
Inlogsysteem

Hoe maak je een veilig inlogsysteem?



Profielwerkstuk

Opleiding: VWO Jan van Egmond College

Leerling: Toine van Wonderen (22079), A5C
Jesse Blom (22798), A5E

Begeleider: Informatica, Menno Merlijn (MLI) 24-6-2020



Jan van Egmond Lyceum
School voor HAVO, VWO (Atheneum, Gymnasium)



ScholenGroep
Primerendse



Voorwoord

Beste lezer,

Wij zijn Jesse en Toine en dit is ons PWS met als onderwerp: de veiligheid van inlogsystemen. We schrijven dit PWS als eindopdracht voor onze middelbare school en we zijn op 24 juni 2020 begonnen en hebben het afgerond op 18 januari 2021.

We zitten op het Jan van Egmond in 6 VWO en we maken allebei al websites voor onze hobby sinds ongeveer de tweede klas. In de derde klas hebben we al een website samen gemaakt en omdat dat zo leuk was wilden we deze keer graag weer een project samendoen wat als onderwerp informatica had. We wilden een onderwerp waar we veel van konden leren en ook iets zelf konden programmeren. Hierdoor zijn we op het onderwerp inlogsystemen gekomen. We hadden eerder al 'eenvoudige' inlogsystemen gebouwd en het leek ons erg leuk om ons er meer in te verdiepen om veiligere inlogsystemen te kunnen bouwen. Tijdens dit project hebben we veel onderzoek gedaan en veel geleerd over hoe en waarom een inlogstelsel nou precies veiliger wordt gemaakt en we hebben dit ook zoveel mogelijk terug laten komen in onze praktische opdracht.

We willen ook graag onze begeleider meneer Merlijn bedanken die ons goed geholpen heeft met dit PWS. We kregen altijd goede hulp en tijdens het schrijven van ons PWS had meneer Merlijn goede feedback en konden we daardoor ons PWS goed verbeteren. Ook als we zelf een vraag hadden wilde Merlijn ons altijd helpen. Bijvoorbeeld toen we tegen problemen aanliepen in onze praktische opdracht hielp meneer Merlijn altijd, waardoor alles uiteindelijk is gelukt.

Tot slot vonden we de samenwerking met elkaar ook heel fijn. We hadden altijd een goede taakverdeling en hadden altijd veel zin om meer te leren over dit onderwerp. We konden elkaar altijd goed feedback geven om elkaar te helpen, waardoor we een mooi PWS gemaakt hebben.

Ik wens u veel leesplezier toe.

Jesse Blom en Toine van Wonderen
18-1-2021



Inhoud

INLEIDING	5
PROBLEEMSTELLING.....	5
HYPOTHESE.....	6
1 DEELVRAAG 1: WAT IS EEN INLOGSYSTEEM?	7
1.1 Registratie	7
1.2 Identificatie	7
1.3 Authenticatie.....	7
1.4 Autorisatie & personalisatie.....	8
2 DEELVRAAG 2: HOE KUNNEN ACCOUNTS OVERGENOMEN WORDEN?.....	9
2.1 Hacken van de database	9
2.2 Rainbow table.....	9
2.3 SQL-injectie	9
2.4 Buffer overflow.....	11
2.5 Credential Stuffing	12
2.6 Bots – Brute-Force Attacks.....	12
2.7 Phishing.....	13
2.8 Keylogger.....	13
3 DEELVRAAG 3: WAT VOOR OPLOSSINGEN ZIJN ER OM EEN INLOGSYSTEEM VEILIGER TE MAKEN?	14
3.1 Veiligheid voor wachtwoorden.....	14
3.2 Two-Factor Authentication	14
3.2.1 E-mail.....	15
3.2.2 SMS.....	15
3.2.3 Authenticator app	16
3.2.4 Yubikey.....	17
3.2.5 Wat is de veiligste 2FA methode?.....	17
3.3 reCAPTCHA.....	18
3.4 Vingerscan:.....	18
3.5 Gezichtsherkenning:	20
4 DEELVRAAG 4: DE INLOGGEGEVENS VAN GEBRUIKERS MOETEN WORDEN OPGESLAGEN IN EEN DATABASE, HOE GAAT DIT VEILIG?	21
4.1 Wat is een database?.....	21
4.2 Hoe maken we deze database veilig?	22
4.2.1 Database firewall	22
4.2.2 Encryptie	23
4.2.3 Speciale tekens.....	23
4.2.4 Prepare statements	24
4.3 Wie kan de data in de database zien?.....	24



5 DEELVRAAG 5: KUN JE MET ÉÉN INLOGMIDDEL OP MEERDERE SITES INLOGGEN EN IS DAT PRIVACYGEVOELIG?	25
5.1 Hoe werkt inloggen met een account van Google?	25
5.2 Voor- en nadelen	26
5.3 Inloggen met IRMA	27
6 PRAKTISCH GEDEELTE (HET INLOGSYSTEEM ZELF MAKEN).....	29
7 SAMENVATTING.....	33
8 KLOPT ONZE HYPOTHESE	34
9 CONCLUSIE.....	34



INLEIDING

Als wij denken aan een veilig inlogstelsel, denken we aan een inlogstelsel waar de gebruiker gecontroleerd wordt door het stelsel of de gebruiker de persoon is wie hij zegt dat hij is. Wij denken dat dit gebeurt doordat de gebruiker een uniek e-mailadres heeft met een bijpassend wachtwoord die alleen de juiste persoon achter het account weet. Voor extra beveiliging denken wij dat er veel gebruik gemaakt wordt van 2-factor authenticatie. Ook verwachten wij bij een veilig inlogstelsel dat de database geen zwakheden heeft waarmee een hacker in het stelsel zou kunnen komen en data kan zien en gebruiken.

Bedrijven hebben een grote verantwoordelijkheid voor het waarborgen van de veiligheid van de data van de gebruikers. Omdat grote bedrijven meer klanten en geld hebben, denken wij dat grote bedrijven een veiliger inlogstelsel hebben dan kleinere bedrijven. Een voorbeeld hiervan is de ING app. Als je de app wilt installeren en koppelen aan jouw bankrekening moet je naast het inloggen ook een code die je via een sms hebt ontvangen invullen. Kleinere bedrijven zullen ook een inlogstelsel hebben, maar die zullen eenvoudiger zijn, bijvoorbeeld zonder 2-factor authenticatie.

