd en kunnen de *authorization hash* en de gebruikersgegevens worden teruggegeven. Er wordt ook een nieuwe *remember2fa cookie* aangemaakt die in een *set-cookie header* wordt meegegeven. Is er geen geldige hash, dan controleert de server of de tweede factor gevalideerd kan worden aan de hand van een geldig TOTP. Indien een geldig TOTP in de *request* aanwezig is wordt de response met de *authorization hash* en de gebruikersgegevens teruggegeven. Wanneer het *key-value*paar ‘”remember2fa”: true’ aanwezig is wordt er een *remember2fa cookie* aangemaakt en teruggegeven.

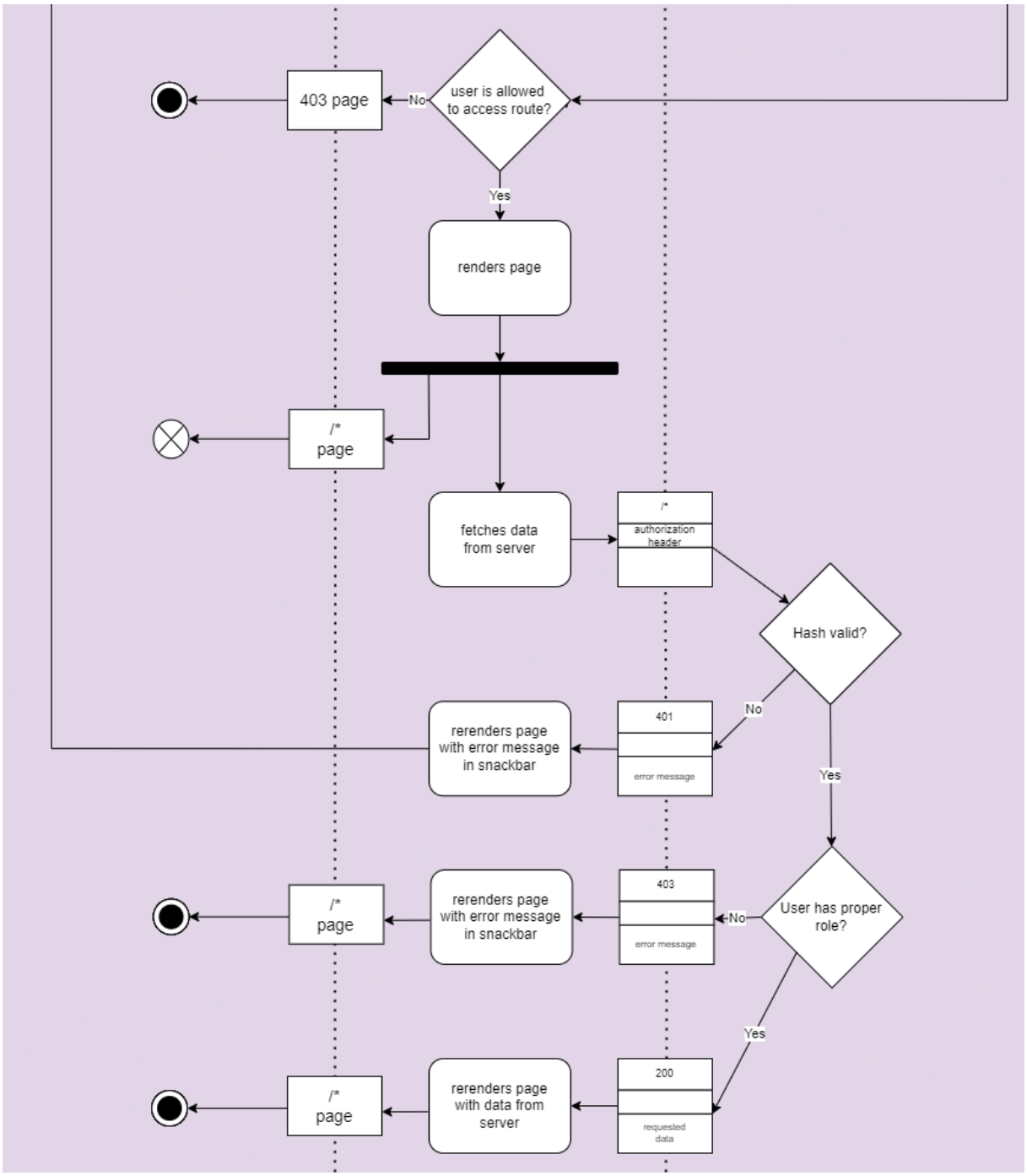
Wanneer er noch een geldige remember2fa hash, noch een geldig TOTP meegegeven werd, geeft de server een *response* met statuscode 400 en een *error message* terug. De client vernieuwt dan de loginpagina. Wanneer er een ongeldig TOTP werd meegegeven wordt de *error message* van de server getoond aan de gebruiker. De vernieuwde loginpagina biedt de gebruiker de mogelijkheid om een (nieuw) TOTP in te geven en desgewenst het vakje om de tweede factor voor 14 dagen te onthouden aan te vinken. De client geeft deze gegevens samen met de gebruikersnaam en het wachtwoord door aan de server. De client doet daartoe een nieuwe *request* naar hetzelfde *login endpoint*.



Figuur 3: 2fa access flow (deel 1)



Figuur 4: 2fa access flow (deel 2)



Figuur 5: 2fa access flow (deel 3)

3.7 DE ENDPOINTS

3.7.1 ORGANISATIONS/:ORGANISATIONID/USERS/:USERID/2FA/GENERATE

3.7.1.1 Request method

Post.

3.7.1.2 Omschrijving

Endpoint om de URL van de QR-code om tweefactorauthenticatie in te stellen op te vragen.

3.7.1.3 Wat doet de backend?

De server genereert een gedeeld geheim voor de geauthenticeerde gebruiker en slaat het geheim op. De server maakt een QR-code aan die het gedeeld geheim, de configuratie, de e-mail en de *issuer* bevat en geeft de QR-code terug aan de client in de vorm van een URL.

3.7.1.4 Request headers

Een *authorization header* met een geldig autorisatietoken is vereist.

3.7.1.5 Request body

Er is geen body vereist.

3.7.1.6 Responses

HTTP-statuscode 201

Wanneer de *request* succesvol verwerkt wordt, wordt een *response* met statuscode 201 met de URL van de QR-code teruggegeven.

```
{
    "otpauthUrl":
"otpauth://totp/Level27:jan%40gmail.com?secret=HJGRKDR3NJMFEZSG&per
iod=30&digits=6&algorithm=SHA1&issuer=Level27"
}
```

HTTP-statuscode 401

Wanneer het token in de *authorization header* ongeldig is of ontbreekt, wordt een *response* met statuscode 401 teruggegeven.

```
{
    "statusCode": 401,
    "message": "Unauthorized"
}
```

3.7.2 ORGANISATIONS/:ORGANISATIONID/USERS/:USERID/2FA/ENABLE

3.7.2.1 Request method

Post.

3.7.2.2 Omschrijving

Endpoint om tweefactorauthenticatie in te schakelen voor de gebruiker.

3.7.2.3 Wat doet de backend?

De server verifieert of tweefactorauthenticatie al ingeschakeld is voor de geauthenticeerde gebruiker. Wanneer dit nog niet het geval is, gaat de server na of er een gedeeld geheim bestaat in de database voor de gebruiker. Is dat het geval, dan controleert de server of er een geldig TOTP meegestuurd is. Zo ja, dan wordt tweefactorauthenticatie ingeschakeld door de *property* 'has2faEnabled' op 'true' te zetten.

3.7.2.4 Request headers

Een *autorization header* met een geldig autorisatietoken is vereist.

3.7.2.5 Request body

```
{
  "otp": "175816"
}
```

3.7.2.6 Responses

HTTP-statuscode 200

Wanneer de *request* succesvol verwerkt wordt, wordt een *response* met statuscode 200 met de gebruikersgegevens teruggegeven.

```
{
  "user": {
    "id": 123,
    "username": "testemail@testemail.com",
    "email": "testemail@testemail.com",
    "firstName": "Test",
    "lastName": "User",
    "roles": [
      "ROLE_SYSTEM",
      "ROLE_APP",
      "ROLE_DOMAIN",
      "ROLE_MAIL"
    ],
    "status": "ok",
    "language": "en",
    "organisation": {
      "id": 123,
      "name": "Test Company",
      "street": "Test-Street",
      "houseNumber": "23",
      "zip": "1000",
```

```
      "city": "Test City",
      "reseller": null
    },
    "country": {
      "id": "BE",
      "name": "Belgium"
    },
    "fullname": "Test User",
    "teams": [
      {
        "id": 123,
        "name": "MyTeam",
        "autoAddSshkey": false
      }
    ],
     "has2faEnabled": true
  }
}
```

HTTP-statuscode 200

Wanneer de tweefactorauthenticatie al ingeschakeld is, wordt een *response* met statuscode 200 teruggegeven.

```
{
    "statusCode": 200,
    "message": "Two-factor authentication already enabled"
}
```

HTTP-statuscode 400

Wanneer geen *one-time password* wordt meegegeven, wordt een *response* met statuscode 400 teruggegeven.

```
{
    "statusCode": 400,
    "message": "One-time password not provided"
}
```

HTTP-statuscode 400

Wanneer een ongeldig *one-time password* wordt meegegeven, wordt een *response* met statuscode 400 teruggegeven.

```
{
    "statusCode": 400,
    "message": "One-time password not valid"
}
```

HTTP-statuscode 400

Wanneer geen gedeeld geheim teruggevonden wordt in de database, wordt een *response* met statuscode 400 teruggegeven.

```
{
    "statusCode": 400,
    "message": "No mfa secret found on the server"
}
```

HTTP-statuscode 401

Wanneer het token in de *authorization header* ongeldig is of ontbreekt, wordt een *response* met statuscode 401 teruggegeven.

```
{
    "statusCode": 401,
    "message": "Unauthorized"
}
```

3.7.3 ORGANISATIONS/:ORGANISATIONID/USERS/:USERID/2FA/DISABLE

3.7.3.1 Request method

Post.

3.7.3.2 Omschrijving

Endpoint om tweefactorauthenticatie uit te schakelen voor de gebruiker.

3.7.3.3 Wat doet de backend?

De server verifieert in eerste instantie of tweefactorauthenticatie al uitgeschakeld is voor de geauthenticeerde gebruiker. Is dat het geval, dan controleert de server of er een geldig TOTP meegestuurd is. Zo ja, dan wordt tweefactorauthenticatie uitgeschakeld door de *property* 'has2faEnabled' op 'false' te zetten en wordt het gedeeld geheim verwijderd uit de database.

3.7.3.4 Request headers

Een *authorization header* met een geldig autorisatietoken is vereist.

3.7.3.5 Request body

```
{
  "otp": "175816"
}
```

3.7.3.6 Responses

HTTP-statuscode 200

Wanneer de *request* succesvol verwerkt wordt, worden een *response* met statuscode 200 met de gebruikersgegevens teruggegeven.

```
{
  "user": {
    "id": 123,
    "username": "testemail@testemail.com",
    "email": "testemail@testemail.com",
    "firstName": "Test",
    "lastName": "User",
    "roles": [
      "ROLE_SYSTEM",
      "ROLE_APP",
      "ROLE_DOMAIN",
      "ROLE_MAIL"
    ],
    "status": "ok",
    "language": "en",
    "organisation": {
      "id": 123,
      "name": "Test Company",
      "street": "Test-Street",
      "houseNumber": "23",
      "zip": "1000",
      "city": "Test City",
      "reseller": null
    },
    "country": {
      "id": "BE",
      "name": "Belgium"
    },
    "fullname": "Test User",
    "teams": [
      {
        "id": 123,
        "name": "MyTeam",
        "autoAddSshkey": false
      }
    ],
     "isTotpEnabled": false
  }
}
```


HTTP-statuscode 200

Wanneer de tweefactorauthenticatie al uitgeschakeld is, wordt een *response* met statuscode 200 teruggegeven.

```
{
    "statusCode": 200,
    "message": "Two-factor authentication already disabled"
}
```

HTTP-statuscode 400

Wanneer geen *one-time password* wordt meegegeven, wordt een *response* met statuscode 400 teruggegeven.

```
{
    "statusCode": 400,
    "message": "One-time password not provided"
}
```

HTTP-statuscode 400

Wanneer een ongeldig *one-time password* wordt meegegeven, wordt een *response* met statuscode 400 teruggegeven.

```
{
    "statusCode": 400,
    "message": "One-time password not valid"
}
```

HTTP-statuscode 401

Wanneer het token in de *authorization header* ongeldig is of ontbreekt, wordt een *response* met statuscode 401 teruggegeven.

```
{
    "statusCode": 401,
    "message": "Unauthorized"
}
```

3.7.4 LOGIN

3.7.4.1 Request method

Post.