PROBLEEMSTELLING

De probleemstelling van ons PWS is "Hoe maak je een veilig inlogstelsel?"

Om deze vraag te kunnen beantwoorden hebben wij eerst een aantal deelvragen gemaakt om meer informatie te vinden hoe je een veilig inlogstelsel kunt maken. Iedere deelvraag is in een apart hoofdstuk opgenomen.

1. Wat is een inlogstelsel?
2. Hoe kunnen accounts overgenomen worden?
3. Wat voor oplossingen zijn er om een inlogstelsel veilig te maken
4. De inloggegevens moeten worden opgeslagen in een database, hoe gaat dit veilig?
5. Kun je met één inlogmiddel op meerdere sites inloggen en is dat privacygevoelig?

Vervolgens hebben we in het praktische gedeelte zelf een inlogstelsel gemaakt die zoveel mogelijk gebruik maakt van wat we in de deelvragen hebben gevonden.

In hoofdstuk 6 is beschreven of het ons gelukt is om een veilig inlogstelsel te maken.



HYPOTHESE

Wij verwachten een goed functionerend inlogstelsel te kunnen maken met een eigen gemaakte database. We willen bij het registreren een naam, e-mail, telefoonnummer en wachtwoord vragen van de gebruiker. Voor extra veiligheid willen we als 2-factor authenticatie een sms sturen naar de gebruiker, zodat de gebruiker 2 apparaten nodig heeft om in te loggen. Als de gebruiker weer wil inloggen moet de gebruiker een e-mailadres en wachtwoord invullen en krijgt de gebruiker weer een sms met een code die hij moet invullen.

Het kan voorkomen dat de gebruiker zijn wachtwoord vergeten is. Als dat het geval is klikt de gebruiker op de link wachtwoord vergeten. Dan vult de gebruiker zijn e-mail in en wordt er een e-mail verstuurd. Als de gebruiker dan op de link in de e-mail klikt wordt de gebruiker weer teruggestuurd naar de site en kan de gebruiker zijn wachtwoord veranderen.

We verwachten niet dat ons inlogstelsel net zo veilig is als bijvoorbeeld Google aangezien Google veel meer geld heeft om veiligere databases en andere technieken te kopen.



1 DEELVRAAG 1: WAT IS EEN INLOGSYSTEEM?

1.1 Registratie

Bij een inlogstelsysteem draait het om te weten wie er wil inloggen. De site moet weten wie de gebruiker achter de computer is, zodat de site de juiste data koppelt aan de juiste persoon. De site laat de gebruiker bij het registreren meestal een naam, emailadres en wachtwoord invullen, zodat de site jou kan opslaan en weer kan herkennen als je opnieuw wilt inloggen.

Als een gebruiker een emailadres invult, wilt het programma dat de gebruiker kan aantonen dat dat e-mailadres daadwerkelijk van hem is. Ook kan zo gecontroleerd worden of de gebruiker zich niet voordoet als een werknemer van een bepaald bedrijf door een mailadres van dat bedrijf te gebruiken. Voorbeeld: Stel er is een gebruiker die op Facebook een account aanmaakt met het e-mailadres van een bekend bedrijf. Dan wil Facebook zeker weten dat het een werknemer van dat bedrijf is. Anders kan de persoon zich voordoen alsof hij werknemer is en kwaadwillende dingen met dat account doen, zoals mensen bedreigen. De rest van de wereld denkt dan dat dat bericht vanuit dat bekende bedrijf komt, wat eigenlijk gewoon een ander persoon is. Een manier om deze fraude tegen te gaan en meer zekerheid te krijgen dat het emailadres bij de juiste persoon hoort, is door een e-mail te verzenden naar het email adres dat geregistreerd werd. Als de gebruiker toegang tot dat mailadres heeft, kan de persoon op de link in de email klikken of de code invoeren om zijn account te activeren.

Soms vragen sites meer data van de gebruiker voor de veiligheid of zodat de site makkelijker de gebruiker kan bereiken, bijvoorbeeld een telefoonnummer, zodat de site gebruik kan maken van 2-factor authenticatie of zodat het bedrijf de gebruiker kan bellen.

Bij het registreren is het ook belangrijk dat als meerdere gebruikers dezelfde naam hebben het inlogstelsysteem toch de gebruikers kan onderscheiden. Dit gebeurt door bij de registratie één uniek veld te gebruiken. Vaak is dit een e-mailadres omdat een e-mail ook uniek is.

[Bron: <https://nl.wikipedia.org/wiki/Registratie>]

1.2 Identificatie

Als iemand wil inloggen op de site waarop hij zich geregistreerd heeft moet hij eerst invullen wie hij is door zijn gebruikersnaam in te vullen. Dit heet identificatie.

Een gebruikersnaam is niet altijd geheim, bijvoorbeeld als de gebruikersnaam een e-mailadres is. Iedereen die jouw e-mailadres kent, kan dit invullen. Het systeem weet nog niet zeker wie er probeert in te loggen. Dat weet het systeem pas na de authenticatie.

[Bron: [https://nl.wikipedia.org/wiki/Identificatie_\(informatica\)](https://nl.wikipedia.org/wiki/Identificatie_(informatica))]

1.3 Authenticatie

Authenticatie is bewijzen dat je werkelijk de persoon bent als dat je bij de identificatie zegt dat je bent. Er zijn vele manieren bedacht, om de gebruiker te laten bewijzen dat hij zichzelf is. Een simpele manier is dat de gebruiker het geheime wachtwoord invult die hij zelf bij de registratie heeft opgegeven.



Om meer zekerheid te krijgen dat de gebruiker is wie hij zegt dat hij is, kan er gebruik gemaakt worden van 2-factor authenticatie. Een voorbeeld hiervan is dat de gebruiker tijdens het inloggen een SMS ontvangt met een code die hij weer invult in het inlogstelsel om verder te kunnen op de website. Hierdoor bewijst de gebruiker dat hij de persoon is die hij zegt dat hij is omdat hij de telefoon bezit die is opgegeven tijdens het registreren. Als iemand anders dit probeert krijgt hij geen SMS op zijn telefoon, omdat hij het verkeerde nummer heeft en dus niet de persoon is die hij voordoet. Een ander voorbeeld is dat de gebruiker een e-mail ontvangt met een link. Als de gebruiker klikt op de link wordt hij doorgestuurd naar de pagina waarvoor je ingelogd moet zijn.

[Bron: <https://nl.wikipedia.org/wiki/Authenticatie>]

[Bron: <https://www.comptest.nl/nl/knowledge-platform/hot-topics-explained/wat-is-authenticatie/>]

1.4 Autorisatie & personalisatie

Maar waarom zouden bedrijven een inlogstelsel willen waarmee de gebruiker zich kan identificeren? Dit willen bedrijven omdat ze vervolgens kunnen autoriseren en personaliseren.

Met autoriseren mag een gebruiker meer of minder dingen doen op de site. Bijvoorbeeld het aanpassen van de teksten op een site mag alleen door de site beheerder worden gedaan. Het plaatsen van producten in een winkelmandje mag iedereen die is ingelogd of producten kunnen alleen beschikbaar zijn voor gebruikers van 18 jaar of ouder, bijvoorbeeld alcohol.

Met personalisatie heeft de gebruiker bijvoorbeeld zijn eigen winkelmandje. Als een gebruiker inlogt op een webshop en daarna iets in zijn winkelmandje stopt, dan kan de site een link leggen tussen het product en het winkelmandje van de gebruiker. Als de gebruiker opnieuw inlogt, dan laat de site het persoonlijke winkelmandje zien van de gebruiker.

De site kan zien wat er allemaal in het winkelmandje gestopt is. De webshop weet wat voor kleding de persoon leuk vindt en kunnen ze daarop inspelen door andere soortgelijke producten te laten zien.

Om te personaliseren hoeft een gebruiker niet altijd in te loggen. Personaliseren kan ook met cookies. De reclames die jij ziet, komt doordat wat jij bekijkt in cookies is opgeslagen. Zo worden er specifieke reclames gemaakt naar de interesses van de persoon. Hier gaan wij verder niet op in.

[Bron: <https://nl.wikipedia.org/wiki/Autorisatie>]



2 DEELVRAAG 2: HOE KUNNEN ACCOUNTS OVERGENOMEN WORDEN?

Ondanks alle veiligheidsmaatregelen proberen sommige mensen toch inloggegevens te bemachtigen van andere gebruikers om zichzelf voor te doen als iemand anders. Dit doen ze bijvoorbeeld om geld te stelen. Er zijn veel verschillende manieren om de inloggegevens te kunnen bemachtigen.

2.1 Hacken van de database

Een manier om inloggegevens te kunnen bemachtigen is door middel van het hacken van de database waar de inloggegevens zijn opgeslagen. Als dit lukt, krijgt de hacker toegang tot data in de database. Voor extra veiligheid worden wachtwoorden en belangrijke data van de gebruikers gehashed. Dat betekent dat het echte wachtwoord is omgezet naar een reeks symbolen die moeilijk achterhaalt en teruggezet kunnen worden door de hacker. De reeks symbolen wordt gemaakt door een hash algoritme. Van deze wachtwoorden wordt vaak alleen de one way hash opgeslagen. Dat betekent dat de hash niet teruggehaald kan worden naar normale tekst. De hash kan nog wel vergeleken worden met het ingevulde wachtwoord om te kijken of het wachtwoord dat is ingevuld juist is. Hash algoritmes voor wachtwoorden zijn: bcrypt, scrypt en PBKDF.

[Bron: <https://nl.wikipedia.org/wiki/Hashfunctie>]

[Bron: <https://allesovercrypto.nl/blog/hash-simpele-uitleg>]

2.2 Rainbow table

Om gehashte wachtwoorden toch om te zetten gebruiken hackers een Rainbow Table. Een Rainbow Table bevat een lijst met standaard hashes die gehashte wachtwoorden om kunnen zetten naar normale tekst. Op deze manier kunnen de wachtwoorden worden gebruikt voor het inloggen als iemand anders. De Rainbow Table kost veel rekenkracht om op te halen, maar als het is opgehaald, kan het wachtwoord heel snel omgezet worden naar een normaal wachtwoord. Een voorbeeld hiervan is stel een gebruiker voert een 'x' wachtwoord in. Deze 'x' wordt willekeurig vermenigvuldigd en laat er nog een reeks aan getallen en letters achteraankomen. Het wachtwoord is nu gehasht. Wanneer de hacker rainbow tables bemachtigt en gebruikt, wordt die hash code voor alle getallen bijvoorbeeld zichtbaar. Getal nummer 49 staat dan bijvoorbeeld voor 0348. Wanneer de hacker 0348 in de hash code ziet staan, weet hij dat nummer 49 in de gebruiker zijn wachtwoord staat.

[Bron: https://nl.wikipedia.org/wiki/Rainbow_table]

2.3 SQL-injectie

Een andere manier om data zonder toestemming uit een database te halen is via SQL-injecties. Met SQL-injecties maakt de hacker connectie met de database via SQL-query's. De hacker schrijft dan een query die data ophaalt uit de database, maar door de voorwaarden in de query te veranderen kan de hacker naast zijn eigen gegevens ook data van andere gebruikers ophalen.

```
SELECT * FROM persoon WHERE achternaam = 'Jansen' OR 'a' = 'a'
```



Deze query is een voorbeeld van een SQL-injectie. Doordat in de voorwaarden staat dat 'a' = 'a' voldoet dit altijd en zal de query alle data ophalen in die table, mits de website geen beveiliging heeft tegen SQL-injecties.

Een andere manier om data op te halen is door gebruik te maken van een apostrof.

```
SELECT * FROM fruit WHERE naam = 't appeltje'
```

In deze query zie je dat de waarde van naam 't appeltje moet zijn, maar de functie van een apostrof is het beginnen en afsluiten van een string en daarom moet je bij een goed werkende query een / voor de apostrof zetten of de apostrof verdubbelen, zodat de apostrof als een string gelezen wordt. Wat er nu gebeurd is dat de apostrof de string afsluit en de query alles uitvoert voor de apostrof wat niks oplevert. Daarna voert de query alles achter de apostrof uit, in dit geval t appeltje, wat een foutmelding veroorzaakt.

```
SELECT * FROM bank WHERE naam = ''; drop table bank--
```

```
SELECT * FROM fruit WHERE naam = ''; insert into fruit values( 4, 'peer' )--
```

Dit zijn nog 2 query's die gebruik maken van de apostrof. In deze query's sluit de hacker de query's af door middel van de apostrof en de punt komma. Daarachter zet de hacker een nieuwe query waardoor de hacker de database kan aanpassen.