3.7.4.2 Omschrijving

Reeds bestaand *endpoint* om in te loggen. Dit *endpoint* wordt gewijzigd.

3.7.4.3 Wat doet de backend?

De server controleert de gebruikersnaam en het wachtwoord. Wanneer de gebruikersnaam teruggevonden wordt en het wachtwoord correct is, gaat de server na of de gebruiker tweefactorauthenticatie ingeschakeld heeft. Is dat het geval, dan kijkt de server of er een geldige *remember2fa* hash teruggevonden wordt in de *cookie header*. Indien er een geldige *remember2fa* hash meegegeven is, wordt een nieuwe cookie aangemaakt en in een *set-cookie header*

teruggegeven. De nieuwe hash wordt opgeslagen in de database en de oude hash wordt verwijderd. Wanneer er geen geldige *remember2fa* hash meegegeven is, wordt nagegaan of er een geldig TOTP is meegestuurd in de body van de *request*. Zo ja, dan wordt er een *remember2fa* cookie aangemaakt en teruggegeven indien het *key-value*paar '"remember2fa": true' teruggevonden wordt in de body van de *request*.

3.7.4.4 Request headers

Er is geen *authorization header* vereist. Eventueel kan een *cookie header* met een *remember2fa* hash meegegeven worden.

3.7.4.5 Request body

```
{
  "username": "jan@gmail.com",
  "password": "mypass",
  "otp": "175816",
  "remember2fa": true
}
```

3.7.4.6 Responses

HTTP-statuscode 201

Wanneer de *request* succesvol verwerkt wordt, worden een *response* met statuscode 201 met de hash en de gebruikersgegevens teruggegeven.

```
{
"hash":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImphbkBnbWFpb
C5jb20iLCJzdWIiOjE0LCJpYXQiOjE2NTUxOTI0NDYsImV4cCI6MTY1ODc5MjQ0Nn0.
GQzx562KTUVefSULuS-iKQTOcA6HbE98mcfV_MH4_Xs",
  "user": {
    "id": 123,
    "username": "testemail@testemail.com",
    "email": "testemail@testemail.com",
    "firstName": "Test",
    "lastName": "User",
    "roles": [
      "ROLE_SYSTEM",
      "ROLE_APP",
      "ROLE_DOMAIN",
      "ROLE_MAIL"
    ],
    "status": "ok",
    "language": "en",
    "organisation": {
      "id": 123,
```

```
      "name": "Test Company",
      "street": "Test-Street",
      "houseNumber": "23",
      "zip": "1000",
      "city": "Test City",
      "reseller": null
    },
    "country": {
      "id": "BE",
      "name": "Belgium"
    },
    "fullname": "Test User",
    "teams": [
      {
        "id": 123,
        "name": "MyTeam",
        "autoAddSshkey": false
      }
    ],
     "has2faEnabled": false
  }
}
```


HTTP-statuscode 400

Wanneer tweefactorauthenticatie ingeschakeld is, de *cookie header* geen geldige *remember2fa* hash bevat en geen *one-time password* in de request body wordt meegegeven, wordt een *response* met statuscode 400 teruggegeven.


```
{
    "statusCode": 400,
    "message": "One-time password not provided"
}
```


HTTP-statuscode 400

Wanneer tweefactorauthenticatie ingeschakeld is, de *cookie header* geen geldige *remember2fa* hash bevat en een ongeldig *one-time password* wordt meegegeven, wordt een *response* met statuscode 400 teruggegeven.


```
{
    "statusCode": 400,
    "message": "One-time password not valid"
}
```

HTTP-statuscode 400

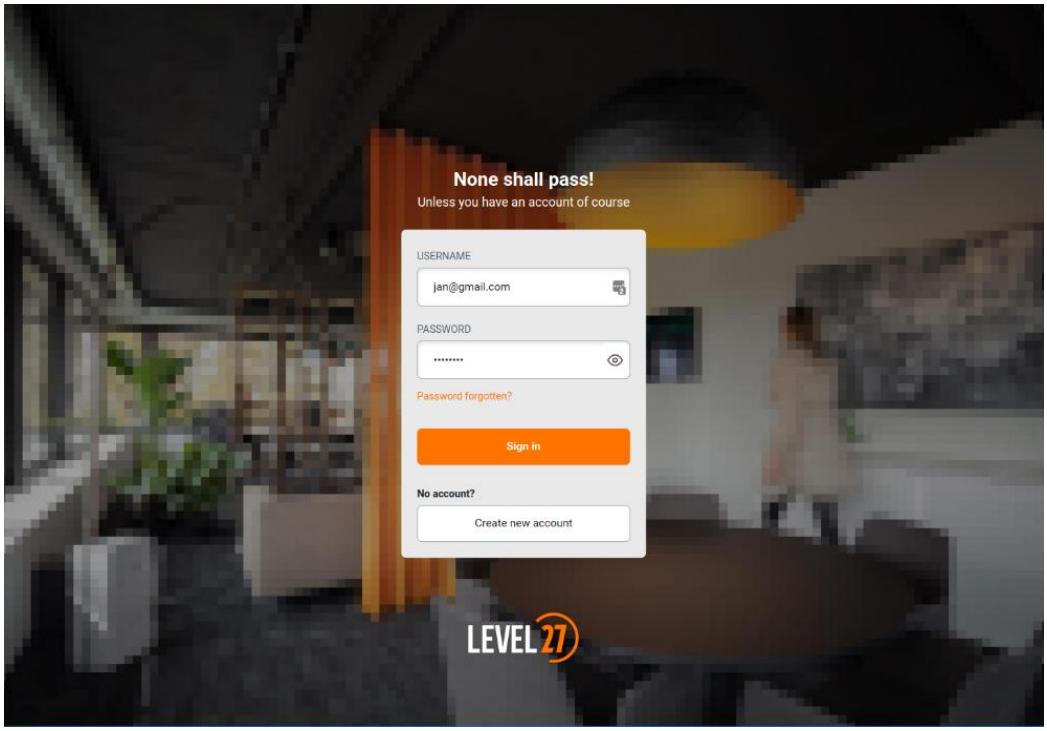
Wanneer de gebruikersnaam niet teruggevonden wordt in de database of het meegegeven wachtwoord ongeldig is, wordt een *response* met statuscode 400 teruggegeven.

```
{
    "statusCode": 400,
    "message": "Invalid credentials"
}
```

3.8 PROOF OF CONCEPT

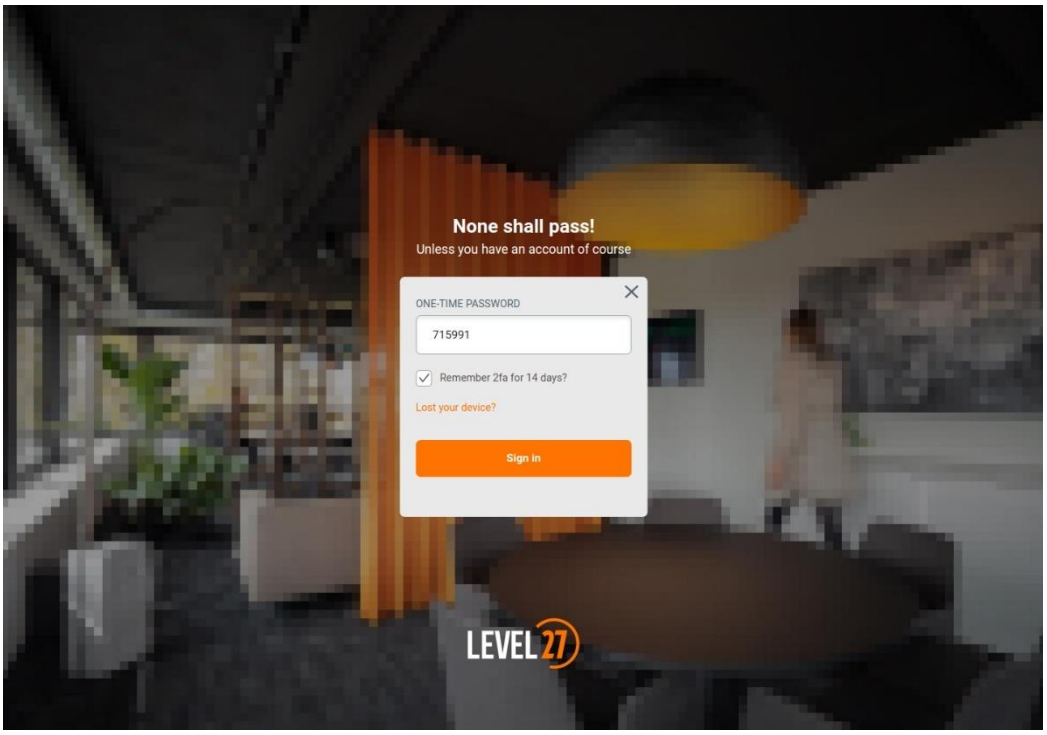
Het laatste onderdeel van het eerste deel van de stageopdracht omvat de eigenlijke code in de vorm van een *proof of concept*. Het frontendgedeelte werd geïmplementeerd in een aparte *branch,* afgesplitst van de *developmentbranch* van het controlepaneel van Level27. Voor het backendgedeelte werd een eigen implementatie gebruikt die gebruikt maakt van het nest.js-framework.

Het inlogscherm blijft aanvankelijk ongewijzigd. De gebruiker moet zijn gebruikersnaam en wachtwoord ingeven om toegang te krijgen tot de applicatie.



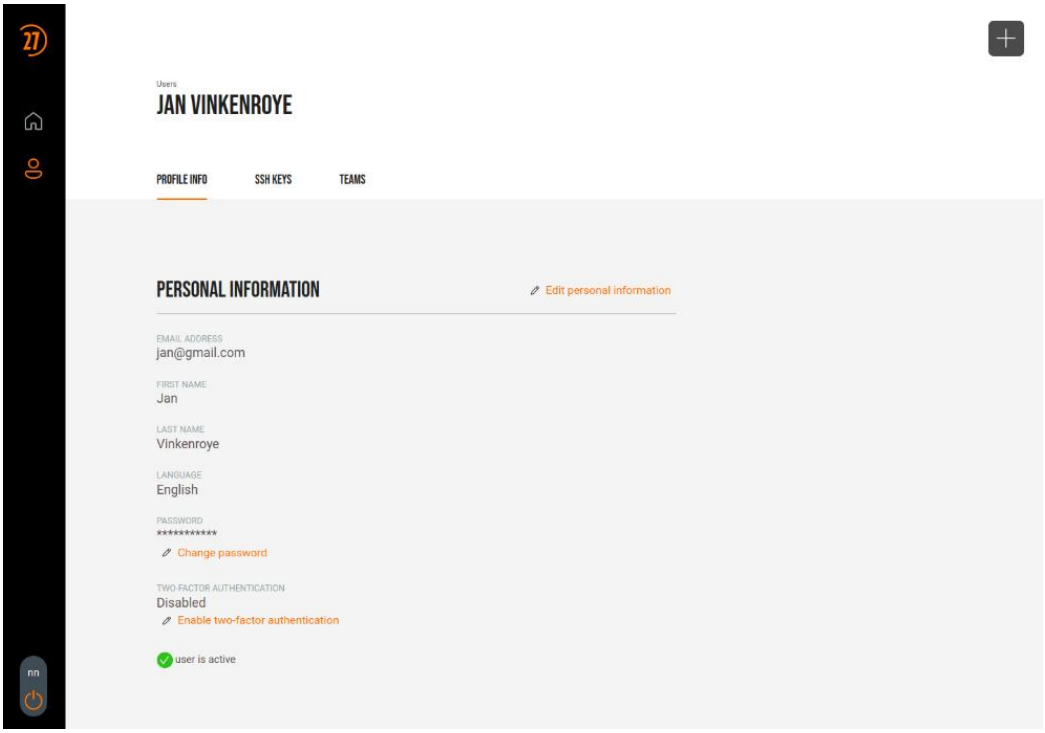
Figuur 6: Het inlogscherm met velden om een gebruikersnaam en een wachtwoord in te geven

Wanneer tweefactorauthenticatie ingeschakeld is voor de gebruiker, dient de gebruiker vervolgens zijn TOTP in te geven. De gebruiker heeft ook de mogelijkheid om de optie aan te vinken om de tweede factor voor 14 dagen bij te houden.