Ook zie je soms dat in de URL van de website een deel van de query staat die de hacker kan aanpassen om data op te halen.

```
https://insecure-website.com/products?category=Gifts
```

Deze URL voert deze query uit:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

Deze query zegt dat de site alle kolommen wil ophalen in de table products in de category gifts die wel released zijn. Maar als de hacker ook producten wil ophalen die unreleased zijn kan de hacker gebruik maken van -- die werken als comments in SQL-query's. De hacker zal dan de URL aanpassen:

```
https://insecure-website.com/products?category=Gifts'--
```

Dan zal de query er zo uitzien:

```
SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1
```

Hierdoor zal de AND released = 1 als comment gelezen worden en zal de site alle producten in de category gifts ophalen. Ook kan je deze query nog combineren met category = 'gifts' OR 1=1, zodat de site alle products ophaalt.

[Bron: <https://portswigger.net/web-security/sql-injection>]

[Bron: <https://nl.wikipedia.org/wiki/SQL-injectie>]



2.4 Buffer overflow

Buffer overflow is ook een manier om data uit de database te bemachtigen. Bij buffer overflow is het de bedoeling dat data in buffers worden overschreden, waardoor er een foutmelding ontstaat en de hacker data bemachtigt. Een buffer is een opslagplaats voor data als de data verplaatst moet worden van de ene naar de andere plek, een geheugen. Buffers hebben een bepaalde grootte en als hackers gebruik willen maken van buffer overflows, zenden hackers data naar de website die groter is dan de buffer. Hierdoor zal er andere data overschreden worden, waardoor bijvoorbeeld het return-adres veranderd en het programma anders zal reageren. Het programma zal bijvoorbeeld crashen of andere data geven aan de hacker. Op deze manier bemachtigt de hacker meer data uit de database. Veel codetalen hebben ingebouwde veiligheid mechanismen om buffer overflows te voorkomen. Echter zijn C en C++ wel sneller vatbaar voor buffer overflows, omdat die geen ingebouwde veiligheid mechanismen hebben.

Voorbeeld:

In dit voorbeeld zijn er 2 variabelen in het geheugen opgeslagen. Op dit moment is variabele A leeg en heeft een grootte van 8 bytes. Variabele B is echter wel vol en heeft een lengte van 2 bytes.

variable name	A								B	
value	[null string]								1979	
hex value	00	00	00	00	00	00	00	00	07	BB

Vervolgens veranderd de hacker de waarde van A door het woord 'excessiv' in te voeren. Dit woord heeft een lengte van 9 bytes, maar A heeft een grootte van 8 bytes. Hierdoor zal de 9de byte B overschrijden en krijgt B de waarde '25856'. Op deze manier kan de hacker dus data veranderen en foutmelding veroorzaken.

variable name	A								B	
value	'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	25856	
hex	65	78	63	65	73	73	69	76	65	00

[Bron: <https://www.sciencedirect.com/topics/computer-science/buffer-overflow>,
<https://nl.wikipedia.org/wiki/Bufferoverflow>]

[Bron:

https://en.wikipedia.org/wiki/Buffer_overflow#:~:text=A%20buffer%20overflow%20occurs%20when,fits%20within%20the%20destination%20buffer]

[Bron: <https://www.imperva.com/learn/application-security/buffer-overflow/>]



2.5 Credential Stuffing

Als een hacker een wachtwoord van iemand heeft, dan kan de hacker niet alleen inloggen op de website van waar de inloggegevens zijn, maar ook bij andere websites waar de gebruiker eventueel een account heeft. Deze manier heet Credential Stuffing. Dit is heel efficiënt, omdat je de gegevens al hebt. Je moet alleen nog de juiste websites vinden waar die gebruiker ook een account heeft met dezelfde gegevens. Dit werkt niet als de gebruiker verschillende wachtwoorden gebruikt voor de verschillende websites.

Een hacker hoeft de wachtwoorden niet altijd zelf bemachtigen. Er worden ook lijsten met gebruikersnamen en wachtwoorden verkocht door andere hackers. Hierdoor kan een datalek op 1 website ook heel veel impact hebben op veel meer websites.

[Bron: <https://www.computable.nl/artikel/blogs/security/7082833/5260614/credential-stuffing-is-het-nieuwe-phishing.html>]

[Bron: https://en.wikipedia.org/wiki/Credential_stuffing]

2.6 Bots – Brute-Force Attacks

Brute-force attacks zijn aanvallen waardoor een bot of software iedere mogelijkheid als wachtwoord probeert. Dat zijn heel veel mogelijkheden.

Omdat er zoveel mogelijkheden zijn kost het wel heel veel rekenkracht van de computer om alle mogelijkheden uit te voeren en neemt het dus ook tijd in. Hoe beter de CPU en GPU hoe sneller de Brute-force attack.

Het probleem met Brute-force attacks is vooral dat er geen algoritmes zijn die het eenvoudiger en sneller maken om een wachtwoord te kraken, maar echt elk wachtwoord 1 voor 1 geprobeerd wordt. Om het aantal mogelijkheden te verkleinen worden woorden uit het woordenboek of de meest voorkomende wachtwoorden gebruikt. Ook kun je filteren dat de eerste letter een hoofdletter is en laatste twee cijfers zijn omdat dit vaak gebruikt wordt in wachtwoorden.

Omdat er veel inlogpogingen gedaan worden kan een site herkennen dat er een Brute-force attack uitgevoerd wordt en er iets tegen doen, zoals het account blokkeren.

Je kan omzeilen dat een account geblokkeerd wordt door hetzelfde wachtwoord op meerdere accounts te gebruiken. Doordat er niet snel achter elkaar op hetzelfde account wordt ingelogd, kan de website moeilijker herkennen dat geprobeerd wordt om met meerdere wachtwoorden in te loggen. Deze manier van Brute-force attack noem je Password Spraying.

Een goede manier om een Brute-force attack tegen te gaan, is het instellen van een re-captcha. Je bent het vast wel eens tegengekomen op een website. Een klein programmaatje waarbij je een aantal juiste afbeeldingen moet aanklikken of een woord opnieuw moet intypen. Dit voorkomt dat bots via brute-force attacks wachtwoorden proberen te raden of heel veel account aanmaken tot dat de database het niet meer aankan

Een andere manier om een Brute-force attack tegen te gaan is throttle. Zo'n applicatie kan een aanval tegenhouden, omdat het mislukte inlogpogingen bijhoudt. Throttle zorgt ervoor dat na 3 mislukte inlogpogingen de aanvaller geblokkeerd wordt of dat de gebruiker 1 minuut moet wachten voordat de aanvaller opnieuw een wachtwoord mag proberen. Hij kan nu minder snel wachtwoorden proberen te kraken.

[Bron: <https://realhosting.nl/helpdesk/wat-is-een-brute-force-attack/>]



[Bron: [https://nl.wikipedia.org/wiki/Brute_force_\(methode\)](https://nl.wikipedia.org/wiki/Brute_force_(methode))]

[Bron: <https://learn.onemonth.com/defensive-hacking-how-to-prevent-a-brute-force-attack/>]

2.7 Phishing

Een andere manier om inloggegevens te bemachtigen is via Phishing.

Phishing is een manier dat de hacker zich voordoeft als een vertrouwd bedrijf zoals Rabobank. De hacker kan dit doen door middel van een bericht te sturen naar de gebruiker met de vraag of hij wil inloggen om iets belangrijks te doen, bijvoorbeeld zijn account te verlengen voordat hun account verloopt.

De hacker kan dit bericht er heel vertrouwelijk uit laten zien door gebruik te maken van het logo van het bedrijf en om in het bericht vertrouwelijke informatie te gebruiken zoals de naam en e-mail van de gebruiker. Als de hacker ook vertrouwelijke informatie gebruikt in het bericht wordt het ook wel spear Phishing genoemd.

Als de gebruiker het bericht vertrouwt en op de link klikt wordt de gebruiker doorgestuurd naar een nagemaakte website die er (bijna) exact hetzelfde eruitziet als de echte site van dat bedrijf. De hacker probeert ook de URL zoveel mogelijk te laten lijken op die van de echte URL door middel van bijvoorbeeld hoofdletters i te gebruiken in plaats van een l. Door het gebruik van het IDN-systeem kan je nu ook gebruik maken van niet-ASCII tekens, bijvoorbeeld ë, waardoor het nog lastiger te zien is of de URL klopt of niet. Hierdoor is het voor de gebruiker erg lastig onderscheid te houden tussen de echte site en de site van de hacker.

Als de gebruiker zijn inloggegevens invult, worden die gegevens naar de hacker gestuurd. De hacker heeft dan meestal al een bot klaarstaan die de gegevens automatisch invult op de echte site en het wachtwoord zelf veranderd. De hacker heeft toegang tot de persoonlijke gegevens en, laten we Rabobank weer als voorbeeld nemen, kan geld van de rekening afschrijven.



2.8 Keylogger

Nog een andere manier is het gebruik van een keylogger. Dit is een programma die stiekem met andere software geïnstalleerd wordt, bijvoorbeeld als je een illegale versie van een game installeert, maar het kan ook via een usb-stick. De keylogger kan precies zien waar de gebruiker is geweest op het internet en alle gegevens doorsturen die de gebruiker intypt. Hiermee kan de hacker dus ook inloggegevens bemachtigen.

[Bron: <https://awaretrain.com/nl-nl/actueel/de-meest-gebruikte-methodes-om-wachtwoorden-buit-te-maken-en-hoe-je-ze-kan-voorkomen/>]



3 DEELVRAAG 3: WAT VOOR OPLOSSINGEN ZIJN ER OM EEN INLOGSYSTEEM VEILIGER TE MAKEN?

In deelvraag 2 hebben we het gehad over vele manieren waardoor een persoon in iemand anders zijn of haar account komt. Nu willen we erachter komen hoe inlogsystemen veiliger gemaakt worden, zodat deze mensen zich niet meer voor kunnen doen als iemand anders. Oplossingen om inlogsystemen veiliger te maken zijn:

3.1 Veiligheid voor wachtwoorden

Om het voor hackers moeilijker te maken om een wachtwoord te raden of te achterhalen met brute force attack, worden eisen gesteld aan de wachtwoorden die de gebruiker zelf opgeeft. Bijvoorbeeld een minimale lengte, gebruik van hoofdletters en kleine letters en gebruik van cijfers en speciale tekens.

Ook wordt er veel geadviseerd om niet hetzelfde wachtwoord op meerdere websites te gebruiken om credential stuffing te voorkomen.

Om het hacken van wachtwoorden via phishing te voorkomen kan een bedrijf vertellen dat zij nooit via mail of berichten vragen om in te loggen. Dit doen banken bijvoorbeeld. Dit heeft wel als nadeel dat zij zelf ook geen berichten of aanbiedingen naar de klant kunnen sturen waarvoor de klant dan moet inloggen om het te kopen

[Bron: <https://www.consumentenbond.nl/internet-privacy/wachtwoord-onthouden>]

[Bron: <https://www.veiligbankieren.nl/actueel/pas-op-voor-phishing-via-sms-en-whatsapp/#:~:text=Nederlandse%20banken%20sturen%20echter%20nooit,om%20beveiligingscode%20terug%20te%20sturen.&text=Bij%20een%20enkele%20bank%20kunt,SMS%20doorsturen%20naar%20een%20telefoonnummer.>]

3.2 Two-Factor Authentication

Als een hacker toch iemands wachtwoord heeft bemachtigd en wilt gebruiken, kunnen inlogsystemen gebruik maken van Two-Factor Authentication.