Figuur 7: Het inlogscherm met een OTP-veld

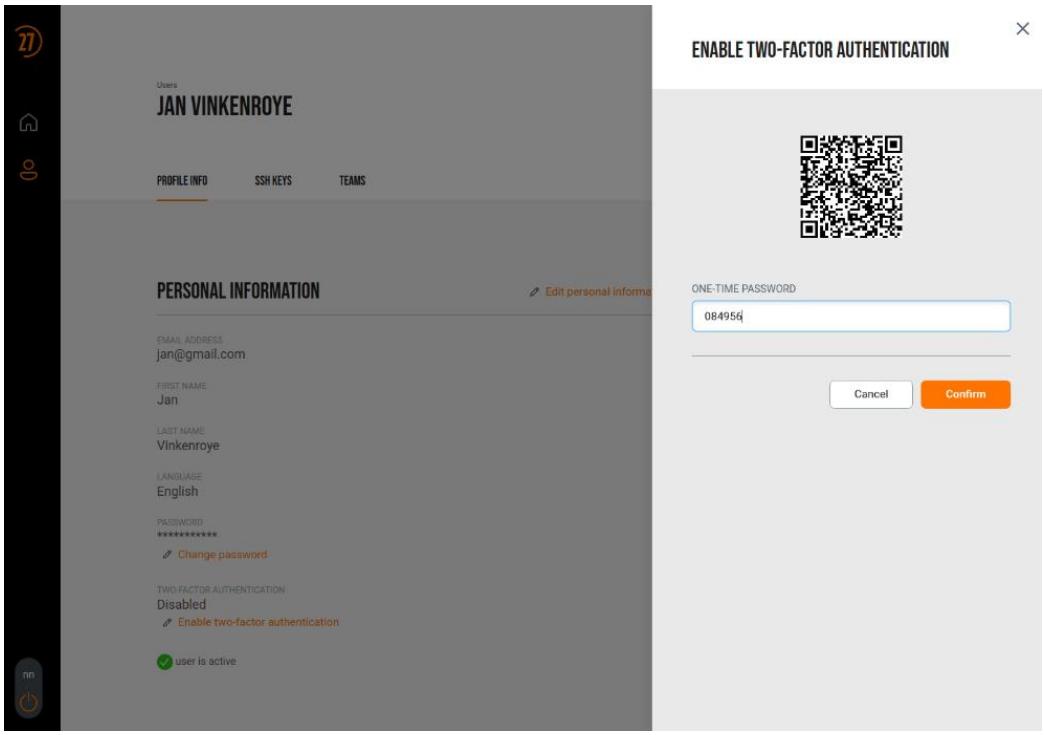
In de applicatie is op het profiel van de gebruiker een extra veld bijgekomen dat de status van de tweefactorauthenticatie aangeeft.



Figuur 8: Het profiel van de gebruiker

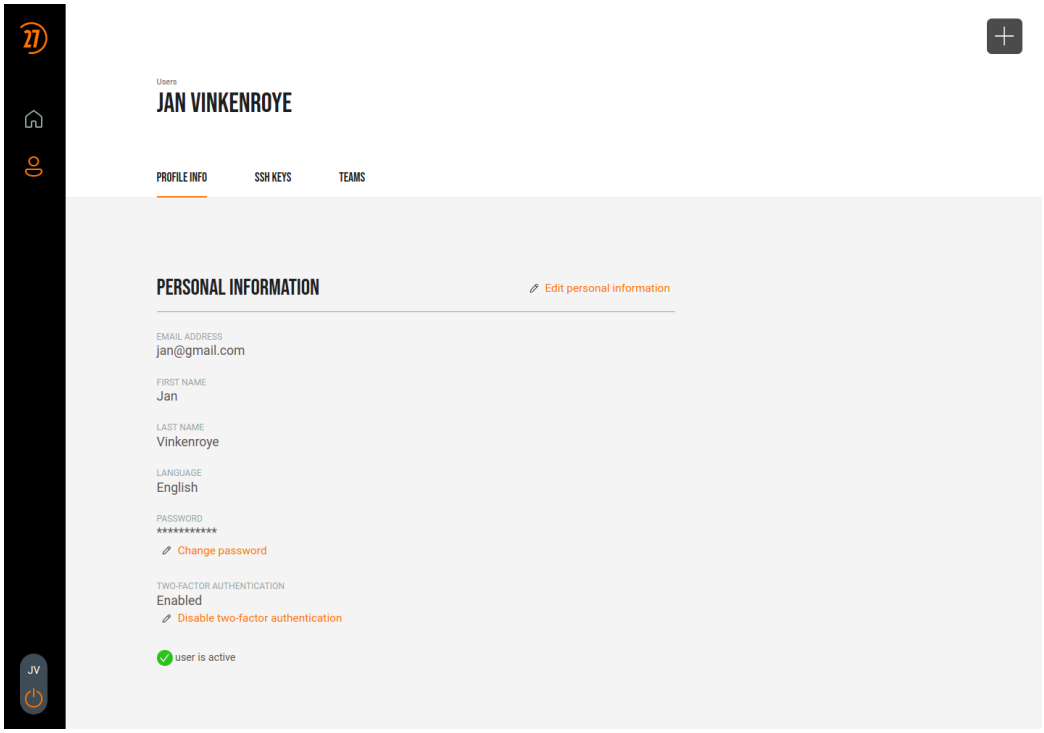
Wanneer tweefactorauthenticatie uitgeschakeld is, kan de gebruiker op een knop ‘Enable two-factor authentication’ klikken. Er verschijnt dan een zijpaneel met de QR-code en een veld om een TOTP in

te geven. De gebruiker kan een *authenticator app* op zijn telefoon instellen met behulp van deze QR-code.



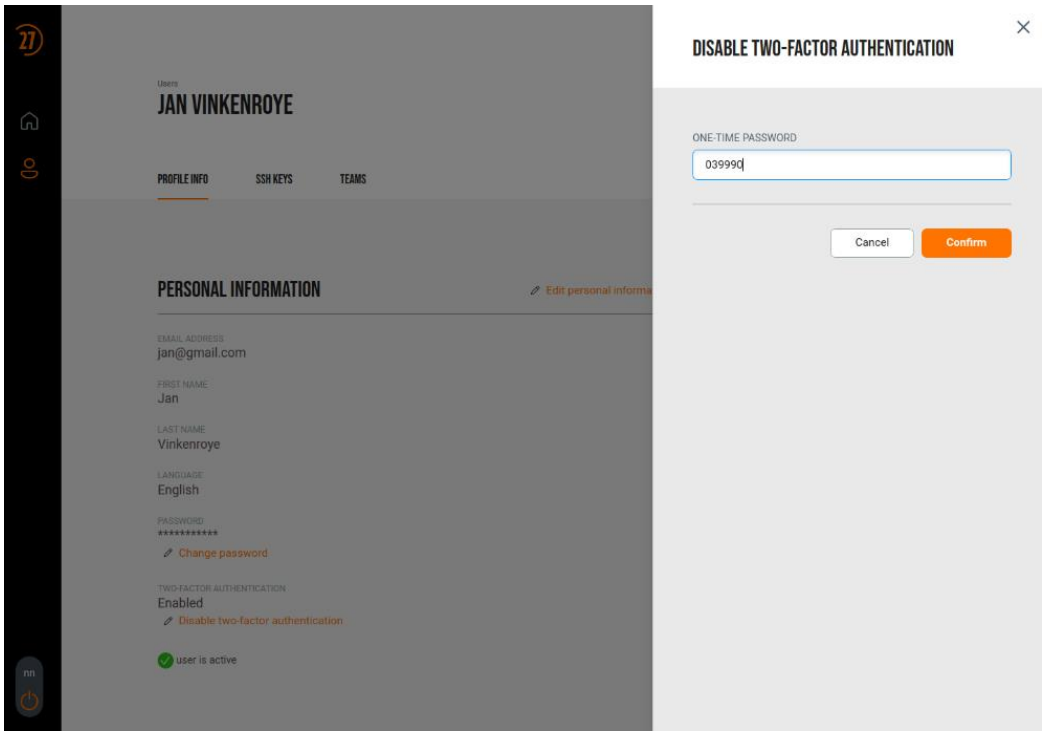
Figuur 9: Het zijpaneel om 2FA in te schakelen

Wanneer hij het TOTP dat gegenereerd wordt door zijn *authenticator app* ingeeft en op ‘Confirm’ klikt, verdwijnt het zijpaneel en geeft het profiel aan dat de tweefactorauthenticatie ingeschakeld is.



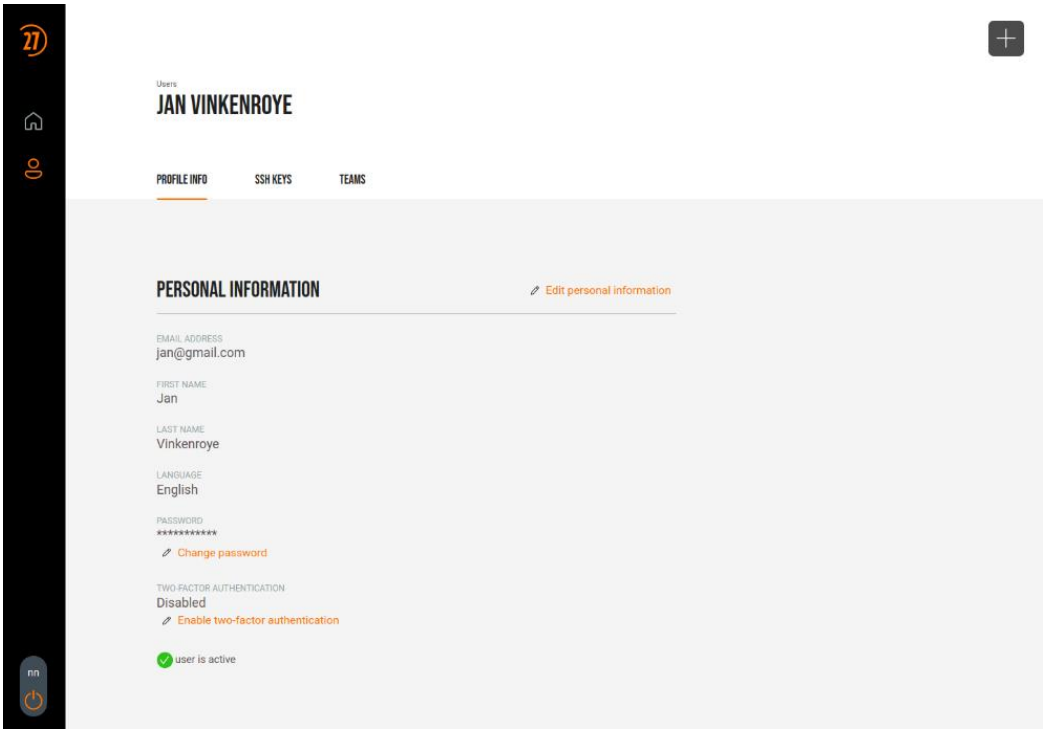
Figuur 10: Het gebruikersprofiel met 2FA ingeschakeld

Wanneer tweefactorauthenticatie ingeschakeld is, kan de gebruiker op een knop ‘Disable two-factor authentication’ klikken. Er verschijnt dan een zijpaneel met een veld om een TOTP in te geven.



Figuur 11: Het zijpaneel om 2FA uit te schakelen

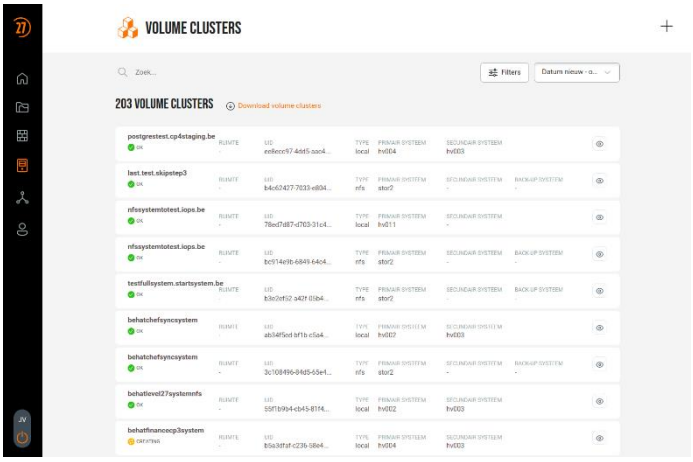
Wanneer hij het correcte TOTP ingeeft en op ‘Confirm’ klikt, verdwijnt het zijpaneel en geeft het profiel aan dat de tweefactorauthenticatie uitgeschakeld is.