Two-Factor Authentication (ook wel 2FA genoemd), is een manier om gebruikers met twee manieren te laten bewijzen wie ze zijn. Hierbij kunnen twee van de drie vormen bewijs ingezet worden, namelijk: iets wat je weet (kennis), iets wat je bezit (bezit) en iets wat je bent (eigenschap). Een voorbeeld van 2FA is een pinpas en pincode. Als je iets wil kopen met je pinpas moet je als eerste je pinpas hebben (iets wat je bezit) en vervolgens je pincode invullen (iets wat je weet). Op deze manier is het veiliger om met je pinpas te betalen, doordat iemand die jouw pinpas heeft en jouw pincode niet weet er nog steeds niet mee kan betalen. Contactloos betalen is daarom ook minder veilig. Alleen het bezit van de pinpas is voldoende om te betalen. Daarom kun je alleen kleine bedragen contactloos betalen en wordt ook bij contactloos betalen soms om de pincode gevraagd.

[Bron: <https://www.vpngids.nl/veilig-internet/surfen/wat-is-twee-staps-verificatie/>]



Hieronder staan een aantal manieren hoe 2FA wordt gebruikt:

3.2.1 E-mail

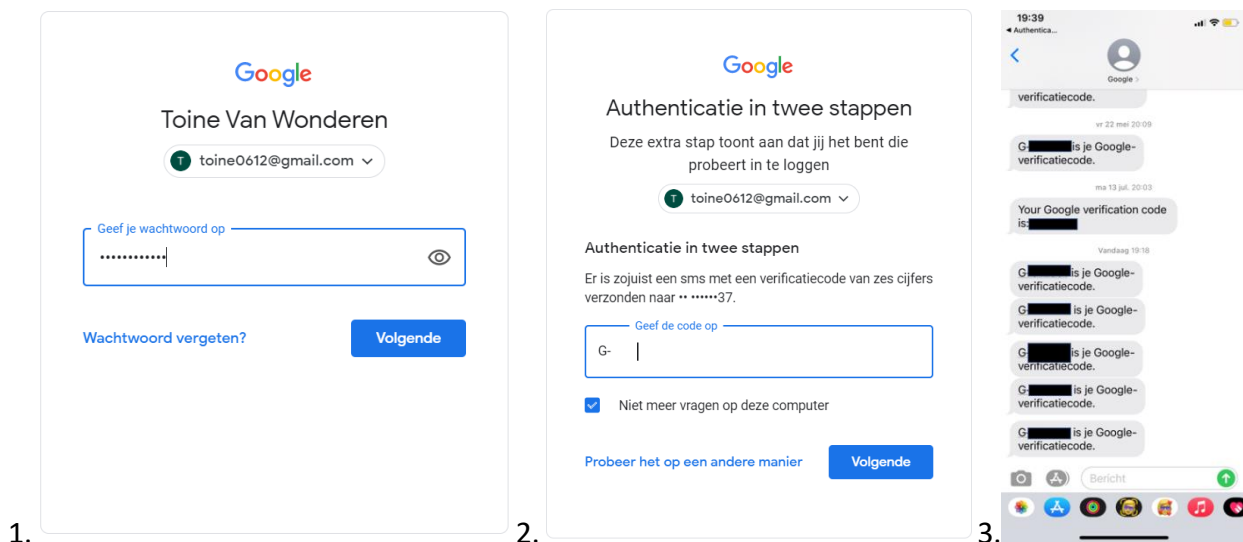
Als een gebruiker in wil loggen, kan een e-mail verstuurd worden naar het mailadres, wat bij de registratie voor deze gebruiker is opgegeven, met een code of een link die je kan gebruiken om toegang te krijgen tot jouw account. Deze manier valt onder de 'iets wat je bezit' (namelijk jouw e-mailadres).

Vaak wordt het mailadres ook gebruikt als een gebruiker zijn wachtwoord vergeten is of wilt veranderen. De gebruiker moet dan zijn e-mailadres invullen en krijgt daarna een mail met een link waarmee hij zijn wachtwoord kan veranderen. Door deze methode moet je ook toegang hebben tot het mailadres en kan niet iedereen zijn wachtwoord veranderen.

3.2.2 SMS

Een andere manier om gebruik te maken van 2FA is door gebruik te maken van een tweede apparaat. Voorbeelden hiervan zijn een code op je telefoon via SMS of via een app.

We gebruiken het Google-account van Toine om 2FA te testen in de volgende voorbeelden. Om 2FA via sms in te stellen moet de gebruiker zijn of haar telefoonnummer invullen en zal er ter verificatie een code verstuurd worden die ingevuld moet worden. Als de gebruiker daarna weer wil inloggen krijgt hij een code op zijn telefoon via de sms die hij dan moet invullen om toegang te krijgen tot zijn of haar account. Op deze manier heeft de gebruiker een wachtwoord en de telefoon die aan het account gekoppeld is nodig. Dit is een voorbeeld:



1. Als eerste vul je je eigen e-mailadres en wachtwoord in;
2. Daarna heeft google een verificatie code gestuurd die je hier moet invullen om toegang te krijgen tot je account;



3. als de juiste is ingevuld krijg je toegang tot jouw account.

3.2.3 Authenticator app

Een authenticator app is een app op je telefoon die je kunt koppelen aan je account zodat je met een 2FA kunt inloggen. De app genereert iedere 30 seconde een code die je moet invoeren als je wilt inloggen. Dit werkt hetzelfde als een code via SMS.

Als je de authenticator app van Google wil gebruiken als 2FA methode moet je de app installeren en koppelen aan het account:

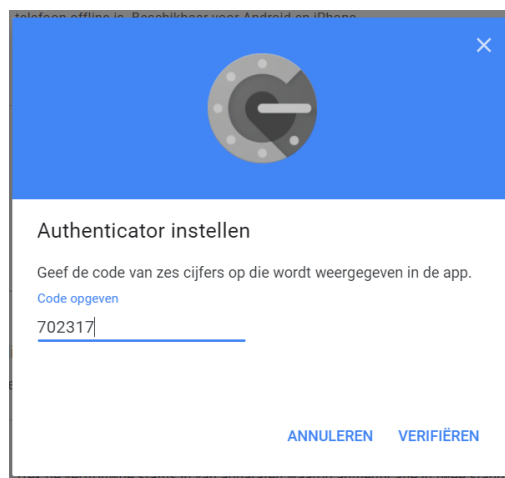
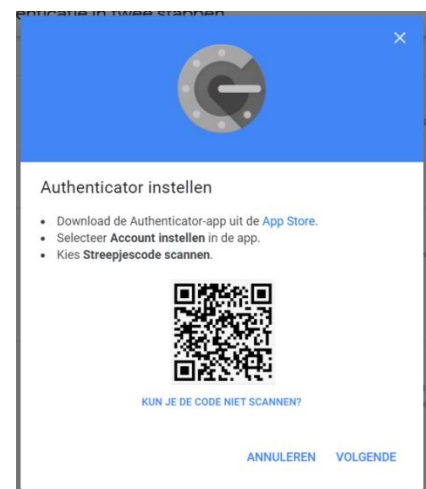
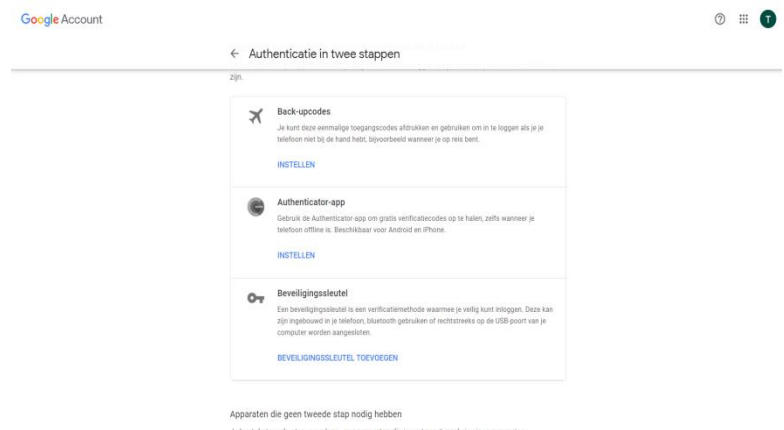
Als eerste ga je naar instellingen om de authenticator app te installeren.

Als je dat gedaan hebt moet je in de app de QR-code scannen die wordt laten zien. Deze QR-code bevat een secret key die zorgt dat de authenticator app gekoppeld wordt aan jouw account.

In de app wordt iedere 30 seconde een nieuwe code laat zien. Je moet de code invullen voordat het verloopt om de app in te stellen.

Als je de volgende keer weer wilt inloggen moet je naast je wachtwoord wederom de nieuwste code van de authenticator app invullen voordat deze verloopt.

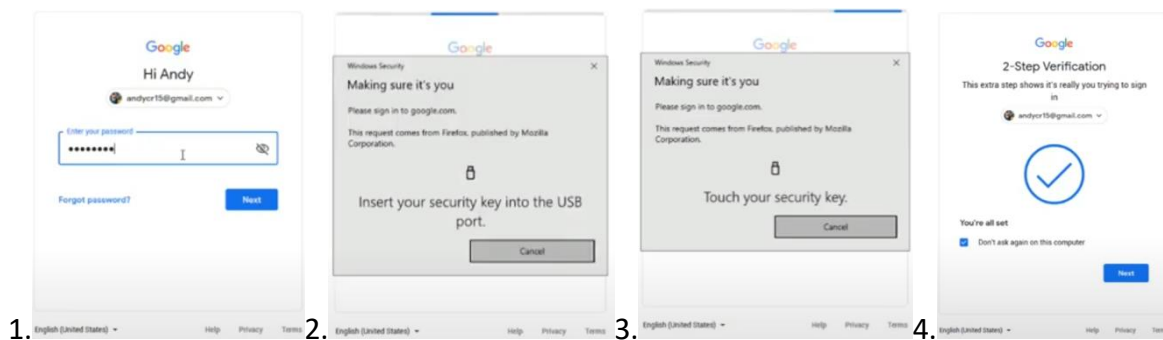
Er zijn ook apparaten die steeds nieuwe codes maken die je moet invullen in het inlogstelsel.



4.2.4 Yubikey

Yubikey is een authenticatiesleutel in de vorm van een usb-stick die je kan gebruiken om in te loggen. Om de Yubikey te installeren moet je naar de webpagina gaan waar je de Yubikey wilt gebruiken en inloggen. Daarna activeer je de Yubikey op die pagina via de instellingen. Als je daarna opnieuw wilt inloggen vul je zoals gewoonlijk jouw e-mailadres en wachtwoord in (Afbeelding 1) en zal de website vragen om de Yubikey in de computer te stoppen (Afbeelding 2) en op de knop te drukken op de Yubikey die vervolgens een toegangscode stuurt naar de pagina (Afbeelding 3) en je toegang geeft tot jouw account (Afbeelding 4).

Yubikey is veilig omdat je een e-mailadres en wachtwoord nodig hebt (iets wat je weet) en daarnaast ook de Yubikey in de computer moet doen (iets wat je bezit). De code wordt ook rechtevreeks doorgegeven aan de computer en wordt niet over een netwerk verstuurd. Hierdoor kan de code heel moeilijk onderscheept worden. Ook is er een Yubikey waarbij je in de plaats van op een knop moet drukken, je vinger moet scannen om de code te versturen. Hierdoor moet je nogmaals bewijzen dat de Yubikey van jou is en is de Yubikey een erg veilige manier om als 2FA te gebruiken.



[Bron: <https://www.yubikeyshop.nl/nl/wat-is-een-yubikey.html>]

[Bron:

https://www.infinigate.nl/nieuws/detail/news/yubikey_bio_veilig_overal_passwordless_inloggen_met_je_vingerafdruk.html]

[Bron: <https://tweakers.net/nieuws/159476/yubico-brengt-yubikey-bio-met-vingerafdrukauthenticatie-uit.html>]

4.2.5 Wat is de veiligste 2FA methode?

In veel gevallen is de sms 2FA een veiligere manier dan via je e-mail. Als iemand jouw e-mailadres en wachtwoord weet van jouw email account, kan de persoon de authenticatie mail openen en zich authenticeren zonder dat jij dat wil. Authenticatie via sms is daarom een veiligere manier, aangezien de telefoon die jij bezit in meeste gevallen altijd bij jou is, tenzij de telefoon gestolen is, maar dan heeft diegene vaak nog een wachtwoord nodig om in je telefoon te komen.