Figuur 12: Het gebruikersprofiel met 2FA uitgeschakeld

4 UITWERKING STAGEOPDRACHT: FEATURES, VERBETERINGEN EN BUGREPARATIES

Het tweede deel van de stageopdracht omvat een aantal kleinere, ad hoc taken in het controlepaneel. Het gaat hierbij om de implementatie van features, verbeteringen en bugreparaties.

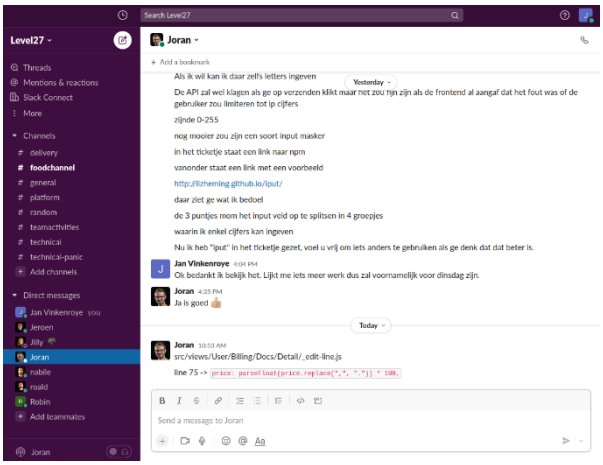


Figuur 13: Het controlepaneel van Level27

4.1 GEBRUIKTE TOOLS

4.1.1 SLACK

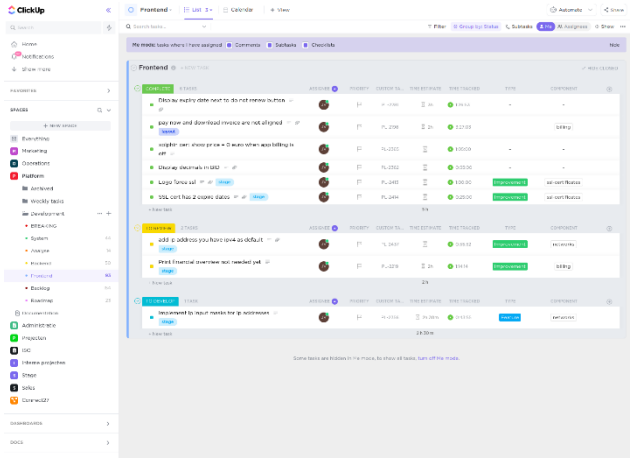
Binnen Level27 wordt Slack gebruikt als communicatietool. De meer algemene conversaties gebeuren in '*channels*' en voor de privéconversaties worden '*direct messages*' gebruikt.



Figuur 14: Een conversatie in Slack

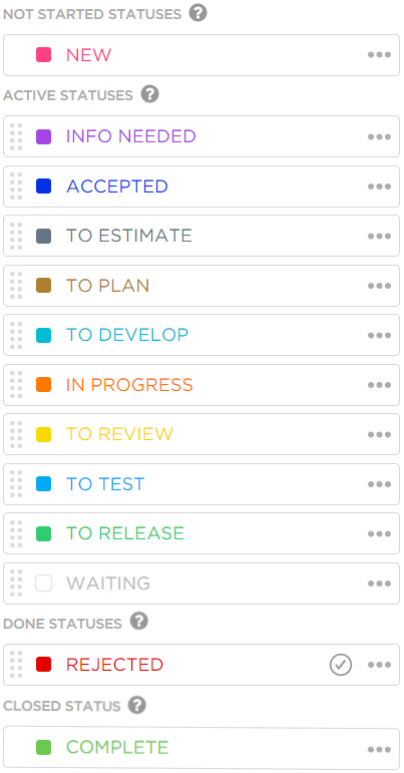
4.1.2 ClickUp

Level27 gebruikt ClickUp als tool voor het beheer van taken. Alle taken van de organisatie zijn onderverdeeld in een tiental '*spaces*'. Binnen elke '*space*' is er een verdere onderverdeling in mappen en lijsten.



Figuur 15: Een overzicht van taken in ClickUp

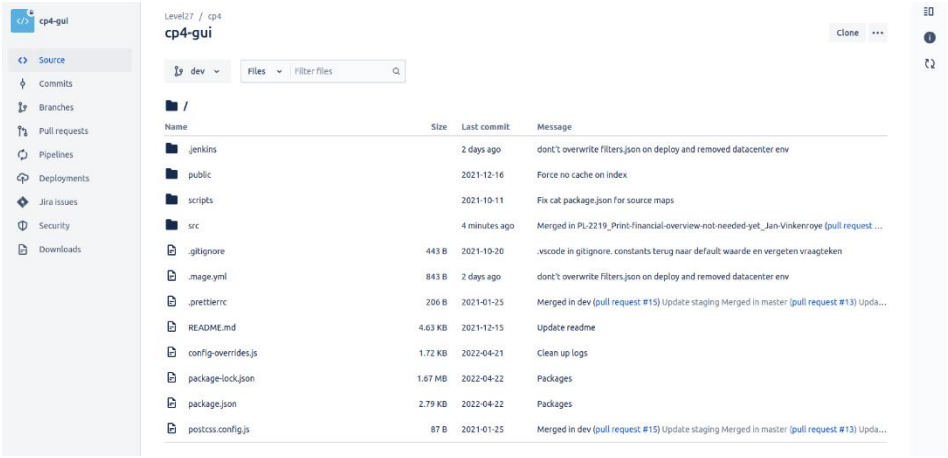
Iedereen kan een nieuwe taak aanmaken. Die taak komt dan in de algemene lijst 'Backlog' terecht. Wanneer de taak aanvaard wordt, wordt ze overgeheveld naar de corresponderende specifieke lijst. De taak doorloopt dan een aantal statussen tot ze hetzij afgehandeld, hetzij verworpen is.



Figuur 16: De statussen van de taken in ClickUp

4.1.3 BITBUCKET

De *repositories* van Level27 worden gehost op Bitbucket. Voor elke taak in ClickUp wordt een aparte *branch* aangemaakt. Wanneer de taak geïmplementeerd is, wordt de ClickUp-status op 'To Review' geplaatst en wordt er een *pull request* aangemaakt om de *branch* samen te voegen met de *developmentbranch*.



Figuur 17: Een repository in Bitbucket

4.2 BESCHRIJVING VAN DE REACT-OMGEVING

4.2.1 REACT EN REACT-DOM

React en React-dom zijn de centrale pakketten in een React.js-webapplicatie. Het controlepaneel van Level27 gebruikt versie 17 van beide bibliotheken.

4.2.2 JSX

Zoals de meeste React-applicaties gebruikt het controlepaneel JSX. JSX staat voor JavaScript XML. Het is een syntactische uitbreiding van JavaScript die toelaat om HTML-achtige elementen in JavaScript te schrijven.

4.2.3 CONTEXT API

In React heeft elke component zijn eigen *state*. Via *props* kan een *parent component* data delen met zijn *child components*. React biedt daarnaast de Context API aan die toelaat om data te delen over een hele *component tree*. De applicatie gebruikt de Context API om gegevens te delen tussen componenten die geen directe ouder-kindrelatie hebben. De applicatie gebruikt verder geen externe *state management libraries* zoals Redux, Mobx, Zustand, Recoil of Jotai.

4.2.4 ATOMIC DESIGN

De applicatie is opgebouwd volgens de principes van *atomic design*. De diverse componenten zijn onderverdeeld in *'atoms'*, '*molecules'*, '*higher-order components'* en '*views'*.

4.2.5 BIBLIOTHEKEN

De applicatie gebruikt een groot aantal bibliotheken of *libraries*. De belangrijkste bibliotheken worden hieronder besproken.

4.2.5.1 React-app-rewired

React-app-rewired is een npm-pakket dat verder bouwt op het pakket Create-react-app. React-app-rewired laat toe om de instellingen van Webpack, Jest en Babel aan te passen zonder gebruik te moeten maken van de 'eject'-functionaliteit in Create-react-app. React-app-rewired maakt gebruik van een bestand met de naam 'config-overrides.js' om instellingen van Webpack te overschrijven.

4.2.5.2 React-router

React-router is de meest populaire *routing library* voor React. Deze bibliotheek maakt navigatie mogelijk tussen verschillende componentweergaven. Daarnaast laat React-router toe de URL in de browser te wijzigen en de gebruikersinterface gesynchroniseerd te houden met de URL.

4.2.5.3 Styled components

Styled-components is een *CSS-in-JS library* die toelaat om CSS-stijlen op componentniveau te definiëren.

4.2.5.4 i18next

i18next is een internationalisatieframework voor JavaScript. React-i18next is de React-implementatie van i18next. React-i18next bevat een '*useTranslation' hook* die een vertaalfunctie teruggeeft. De vertalingen worden opgeslagen in JSON-bestanden. Voor elke taal wordt er een aparte directory aangemaakt onder de directory 'translations' en voor elke context een apart vertaalbestand.

4.2.5.5 Chart.js

Chart.js is een bibliotheek om geanimeerde, interactieve grafieken te maken.

4.2.5.6 Socket.io

Socket.io-client is een bibliotheek die real-time bidirectionele *event-based* communicatie met een Socket.io-server toelaat. Socket.io gebruikt WebSockets als transportmiddel waar mogelijk.

4.2.5.7 React-hook-form

React-hook-form is een bibliotheek om formulieren te valideren in React. React-hook-form maakt gebruik van *hooks* en gebruikt ongecontroleerde componenten en haalt de ingevoerde waarden op van de DOM via *refs*.

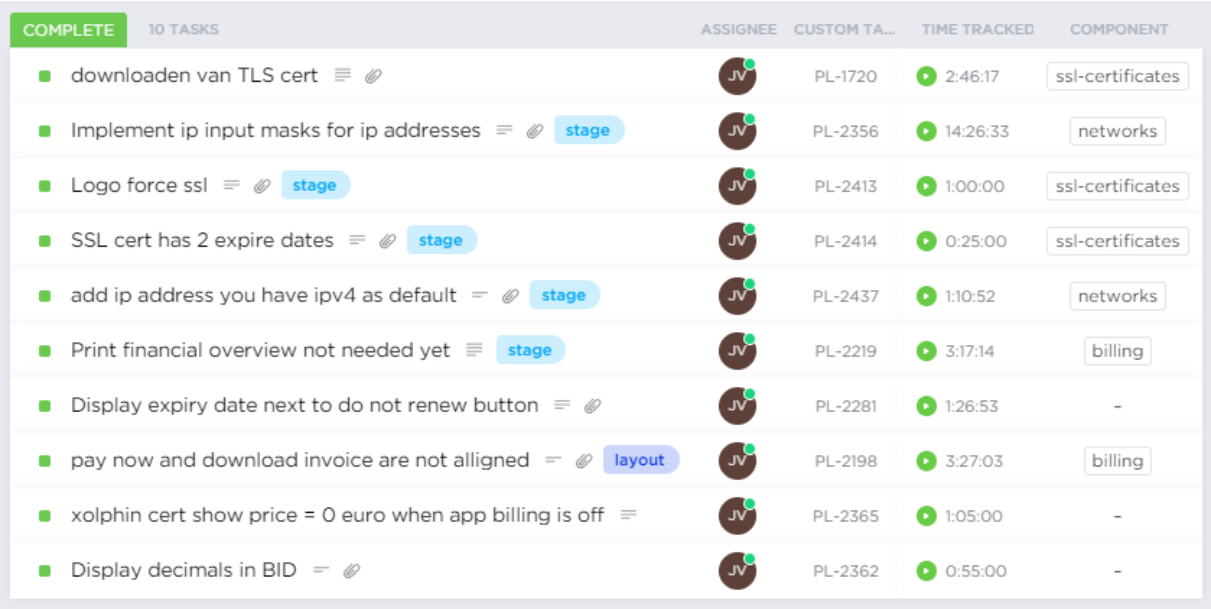
4.2.5.8 React-input-mask

React-input-mask voorziet een component met de naam 'InputMask'. Deze component laat toe om een invoermasker met een tekenreekssjabloon in te stellen zodat de input van de gebruiker beperkt kan worden tot een specifiek invoerformaat.

4.3 UITGEVOERDE TAKEN

Hieronder volgt een opsomming van alle taken die werden uitgevoerd in het kader van de stageopdracht. De naam, spelling en schrijfwijze van de taken werden rechtstreeks overgenomen van ClickUp.

- Display expiry date next to do not renew button
- Pay now and download invoice are not aligned
- Display decimals in BID
- Xolphin cert show price = 0 euro when app billing is off
- Download TLS cert
- Implement ip input masks for ip addresses
- Logo force ssl
- SSL cert has 2 expire dates
- Add ip address you have ipv4 default
- Print financial overview not needed yet



Figuur 18: Overzicht van de uitgevoerde taken in ClickUp

II. ONDERZOEKSOPDRACHT

1 PROBLEEMSTELLING EN ONDERZOEKSVRAAG

Het controlepaneel van Level27 gebruikt momenteel een wachtwoord om zijn gebruikers te authenticeren. Om de veiligheid van de authenticatie te verhogen, wil Level27 tweefactorauthenticatie implementeren.

Het authenticatielandschap van vandaag is erg complex. Er bestaan talloze implementaties, termen, filosofieën, standaarden, *best practices* en *worst practices*. De eerste doelstelling van deze onderzoeksopdracht is om een globaal overzicht te bieden van het hedendaagse authenticatielandschap.

Een tweede doelstelling is meer specifiek. De onderzoeksopdracht wil nagaan welke bijkomende verificatiemethoden er allemaal gebruikt worden in het kader van multifactorauthenticatie. De onderzoeksopdracht wil ook de voor- en nadelen van deze methoden op het vlak van veiligheid, gebruiksgemak en implementatiekost identificeren.

Een laatste doelstelling is de afweging van de voor- en nadelen van de diverse methoden, rekening houdend met de concrete casus van het controlepaneel van Level27. De bedoeling is om te komen tot een advies over de meest aangewezen bijkomende verificatiemethode.

De hoofdvraag is de volgende: Op welke manier kan Level27 het authenticatieproces van zijn controlepaneel beter beveiligen?

Op basis van de drie doelstellingen kunnen drie deelvragen geformuleerd worden:

- Hoe ziet het huidige authenticatielandschap eruit? Welke authenticatiemethoden en authenticatiestandaarden bestaan er?
- Welke verificatiemethoden bestaan er allemaal? Wat zijn de voor- en nadelen van deze methoden op het vlak van veiligheid, gebruiksgemak en implementatiekost?
- Welke bijkomende verificatiemethode is het meest aangewezen om als tweede factor te implementeren in het controlepaneel van Level27?

2 METHODE VAN ONDERZOEK

Het onderzoek is een literatuurstudie.

In hoofdstuk 3 wordt het begrip ‘authenticatie’ omschreven en wordt het begrip afgebakend van de verwante begrippen ‘identificatie’ en ‘autorisatie’. Ook wordt ingegaan op de verschillende authenticatiefactoren.

Hoofdstuk 4 bespreekt de problemen met het gebruik van wachtwoorden en omschrijft een aantal mogelijke oplossingen voor die problemen.

Hoofdstuk 5 gaat kort in op het concept van gefedereerd identiteitsbeheer.

Hoofdstuk 6 belicht het begrip ‘*one-time password*’ (OTP) en omschrijft de verschillende soorten OTP’s.

In hoofdstuk 7 worden een aantal authenticatiestandaarden bekeken. Het gaat om de authenticatieprotocollen van de FIDO Alliance en de authenticatiealgoritmes van Initiative for Open Authentication (OATH).

Hoofdstuk 8 biedt een overzicht van de meest gebruikte bijkomende verificatiemethoden.

In hoofdstuk 9 worden deze verificatiemethoden met elkaar vergeleken op het vlak van veiligheid, gebruiksvriendelijkheid en implementatiekost.

In hoofdstuk 10 tenslotte wordt er een aanbeveling gedaan aan Level27 over welke bijkomende verificatiemethode het meest geschikt is om te gebruiken in hun controlepaneel.

3 AUTHENTICATIE

3.1 IDENTIFICATIE, AUTHENTICATIE EN AUTORISATIE

Authenticatie is als concept gerelateerd aan de concepten van identificatie en autorisatie.

Identificatie is het proces waarbij een gebruiker van een systeem een identiteit claimt. Klassiek gebeurt dat door een e-mailadres of een gebruikersnaam in te geven.

In de fase van de **authenticatie** wordt deze claim gecontroleerd. De gebruiker levert een identiteitsbewijs aan op basis waarvan zijn geclaimde identiteit geverifieerd wordt. Dit identiteitsbewijs heeft de vorm van een geheim dat enkel gekend is door de gebruiker en door het systeem dat instaat voor de authenticatie [1]. Het wachtwoord is het meest bekende en meest gebruikte identiteitsbewijs.

Eens een gebruiker geauthenticeerd is, kan hij op basis van zijn identiteit toegang krijgen tot bepaalde resources. Dit noemt men **autorisatie**. Er bestaan verschillende soorten autorisaties: leesrechten, schrijfrechten, verwijderrechten … Om het autorisatiebeheer te vergemakkelijken, worden rechten vaak toegekend aan groepen personen. Dit heet *role-based access control*.

3.2 AUTHENTICATIEFACTOREN

Authenticatie kan op verschillende manieren gebeuren. In de literatuur worden meestal drie authenticatiefactoren onderscheiden: kennis (iets dat de gebruiker weet), bezit (iets dat de gebruiker heeft) en inherentie (iets wat de gebruiker is).

3.2.1 KENNIS

Authenticatie die gebaseerd is op kennis maakt gebruik van geheime informatie. Het is belangrijk dat deze informatie enkel gekend is door de gebruiker en door het authenticatiesysteem en dat het geheim niet makkelijk te raden valt. Voorbeelden van authenticatiemethoden gebaseerd op kennis zijn wachtwoorden, pincodes, geheime zinnen of *passphrases*, *dotted patterns* en *security questions*.

3.2.2 BEZIT

Authenticatie op basis van bezit komt in de computerwereld meestal voor in de vorm van een *security token*. Deze tokens bevatten een geheime code waarmee de identiteit van de gebruiker geverifieerd wordt.

3.2.3 INHERENTIE

Bij authenticatie op basis van inherentie, gaat het meestal om biometrische methoden als vingerafdrukherkenning, gezichtsherkenning, stemherkenning of irisherkenning.

3.2.4 ANDERE FACTOREN

Sommige auteurs beschouwen locatie als vierde authenticatiefactor [2]. Gebruikers kunnen bijvoorbeeld enkel toegang krijgen tot een systeem wanneer zij zich in het bedrijfsnetwerk bevinden.

Deze factor wordt meestal gebruikt in combinatie met een andere authenticatiefactor. Andere auteurs zien locatie eerder als een autorisatiefactor [3].

Ook tijd wordt soms als een authenticatiefactor beschouwd [4]. Als deze factor gebruikt wordt, is authenticatie enkel mogelijk binnen een bepaald tijdsbereik. In de praktijk wordt authenticatie op basis van tijd bijna altijd samen met andere authenticatiefactoren gebruikt.

4 NAAR EEN STERKERE AUTHENTICATIE

4.1 DE PROBLEMEN MET WACHTWOORDEN

Statische wachtwoorden zijn het meest gebruikte authenticatiemiddel. Het grote voordeel van authenticatie met wachtwoorden is dat het goedkoop en makkelijk te implementeren is. Daarnaast is authenticatie met behulp van wachtwoorden ook makkelijk te begrijpen en te gebruiken.

Een goed wachtwoord heeft de volgende eigenschappen:

- Het is voldoende **lang**. Korte wachtwoorden hebben een lage entropie en zijn daarom kwetsbaar voor *brute force attacks*.
- Het is voldoende **complex.** Een wachtwoord bestaat best uit een combinatie van hoofdletters, kleine letters, nummers en speciale tekens. Weinig complexe wachtwoorden hebben eveneens een lagere entropie en een hogere kwetsbaarheid voor *brute force attacks*.
- Het is volledig **willekeurig**. Enigszins voorspelbare wachtwoorden houden het risico van *dictionary attacks* en *rainbow table attacks* in, ten minste voor zover er geen *salting* van de wachtwoorden gebeurt.
- Het is **uniek**. Wanneer eenzelfde wachtwoord gebruikt wordt op meerdere plaatsen, krijgt de hacker of dief die één wachtwoord weet te bemachtigen in één keer toegang tot verschillende systemen.

Idealiter zou elke gebruiker dus lange, complexe en volledig willekeurige wachtwoorden moeten gebruiken, die bovendien uniek zijn voor elke account. Dit is niet erg gebruiksvriendelijk. Het is bovendien in de praktijk enkel mogelijk wanneer de wachtwoorden ergens worden opgeslagen of opgeschreven, wat het risico op diefstal inhoudt.

Daarenboven zou elke authenticatieserver de wachtwoorden moeten *salt*en en hashen met een veilig algoritme. De praktijk wijst uit dat dit niet altijd gebeurt.

Tenslotte is zelfs het perfecte wachtwoord kwetsbaar voor onderscheppingsaanvallen als *man-in-the-middle attacks*, *man-in-the-browser attacks* en *replay attacks*.

4.2 MULTIFACTORAUTHENTICATIE

Om de veiligheid van de authenticatie te verhogen, wordt er meer en meer gebruik gemaakt van multifactorauthenticatie. Bij multifactorauthenticatie wordt er meer dan één authenticatiefactor gelijktijdig gebruikt om de identiteit van een persoon te bewijzen. Vaak wordt ook de term tweefactorauthenticatie gebruikt. In dat geval worden precies twee factoren gebruikt.

Wanneer bij een poging tot authenticatie minstens één van de gevraagde factoren ontbreekt of incorrect is, wordt de gebruiker niet geauthenticeerd. De veiligheid ligt dus hoger dan wanneer er slechts één enkele factor gebruikt wordt. Een hacker moet er al in slagen om twee verschillende factoren te compromitteren.

In de praktijk wordt als eerste factor meestal de factor kennis gebruikt in de vorm van een wachtwoord. Als bijkomende verificatiemethode zijn allerlei vormen van *one-time passwords* het populairst. De laatste jaren is ook biometrische verificatie in opmars als tweede of derde factor. In hoofdstuk 8 wordt dieper ingegaan op de verschillende mogelijke bijkomende verificatiemethoden.

Er zijn verschillende manieren om met multifactorauthenticatie om te gaan:

- Multifactorauthenticatie kan systematisch vereist zijn voor elke authenticatie van elke gebruiker
- Multifactorauthenticatie kan enkel vereist zijn voor die gebruikers die zelf aangegeven hebben gebruik te willen maken van multifactorauthenticatie
- Multifactorauthenticatie kan enkel vereist zijn in het geval van ongebruikelijke of enigszins verdachte situaties, zoals een gebruiker die probeert in te loggen vanaf een nieuw toestel of vanaf een nieuw IP-adres
- Multifactorauthenticatie kan enkel vereist zijn voor de toegang tot bepaalde gevoelige resources
- …

4.3 TWEESTAPSVERIFICATIE

Tweestapsverificatie is een begrip dat verwant is aan tweefactorauthenticatie. Bij tweestapsverificatie moet de gebruiker twee verschillende methoden gebruiken om zich te kunnen authenticeren. In tegenstelling tot tweefactorauthenticatie, kan er tweemaal gebruik gemaakt worden van dezelfde factor [5]. In de praktijk is dat meestal de factor kennis.

4.4 PASSWORDLESS AUTHENTICATIE

Passwordless authenticatie is een vorm van authenticatie die geen gebruik maakt van een wachtwoord of een andere op kennis gebaseerde methode. De gebruiker claimt zijn identiteit door een gebruikersnaam, e-mailadres of telefoonnummer in te geven en daarna wordt het authenticatieproces doorlopen waarbij de gebruiker een op bezit of inherentie gebaseerd identiteitsbewijs aanlevert.

Passwordless authenticatie wordt vaak verward met tweefactorauthenticatie. Beide maken gebruik van een verificatiemethode gebaseerd op bezit of inherentie. Bij tweefactorauthenticatie wordt deze methode echter bovenop het klassieke wachtwoord gebruikt. Bij *passwordless* authenticatie is er meestal slechts één authenticatiefactor.

4.5 STEP-UPAUTHENTICATIE

*Step-up*authenticatie is een vorm van authenticatie waarbij er gebruik gemaakt wordt van verschillende authenticatieniveaus. Een gebruiker kan bijvoorbeeld in eerste instantie inloggen met enkel een gebruikersnaam en een wachtwoord. Voor de authenticatieserver bevindt hij zich dan op ‘authenticatieniveau 1’. Wanneer diezelfde gebruiker later een bijkomende authenticatiefactor

aanlevert, zal de authenticatieserver het authenticatieniveau van de gebruikerssessie verhogen naar een hoger niveau. Aan elke resource kan een bepaald minimumauthenticatieniveau gekoppeld worden in het kader van het autorisatiebeleid. Op die manier kan de sterkte van de gevraagde authenticatie afhankelijk gemaakt worden van de gevoeligheid van de door de gebruiker gevraagde resources. Ook de time-outs van de sessies kunnen afhankelijk gemaakt worden van het authenticatieniveau [6].

4.6 OUT-OF-BANDAUTHENTICATIE

*Out-of-band*authenticatie is een vorm van multifactorauthenticatie waarbij de verschillende verificatiemethoden een afzonderlijk communicatiekanaal gebruiken. Dit biedt een extra bescherming tegen *man-in-the-middle attacks* aangezien een hacker beide communicatiekanalen moet compromitteren. Meestal wordt er naast het internet gebruik gemaakt van een mobiele telefoonverbinding als tweede communicatiekanaal.

Voorbeelden van *out-of-band*methoden zijn verificatie via telefoongesprek of sms.

5 GEFEDEREERD IDENTITEITSBEHEER

Het concept van gefedereerd identiteitsbeheer verwijst naar het delen van de elektronische identiteiten van gebruikers tussen verschillende systemen. De referenties van de gebruiker worden op één centrale plaats bijgehouden. Gebruikers kunnen zich authenticeren bij het centrale systeem en vervolgens toegang krijgen tot andere gefedereerde systemen zonder bij elk systeem afzonderlijk te moeten inloggen.

5.1 SINGLE SIGN-ON (SSO)

Single sign-on of éénmalige aanmelding is een manier van authenticatie waarbij sessie-informatie van de gebruikers gedeeld wordt tussen verschillende softwaresystemen. De gebruikers moeten hun referenties slechts één keer ingeven om toegang te krijgen tot verschillende applicaties.

De gebruiker krijgt een authenticatietoken wanneer hij inlogt bij de SSO-server. Wanneer de gebruiker toegang wil krijgen tot een applicatie, wordt er nagegaan of hij al een geldig authenticatietoken bezit. Is dat niet het geval, dan wordt hij omgeleid naar de centrale SSO-server waar hij zich kan aanmelden.

SSO-implementaties maken dikwijls gebruik van federatieprotocollen als SAML, OpenID Connect of OAuth.

5.2 SOCIAL LOGIN

Social login is een vorm van *single sign-on* waarbij een derde partij gebruik maakt van de gebruikersinformatie van sociale netwerken of computergerelateerde bedrijven om zijn gebruikers te identificeren en te authenticeren.

De gebruiker kan zich aanmelden bij een applicatie door gebruik te maken van zijn login van diensten als Facebook, LinkedIn, Google, Microsoft, Apple of Twitter. De gebruiker hoeft dus geen aparte account aan te maken en geen apart wachtwoord te onthouden.

5.3 FEDERATIEPROTOCOLLEN

5.3.1 OPEN AUTHORIZATION (OAUTH)

OAuth 2.0 is momenteel de meest gebruikte standaard voor autorisatie. OAuth maakt gebruik van JSON Web Tokens (JWT's) die toegang verlenen tot een bepaalde resource voor een bepaalde duur.

Authenticatie maakt geen deel uit van het OAuth-protocol [7]. OAuth kan wel gecombineerd worden met een authenticatieprotocol als OpenID Connect of SAML.

5.3.2 OpenID Connect

OpenID Connect is een authenticatieprotocol dat verder bouwt op OAuth 2.0. Er wordt een extra JWT gebruikt als identiteitstoken. OpenID Connect wordt veel gebruikt in applicaties die gericht zijn op eindgebruikers.

5.3.3 Security Assertion Markup Language (SAML)

SAML staat volledig los van OAuth. SAML gebruikt Extensible Markup Language (XML) om authenticatieberichten uit te wisselen tussen *identity providers* en *service providers*. SAML-tokens bevatten geen gebruikersgegevens [8]. SAML wordt vooral gebruikt voor interne applicaties in de bedrijfswereld.

6 ONE-TIME PASSWORDS

Een *one-time password* (OTP), ook wel dynamisch wachtwoord, is een wachtwoord waarvan de geldigheid beperkt is tot één enkele loginsessie of transactie. OTP's worden gebruikt door *security tokens*, *challenge-response tokens* en bij diverse vormen van *out-of-band*authenticatie zoals sms of e-mail.

Bij *security tokens* vindt er vooraf een informatie-uitwisseling plaats tussen de authenticatieserver en het token. Er worden afspraken gemaakt over het te gebruiken algoritme en de *seed*waarden worden gedeeld. Op basis van dit algoritme en deze *seed*waarden kunnen de server en het token onafhankelijk van elkaar dezelfde eindeloze stroom aan pseudo-willekeurige wachtwoorden genereren.

Bij *out-of-band*mechanismen worden de OTP's door de server aangemaakt en doorgestuurd naar de een toestel van de gebruiker.

Bij *challenge-response tokens* is het OTP de oplossing van de challenge die aangemaakt wordt door de server.

De gebruiker moet in al de systemen het OTP ingeven in de gebruikersinterface om zich te kunnen authenticeren.

Het grote voordeel van OTP's ten opzichte van statische wachtwoorden is dat OTP's minder kwetsbaar zijn voor onderscheppingsaanvallen.

6.1 LOCKSTEP-SYNCHRONIZED ONE-TIME PASSWORDS

Lockstep-synchonised OTP's worden aangemaakt door een algoritme dat gebruik maakt van een teller. Deze OTP's worden ook wel *event-based* OTP's genoemd.

Bij de installatie van de verbinding vindt er een synchronisatie van de teller met de authenticatieserver plaats. Daardoor weet de server welke wachtwoordwaarde moet verwacht worden. De meeste tokens hebben een knop waarop de gebruiker moet klikken. Die knop verhoogt de teller en genereert een nieuw wachtwoord [1]. Iedere keer dat er een OTP gevalideerd wordt, vindt er indien nodig opnieuw een synchronisatie van de teller plaats om te voorkomen dat het toestel en de server gedesynchroniseerd geraken. Indien de server het ingegeven OTP niet herkent, zal hij de teller een aantal keer verhogen om te kijken of het ingegeven OTP gevalideerd kan worden [9].

Een nadeel van *lockstep-synchronised* OTP's is dat het toestel en de server gedesynchroniseerd kunnen geraken wanneer de gebruiker een aantal keren op de knop klikt zonder het wachtwoord te valideren. De teller van het toestel wordt dan verhoogd zonder dat de server hiervan op de hoogte is. In dat geval zal de gebruiker niet meer kunnen inloggen en moet er opnieuw een synchronisatie plaatsvinden.

Een ander nadeel is dat, wanneer er meerdere authenticatieservers gebruikt worden, de tellers ook tussen de servers onderling moeten worden gesynchroniseerd.

Een laatste nadeel is dat de gebruiker slechts één toestel kan gebruiken om zich aan te melden. Dit laatste is wel een voordeel vanuit veiligheidsstandpunt, aangezien het de gebruiker verplicht om slechts één toestel te gebruiken voor het genereren van de OTP's.

6.2 TIME-SYNCHRONIZED ONE-TIME PASSWORDS

Time-synchronised one-time passwords worden aangemaakt door een cryptografisch algoritme dat gebruik maakt van de tijd.

Bij de registratie wordt een gedeeld geheim uitgewisseld tussen de authenticatieserver en het token. Er worden ook afspraken gemaakt over de starttijdswaarde en het tijdsinterval. Dit gebeurt vaak door een QR-code te scannen. Elk tijdsinterval wordt er een nieuw OTP aangemaakt. Typisch wordt 30 seconden of 1 minuut als tijdsinterval gebruikt. Het token en de authenticatieserver genereren het OTP onafhankelijk van elkaar op basis van hetzelfde gedeeld geheim, dezelfde tijdswaarde en hetzelfde tijdsinterval. Daardoor kan de authenticatieserver het bezit van het token controleren, zolang de tijdswaarde gesynchroniseerd blijft.

De geldigheid van een *time-synchronised* OTP is in principe beperkt tot het tijdsinterval waarop het OTP betrekking heeft. Vanuit het oogpunt van gebruiksvriendelijkheid wordt de geldigheid meestal uitgebreid tot het vorige en het volgende tijdsinterval [10].

In tegenstelling tot andere OTP's kunnen *time-synchronised* OTP's kunnen standaard hergebruikt worden binnen het tijdsinterval.

Een mogelijk nadeel van *time-synchronised* OTP's is dat de timers van het token en de server na verloop van tijd gedesynchroniseerd kunnen geraken. Het gevolg hiervan is dat de gegenereerde wachtwoorden niet meer geldig zijn totdat er opnieuw een synchronisatie plaatsvindt.

Een nadeel van tijdssynchronisatie ten opzichte van *lockstep*synchronisatie is dat de tijdswaarde algemeen bekend is, in tegenstelling tot de teller. Bij *lockstep*synchronisatie moet een hacker die erin slaagt om het gedeeld geheim te bemachtigen altijd nog de teller zien te achterhalen.

6.3 TRANSMISSION-BASED ONE-TIME PASSWORDS

*Out-of-band*systemen werken op een andere manier. Hier maakt enkel de authenticatieserver het OTP aan. Wanneer de gebruiker zich wil aanmelden, stuurt de server het aangemaakte OTP naar een toestel van de gebruiker via sms, email of telefoongesprek. Het toestel van de gebruiker dient louter als communicatiemiddel. Er is dus geen gedeeld geheim dat gestolen kan worden. Het is één van de makkelijkste manieren om een OTP te implementeren [11].

6.4 CHALLENGE-RESPONSE-BASED ONE-TIME PASSWORDS

Een *challenge-response token* gebruikt asymmetrische cryptografie om het bewijs van het bezit van een private sleutel te leveren. De authenticatieserver maakt een challenge aan die geëncrypteerd wordt met de publieke sleutel van de gebruiker. Die challenge kan enkel opgelost worden door gebruik te maken van de overeenstemmende private sleutel. De gebruiker kan het bewijs van het bezit van zijn private sleutel leveren door de ontcijferde challenge als OTP door te sturen naar de server.

Challenge-response tokens zijn wat minder gebruiksvriendelijk dan andere OTP-tokens omdat er een extra stap nodig is, namelijk het ingeven van de challenge op het toestel. De implementatie van een *challenge generator* in de authenticatieserver is ook iets complexer.

Het voordeel van *challenge-response tokens* ten opzichte van *security tokens* is dat er geen synchronisatie hoeft te gebeuren tussen het token en de server.

7 AUTHENTICATIESTANDAARDEN

7.1 FAST IDENTITY ONLINE (FIDO)

De FIDO Alliance is een initiatief om te komen tot open, universele authenticatiestandaarden. Het doel van de organisatie is om de afhankelijkheid van wachtwoorden te verminderen.

FIDO wordt ondersteund door een groot aantal gerenommeerde bedrijven waaronder Amazon, Apple, Google, Mastercard, Meta, Microsoft, PayPal, RSA, VISA en Yubico [12]. FIDO ondersteunt technologieën als USB-tokens, chipkaarten, NFC en biometrische authenticatie.

7.1.1 UNIVERSAL SECOND FACTOR (U2F)

De U2F-standaard werd ontwikkeld door Google en Yubico en werd voor het eerst gepubliceerd in 2014. Dit protocol vereist het gebruik van een fysieke USB- of NFC-sleutel voor de verificatie. U2F wordt gebruikt als tweede, additionele factor om de veiligheid van een login op basis van een wachtwoord te vergroten.

7.1.2 UNIVERSAL AUTHENTICATION FRAMEWORK (UAF)

UAF is een standaard die ongeveer gelijktijdig met U2F tot stand kwam. UAF richt zich op *passwordless* authenticatie. Gebruikers kunnen een toestel registreren bij een webapplicatie en dat toestel gebruiken om hun identiteit te bewijzen.

7.1.3 FIDO2

FIDO2 is de overkoepelende term voor de nieuwste FIDO-specificaties. FIDO2 bestaat uit de WebAuthn-specificatie van het World Wide Web Consortium (W3C) en het Client to Authenticator Protocol (CTAP). FIDO2 kan op zichzelf of als extra factor gebruikt worden. FIDO2 vervangt in die zin UAF en U2F.

7.1.3.1 Web Authentication (WebAuthn) specification

WebAuthn is een authenticatiestandaard die webapplicaties in staat stelt om gebruikers op een eenvoudige en veilige manier te authenticeren. WebAuthn kan ingebouwd worden in browsers en platformen. Momenteel bieden de laatste versies van Chrome, Firefox, Android Browser, Edge, Safari en Opera [13] ondersteuning aan voor WebAuthn. Ook Windows 10 en Android ondersteunen WebAuthn [14].

De WebAuthn-specificatie definieert drie partijen:

- de ‘FIDO2 Authenticator’: dit kan een externe of een interne *authenticator* zijn
- de ‘WebAuthn Client’: meestal de browser
- de ‘WebAuthn Relying Party’: de webapplicatie

WebAuthn kent twee ceremonies: registratie en authenticatie.

Bij de **registratie** maakt de authenticator een asymmetrisch sleutelpaar aan. De private sleutel wordt samen met informatie over de webapplicatie bijgehouden op het toestel van de gebruiker. De publieke sleutel wordt doorgestuurd naar de webapplicatie, die de sleutel opslaat.

Authenticatie gebeurt door middel van een *challenge-response*mechanisme waarbij de gebruiker het bezit van zijn private sleutel bewijst. De private sleutel kan slechts gebruikt worden nadat de gebruiker eerst de sleutel ontgrendelt. Die ontgrendeling kan bijvoorbeeld gebeuren door een pincode in te geven of een vingerafdruk te scannen.

WebAuthn werd in maart 2019 door W3C gepromoveerd tot officiële webstandaard [15].

7.1.3.2 Client to Authenticator Protocol (CTAP)

CTAP is een standaard voor de communicatie tussen een externe *authenticator*, zoals een *security key* of een mobiele telefoon, en een clientapplicatie die WebAuthn ondersteunt. CTAP bouwt verder op U2F en is complementair aan WebAuthn.

7.2 Initiative for Open Authentication (OATH)

OATH is een samenwerkingsverband tussen een groot aantal informatiebeveiligingsbedrijven. Het doel van OATH is om open standaarden voor referentiearchitectuur voor een sterke authenticatie te ontwikkelen. Tot de leden behoren onder andere Citrix, IBM en Verisign.

OATH specifieert drie gestandaardiseerde authenticatiealgoritmes: *HMAC-based one-time password* (HOTP), *time-based one-time password* (TOTP) en *OATH challenge-response algorithm* (OCRA).

7.2.1 HMAC-Based One-Time Password (HOTP)

Het *HMAC-based-OTP*-algoritme (HOTP) is de OATH-standaard voor *lockstep-synchronised* OTP's. Het is een algoritme om OTP's te genereren op basis van een *hash-based message authentication code* (HMAC). De standaard werd voor het eerst gepubliceerd in december 2005 [16].

Er worden een aantal parameters gedeeld tussen de *authenticator* en de geauthenticeerde: het gedeeld geheim, de cryptografische hashmethode, de beginwaarde van de teller en de lengte van het OTP. De *authenticator* en de geauthenticeerde genereren de OTP's onafhankelijk van elkaar op basis van die parameters.

HOTP wordt ondersteund door Yubiko Yubikey en door sommige mobiele *authenticator apps* als Google Authenticator, Duo Mobile en FreeOTP.

7.2.2 Time-Based One-Time Password (TOTP)

Het TOTP-algoritme is de OATH-standaard voor *time-synchronised* OTP's [11]. De voorlopige versie werd voor het eerst gepubliceerd in 2008.

TOTP is een uitbreiding van HOTP. Bij TOTP worden er nog twee extra parameters gedeeld tussen de *authenticator* en de geauthenticeerde, namelijk de Unix-tijd die als startwaarde gebruikt wordt en

het tijdsinterval. TOTP gebruikt hetzelfde algoritme als HOTP maar vervangt de teller door het aantal gepasseerde tijdsintervallen tussen de huidige tijd en de starttijd [17].

Twilio Authy, Google Authenticator, Microsoft Authenticator, Duo Mobile en vrijwel alle andere mobiele *authenticator apps* ondersteunen TOTP. Ook Yubico Yubikey ondersteunt TOTP.

7.2.3 OATH CHALLENGE-RESPONSE ALGORITHM (OCRA)

OCRA is de OATH-standaard voor *challenge-response*authenticatie. OCRA werd voor het eerst gepubliceerd in 2010 [18].

Ook dit algoritme bouwt verder op HOTP. Het verschil met de voorgaande standaarden is drieledig.

Ten eerste is OCRA een asynchroon authenticatiesysteem. Het gebruikt daarvoor een *challenge-response*schema.

Ten tweede heeft de geauthenticeerde nu ook de mogelijkheid om de server te identificeren. De authenticatie kan dus in twee richtingen gebeuren.

Tenslotte laat OCRA in tegenstelling tot HOTP en TOTP het gebruik van op sleutels gebaseerde elektronische handtekeningen toe. Deze handtekeningen zijn een variatie op het *challenge-response*mechanisme waarbij de data die ondertekend moet worden gebruikt wordt om de challenge te definiëren.

8 BIJKOMENDE VERIFICATIEMETHODEN

Dit hoofdstuk biedt een overzicht van een aantal veel gebruikte verificatiemethoden. Het gaat om methoden die als bijkomende factor gebruikt worden in het kader van multifactorauthenticatie. De meeste van deze methoden kunnen daarnaast ook op zichzelf gebruikt worden als *passwordless* authenticatie.

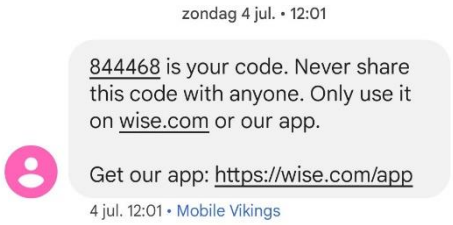
8.1 TELEFOONGESPREK

Verificatie kan gebeuren door middel van een telefoongesprek.

Een klassiek voorbeeld is een telefoontje van een medewerker van een bank om een verdachte transactie te verifiëren [1]. Verificatie door middel van een telefoongesprek kan ook op een geautomatiseerde manier gebeuren. Vaak wordt er een code doorgegeven die de gebruiker moet ingeven in een loginscherm.

8.2 SMS

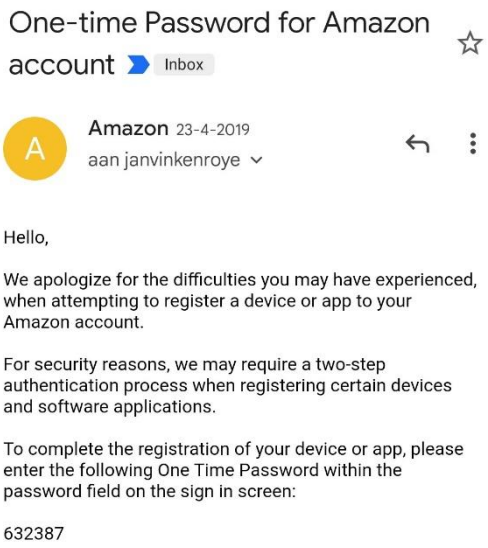
Verificatie via sms werkt op een gelijkaardige manier. Er wordt een code aangemaakt en verzonden via een sms-bericht naar het telefoonnummer van de gebruiker. De gebruiker heeft deze code nodig om zich te kunnen authenticeren.



Figuur 19: Sms-verificatie

8.3 E-MAIL

Bij verificatie via e-mail wordt er een automatisch bericht gestuurd naar de mailbox die geregistreerd werd door de gebruiker. Dit e-mailbericht bevat een code die ingegeven moet worden of een link waarop geklikt moet worden om toegang te krijgen.



Figuur 20: E-mailverificatie

8.4 PUSHNOTIFICATIES

Bij deze methode stuurt de authenticatieserver een pushnotificatie naar een app op alle toestellen die de gebruiker geregistreerd heeft. De notificatie bevat gegevens over de vermoedelijke locatie van de login op basis van het IP-adres. De gebruiker kan dan beslissen om de login al dan niet toe te laten. Pushnotificaties worden onder meer gebruikt door Apple, Google, Microsoft en Duo Security.

Figuur 21: Verificatie met een pushnotificatie

8.5 QR-CODE

Authenticatie met behulp van een *quick response code* of QR-code wordt vaak gebruikt voor kinderen en mensen die moeite hebben met de traditionele manier van authenticatie met behulp van een wachtwoord. De QR-code bevat de gebruikersnaam en soms ook een wachtwoord.

Bij de registratie ontvangt de gebruiker een QR-code die hem toegang verleent tot de applicatie. Hij kan zich aanmelden door een afdruk van de QR-code te scannen met de camera van zijn toestel.

Vaak wordt er dan een extra authenticatiefactor gevraagd, bijvoorbeeld in de vorm van een pictogram.

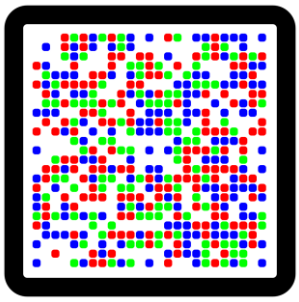
QR-codes kunnen echter ook gebruikt worden als bijkomende factor in een multifactorauthenticatieproces.

Figuur 22: Een QR-code [19]

8.6 CrontoSign

CrontoSign is een verificatiemethode die door sommige banken gebruikt wordt voor de verificatie van betalingen. De gebruiker moet een gekleurde mozaïekcode scannen met een app op zijn mobiele telefoon. De gegevens uit de mozaïekcode worden ontsleuteld en samen met een *transaction authentication number* (TAN) getoond op de telefoon. Wanneer de gebruiker de TAN-code ingeeft op zijn pc, kan hij de transactie voltooien.

Andere banken gebruiken hetzelfde systeem onder de naam photoTAN.



Figuur 23: Een CrontoSign- of photoTAN-code [20]

8.7 Security tokens

Security tokens zijn fysieke toestellen die een wachtwoord bevatten.

8.7.1 Niet-geconnecteerde en geconnecteerde tokens

8.7.1.1 Niet-geconnecteerde tokens

Niet-geconnecteerde tokens bevatten meestal een ingebouwd scherm om het wachtwoord af te lezen en soms ook een ingebouwd toetsenbord. De meest populaire niet-geconnecteerde tokens gebruiken *one-time passwords* als authenticatiemiddel. Voorbeelden van niet-geconnecteerde tokens zijn de kaartlezers die gebruikt worden voor internetbankieren en RSA SecurID-tokens.

Figuur 24: Kaartlezers voor internetbankieren [21]

Figuur 25: Een RSA SecurID-token [22]

De laatste jaren is het gebruik van softwaretokens in opmars. Softwaretokens emuleren niet-geconnecteerde hardwaretokens. Deze tokens nemen dikwijls de vorm aan van een app op een mobiele telefoon als Twilio Authy, Google Authenticator of Microsoft Authenticator.

8.7.1.2 Geconnecteerde tokens

Geconnecteerde tokens zijn tokens die verbonden moeten worden met het toestel waarop de gebruiker zich probeert te authenticeren. Wanneer die verbinding gemaakt wordt, geeft het token de authenticatiecode automatisch door aan het toestel waarmee geauthenticeerd wordt.

Geconnecteerde tokens worden ook wel *security keys* genoemd.

Voorbeelden van geconnecteerde tokens zijn chipkaarten, simkaarten, USB-tokens en FIDO-tokens als Yubico Yubikey en Google Titan.

De connectie kan zowel een fysieke connectie als een contactloze connectie zijn. Contactloze tokens worden veel gebruikt voor *keyless entry* en bij elektronische betaalsystemen.

Figuur 26: Een chipkaart [23]

8.7.2 SOORTEN TOKENS

8.7.2.1 Static password token

Een *static password token* is een token dat één enkel wachtwoord bevat. Dit wachtwoord is niet zichtbaar voor de gebruiker en wordt bij elke authenticatie verzonden naar de authenticatieserver.

8.7.2.2 One-time password (OTP) token

Een *one-time password (OTP) token* is een token dat een nieuw wachtwoord aanmaakt wanneer er op een knop gedrukt wordt of wanneer een bepaald tijdsperiode passeert.

8.7.2.3 Public key infrastructure (PKI) token

Een *public key infrastructure (PKI) token* is een token dat een private sleutel en digitale certificaten bevat. Digitale certificaten worden uitgereikt en beheerd door een certificaatauthoriteit. De certificaatauthoriteit waarborgt de authenticiteit van het certificaat. De server zal een gebruiker authenticeren door de certificaten van een vertrouwde certificaatauthoriteit te aanvaarden.

De server kan dan de bijbehorende publieke sleutel gebruiken om de gebruiker te authenticeren.

X.509 is de meest gebruikte standaard voor digitale certificaten.

Figuur 27: PKI-tokens [25]

8.7.2.4 FIDO security keys

FIDO security keys maken gebruik van de U2F- en/of de FIDO2-specificatie. Yubico Yubikey, Google Titan en Thetis Fido U2F zijn voorbeelden van FIDO-tokens.

Figuur 28: De Yubikey4

8.8 BIOMETRISCHE GEGEVENS

Biometrische verificatie maakt gebruik van biometrische gegevens om gebruikers te authenticeren. De meest gebruikte biometrische gegevens zijn vingerafdrukken, retina- of irisscans, stemherkenning en gezichtsherkenning [26].

Bij de registratie scant de gebruiker het gevraagde biometrische item. De server slaat de biometrische gegevens op. De gebruiker kan zich dan authenticeren door het biometrische item opnieuw te scannen en aan te bieden aan de server.

9 VOOR- EN NADELEN VAN DE VERSCHILLENDE VERIFICATIEMETHODEN

De ideale bijkomende verificatiemethode bestaat niet. Elke verificatiemethode heeft zijn eigen voor- en nadelen. Het hangt dan ook sterk af van de organisatie wat de beste keuze is.

Toch zijn er bepaalde methodes die in het algemeen beter zijn dan andere methodes. Er moet wel altijd een afweging gemaakt worden tussen veiligheid, gebruiksvriendelijkheid en implementatiekost. Onderstaande tabel geeft een overzicht van hoe de meest gebruikte verificatiemethoden scoren op deze drie criteria.

TABEL 2: VERGELIJKING TUSSEN DE VERSCHILLENDE BIJKOMENDE VERIFICATIEMETHODEN

	VEILIGHEID	GEBRUIKS-VRIENDELIJKHEID	KOSTPRIJS EN FLEXIBILITEIT
TELEFOONGESPREK	~	-	-
SMS	-	~	-
E-MAIL	-	-	+
PUSHNOTIFICATIE	+	++	+
QR-CODE	~	+	--
CRONTOSIGN/PHOTOTAN	+	+	--
HARDWARE OTP-TOKEN	+	-	--
SOFTWARE OTP-TOKEN	~	~	++
PKI-TOKEN	++	-	-
FIDO SECURITY KEY	++	~	-
BIOMETRISCH VIA TELEFOON	+	++	-

9.1 VEILIGHEID

Verificatie via sms en verificatie via e-mail zijn waarschijnlijk de minst veilige methoden [28]. E-mails worden niet altijd via veilige protocollen verzonden. Niet-geëncrypteerde e-mailberichten kunnen dan ook vrij makkelijk onderschept worden. Het is bovendien mogelijk dat de gebruiker hetzelfde wachtwoord gebruikt voor zijn e-mailadres als voor de dienst waarbij hij zich wil authenticeren. In dat geval is er niet echt sprake van een tweede factor. Ook sms-verkeer kan in principe onderschept worden. Verificatie via sms is bovendien kwetsbaar voor *SIM swap attacks* of andere vormen van *social engineering* waarbij aanvallers zich richten op de telefoonbedrijven om toegang te krijgen tot telefoonnummer van de gebruiker.

De meest veilige methoden zijn de methoden die gebruik maken van een private sleutel die bijgehouden wordt op een hardware token zoals de PKI-tokens en de FIDO-tokens.

OTP-tokens bevinden zich op het vlak van veiligheid tussen *out-of-band*mechanismen en methodes die gebruik maken van een asymmetrisch sleutelpaar. Het OTP wordt in *plaintext* verzonden en is dus kwetsbaar voor *man-in-the-middle attacks*. De code is bovendien meestal zichtbaar op het scherm wanneer ze door de gebruiker ingegeven wordt waardoor ze kwetsbaar wordt voor *eavesdropping*. Softwaretokens zijn iets minder veilig dan hardwaretokens. Bij een telefoon is het vaak makkelijk te achterhalen wie de eigenaar van het toestel is. Ook inloggegevens als gebruikersnamen of e-mailadressen zijn dikwijls makkelijk op te sporen. Wanneer de dief toegang krijgt tot de *authenticator app* is het extraheren van het gedeeld geheim niet moeilijk.

Pushnotificaties zijn over het algemeen een vrij veilige verificatiemethode. Bij pushnotificaties is er wel het risico dat de gebruiker geneigd kan zijn om de informatie in de notificatie te negeren en zonder nadenken op goedkeuren te klikken. Pushnotificaties kunnen bovendien ook onderschept worden en er zijn gevallen bekend waarbij pushservices gecompromitteerd zijn.

Biometrische verificatie is in principe een heel veilige verificatiemethode. Het fysieke aspect van biometrische authenticatie maakt het bijna onmogelijk voor hackers om deze gegevens te repliceren. Bij biometrische verificatie speelt echter ook het aspect van de privacy. Het opslaan van biometrische gegevens op een server brengt voor de gebruiker een apart veiligheidsrisico mee.

9.2 GEBRUIKSVRIENDELIJKHEID

Het grote nadeel op het vlak van gebruiksvriendelijkheid van specifieke hardwaretokens zoals hardware OTP-tokens, FIDO-tokens en PKI-tokens is dat de gebruiker het token altijd bij zich moet hebben. Verder moeten deze toestellen ook voorzien worden van een batterij, waarvan de levensduur beperkt is.

Weliswaar geldt hetzelfde voor systemen die gebruik maken van de mobiele telefoon van de gebruiker, maar de meeste gebruikers dragen hun telefoon sowieso al bij zich.

Pushnotificaties en biometrische verificatie zijn de meest gebruiksvriendelijke systemen. De verificatie verloopt bij deze methoden erg snel en er is nauwelijks input van de gebruiker vereist.

*Out-of-band*methoden als sms en e-mail hebben als nadeel dat het soms enige tijd duurt voordat de sms of de e-mail met het OTP ontvangen wordt. In dat geval is het bovendien mogelijk dat de termijn van het OTP verstreken is en dat de gebruiker een nieuw verificatiebericht moet aanvragen. E-mailberichten hebben het bijkomend probleem dat ze soms in de spamfolder terechtkomen. Bij *out-of-band*methoden is er ook altijd de noodzaak om verbonden te zijn met het secundaire communicatiekanaal.

In het geval van e-mail en software OTP-tokens moet de gebruiker een aparte app openen om toegang te krijgen tot zijn OTP. Bij *out-of-band*mechanismen en OTP-tokens is er ook de extra stap van het ingeven van het OTP.

9.3 IMPLEMENTATIEKOST EN FLEXIBILITEIT

Op het vlak van implementatiekost en flexibiliteit komen de software OTP-tokens als winnaar naar voren. Het grote voordeel van softwaretokens is de flexibiliteit. Een nieuw token kan gemakkelijk en gratis digitaal aangemaakt worden. Ook het verwijderen van een token is heel eenvoudig.

Ook e-mailverificatie en pushnotificaties zijn vrij makkelijk en goedkoop te implementeren.

Hardware tokens zijn vaak een dure manier om authenticatie te implementeren. Voor elke gebruiker moet een apart token voorzien worden. De toestellen moeten bovendien vaak vervangen worden, wat een extra kost met zich meebrengt. In het geval van PKI-tokens en FIDO-tokens wordt meestal van de gebruiker verwacht om zijn eigen token aan te schaffen.

Biometrische verificatie is vooralsnog een vrij dure methode, maar wordt meer en meer robuust en betaalbaar. Een nadeel is dat biometrische authenticatie niet de meest accurate vorm van authenticatie is. Er is een risico op *false positives* en *false negatives*. Biometrische gegevens zijn ook enigszins veranderlijk over de tijd. Een ander nadeel is dat biometrische gegevens niet vervangbaar zijn. Wanneer een hacker erin slaagt om een vingerafdruk te verkrijgen, kan er onmogelijk een nieuwe vinger aangemaakt worden om aan de gehackte gebruiker te geven. De enige optie is dan de authenticatiemethode zelf te wijzigen.

10 CONCLUSIE

Authenticatie is altijd een afweging tussen veiligheid, gebruiksvriendelijkheid en kostprijs. Uiteindelijk werd voor het controlepaneel van Level27 gekozen voor een softwarematig *one-time password* als bijkomende verificatiemethode. De grote mate van flexibiliteit en de eenvoud van de implementatie gaf daarbij de doorslag.

BRONNENLIJST

[1] R. A. Grimes, Hacking Multifactor Authentication, Indianapolis, Indiana: John Wiley & Sons, Inc., 2021.

[2] Z. Choi en D. Zage, „Addressing insider threat using “where you are” as fourth factor authentication,” in *IEEE International Carnahan Conference on Security Technology (ICCST)*, Newton, 2012.

[3] C. Burns, „Is location an Authentication Factor?,” 26 08 2021. [Online]. Available: https://medium.com/os-techblog/is-location-an-authentication-factor-9ed33d633993. [Geopend 8 april 2022].

[4] R. Dias, „The 5 Factors of Authentication,” 08 12 2017. [Online]. Available: https://dojowithrenan.medium.com/the-5-factors-of-authentication-bcb79d354c13. [Geopend 8 april 2022].

[5] Savvy Security, „2 Factor Authentication vs 2 Step Verification: What’s The Difference?,” 09 09 2021. [Online]. Available: https://cheapsslsecurity.com/blog/2-factor-authentication-vs-2-step-verification-whats-the-difference/. [Geopend 6 april 2022].

[6] Y. Wilson en A. Hingnikar, Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0, San Fransisco: Apress, 2019.

[7] J. Richer en A. Sanso, OAuth2 in Action, Shelter Island: Manning Publications Co., 2017.

[8] I. Indu, P. R. Anand en V. Bhaskar, „Identity and access management in cloud environment: Mechanisms and challenges,” *Engineering Science and Technology, an International Journal,* vol. 21, nr. 4, pp. 574-588, 2018.

[9] Y.-S. Lee, H. T. Lim en H. Lee, „A study on efficient OTP generation using stream cipher with random digit,” in *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on Advanced communication technology*, Gangwon-Do, 2010.

[10] M. Schwartz en M. Machulak, Securing the Perimeter, Apress, 2018.

[11] T. Reinert, „The Developer’s Guide to One-Time Passwords (OTPs),” 28 06 2021. [Online]. Available: https://workos.com/blog/guide-to-one-time-passwords-otps. [Geopend 8 april 2022].

[12] „FIDO Members,” [Online]. Available: https://fidoalliance.org/members/. [Geopend 2 mei 2022].

[13] „Can I use webauthn?," [Online]. Available: https://caniuse.com/?search=webauthn. [Geopend 2 mei 2022].

[14] FIDO Alliance, „FIDO2: Web Authentication (WebAuthn)," [Online]. Available: https://fidoalliance.org/fido2/fido2-web-authentication-webauthn/. [Geopend 3 mei 2022].

[15] C. Cimpanu, „W3C finalizes Web Authentication (WebAuthn) standard," 4 maart 2019. [Online]. Available: https://www.zdnet.com/article/w3c-finalizes-web-authentication-webauthn-standard/. [Geopend 2 mei 2022].

[16] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache en O. Ranen, „HOTP: An HMAC-Based One-Time Password Algorithm," december 2005. [Online]. Available: https://www.ietf.org/rfc/rfc4226.txt. [Geopend 5 mei 2022].

[17] D. M'Raihi, S. Machani, M. Pei en J. Rydell, „TOTP: Time-Based One-Time Password Algorithm," mei 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6238. [Geopend 5 mei 2022].

[18] D. M'Raihi, J. Rydell, S. Bajaj, S. Machani en D. Naccache, „OCRA: OATH Challenge-Response Algorithm," 8 maart 2010. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6287. [Geopend 6 mei 2022].

[19] Bobmath, Artist, *QR Code Model 1 Example.* [Art]. 2013.

[20] G. Fischer en Calypso10, Artists, *PhotoTAN mit Orientierungsmarkierungen.* [Art]. 2013.

[21] M0tty, Artist, *Différents modèles de lecteurs de cartes bancaires.* [Art]. 2015.

[22] A. Klink, Artist, *An RSA SecurID SID800 token with USB connector.* [Art]. 2008.

[23] Pagelmp, Artist, *A sample RuPay Debit Card.* [Art]. 2016.

[24] Aldolat, Artist, *Yubikey4.* [Art]. 2020.

[25] Kharitonov, Artist, *JaCarta smart card, three different USB tokens & microSD token.* [Art]. 2014.

[26] N-Able, „Common Network Authentication Methods," 24 04 2019. [Online]. Available: https://www.n-able.com/blog/network-authentication-methods. [Geopend 5 april 2022].

[27] A. Waugh, „Which MFA methods should you use?," 15 03 2021. [Online]. Available: https://pushsecurity.com/blog/which-mfa-methods-should-you-use/. [Geopend 1 april 2022].

[28] C. Mulliner, R. Borgaonkar en J.-P. Seifert, „SMS-based One-Time Passwords: Attacks and Defense," Technische Universität Berlin, Berlin, 2014.