Ondanks dat je nu een tweede apparaat nodig hebt om in te loggen, is er wel kritiek op het gebruik van SMS als 2FA methode. Het is nog steeds te makkelijk om via spoofing of phishing een sms-bericht te onderscheppen. Ieder sms-bericht heeft zijn adres informatie en door middel van spoofing kan de hacker die informatie veranderen en naar een ander mobiel versturen. Ook kan er



gebruik gemaakt worden van phishing om de 4-cijferige pincode te stelen en de sms'jes te kunnen lezen.

Het National Institute of Standards and Technology (NIST) van de VS heeft in 2016 gezegd dat sms-berichten makkelijk te onderscheppen zijn en het daarom niet veilig is. Daarentegen raadt het NIST aan om een authenticator app te gebruiken, omdat zulke apps beter versleuteld zijn in tegenstelling tot sms-berichten. In 2014 had nu.nl ook al een bericht uitgebracht waarin stond dat hackers sms'jes konden lezen, telefoongesprekken konden luisteren en je locatie konden volgen van alle providers die gebruik maakte van ss7-protocol.

Bij Yubikey heb je net zoals bij een authenticator app een tweede apparaat nodig waarbij je eerst jezelf moet identificeren, namelijk de Yubikey of je telefoon. Het verschil is dat de code bij een authenticator app over een netwerk verstuurd wordt en bij de Yubikey de code via de USB-poort gegeven wordt. Daarom denken wij dat de Yubikey nog net iets veiliger is dan een authenticator app. Wel is het nadeel dat je een Yubikey moet aanschaffen wat prijzig kan zijn in tegenstelling tot een telefoon die iedereen waarschijnlijk al heeft. Daarom gebruiken wij voor ons praktische opdracht een authenticator app.

[Bron: <https://www.techzine.be/nieuws/security/16875/reddit-hack-toont-beperkingen-tweestapsverificatie-via-sms/>]

[Bron: <https://www.onespan.com/blog/nist-softens-guidance-sms-authentication>]

[Bron: <https://www.nu.nl/internet/3957282/telefoongesprekken-en-smsjes-gemakkelijk-onderscheppen-lek.html>, <https://securityboulevard.com/2020/01/why-2fa-sms-is-a-bad-idea/>]

[Bron: https://en.wikipedia.org/wiki/SMS_spoofing]

3.3 reCAPTCHA

ReCAPTCHA betekent Completely Automated Public Turing test to tell Computers and Humans Apart en is een reactietest die in de gegevensverwerking wordt gebruikt om te bepalen of er al dan niet sprake is van een menselijke gebruiker. Anders gezegd, met het script (dat Google heeft gemaakt), kan worden gecheckt of de gebruiker een mens is en geen robot. Dit doet het bedrijf doormiddel van een kleine challenge, bijvoorbeeld een geluidsfragment, een gek geschreven code over te typen of een selectie van plaatjes aanklikken wat een mens wel kan, maar heel moeilijk is voor een computer om te herkennen. Zo'n kleine challenge heeft iedereen vast wel eens gedaan waarbij je plaatjes van bijvoorbeeld een zebrapad moet aanvinken. Lukt dit dan kan de gebruiker door. Lukt dit niet wordt de gebruiker tegengehouden. Door op deze manier bots tegen te houden, wordt veel spam tegengegaan.

[Bron: <http://www.pchulpnoord.nl/computer-tips-en-advies/leuk-weetje-over-recaptcha#:~:text=ReCaptcha%20is%20een%20versie%20van,tell%20Computers%20and%20Humans%20Apart.>]

3.4 Vingerscan:

Een andere manier van beveiliging is de vingerscan. Met deze methode leg je een vinger op het scan apparaat die vervolgens jouw vingerafdruk opslaat, zodat als je jouw vinger de volgende keer



op het scan apparaat legt, het apparaat jou herkent. Er zijn 3 manieren om een vingerscan te maken:

Optische scanner

Bij de optische scanner leg je jouw vinger op de scanner die vervolgens een foto maakt van je vingerafdruk. Deze manier van vingerscan is erg oud en is makkelijk te vervalsen door een foto van iemand anders zijn of haar vinger voor de scanner te houden.

Capacitieve scanner

De capacitieve scanner maakt gebruik van een condensator. Als je jouw vinger op de scanner legt, zullen delen van je huid de scanner raken en sommige delen niet. De scanner ziet de delen die de scanner raken. Hiermee kan de scanner een uniek beeld van jouw vinger maken.

Ultrasone sensor

De ultrasone sensor maakt gebruik van ultrageluid. Als je jouw vinger op de scanner legt, zal de sensor geluidsgolven sturen naar de vinger die verschillend weerkaatsen. Hierdoor krijgt de sensor een 3D beeld van jouw vinger om vervolgens veilig in te loggen.

[Bron: <https://techpulse.be/achtergrond/212622/zo-werkt-een-vingerafdrukscanner/#:~:text=Capacitieve%20scanner&text=Door%20je%20vinger%20op%20de,je%20vinger%20de%20scanner%20raakt.>]

De nadelen bij deze scanners zijn dat je niet met natte of vieze vingers deze scanners kan gebruiken doordat ze jouw vingerafdruk niet zullen herkennen. Een Frans bedrijf Idema heeft daarom een scanner gemaakt die op afstand je vinger kan scannen. De scanner zendt groen licht uit waarmee de vinger belicht wordt en vervolgens maken de camera's een 3D model van de vinger. Het bedrijf zegt dat deze manier niet alleen beter en sneller is, maar ook hygiënischer, omdat je niet met je vinger op een scanner hoeft te drukken. Hierdoor denken ze dat het ook bruikbaar is als authenticatie middel.

[Bron: <https://www.kaspersky.nl/blog/new-fingerprint-reading-technologies/23850/>]

Een simpel voorbeeld hoe vingerscan gebruikt wordt is je telefoon. Telefoons maakten vroeger meestal gebruik van een cijfercode, maar tegenwoordig maken telefoons ook gebruik van een vingerscan als extra inlogmiddel. Telefoons gebruiken de capacitieve scanner. Het gebruik van vingerscan maakt het veel eenvoudiger om in te loggen op je telefoon en jezelf te identificeren in plaats van iedere keer je code in te vullen.

Naast dat het eenvoudiger is, is het ook veiliger, mits ze als extra inlogmiddel te gebruiken. Dat komt, omdat je vingerafdruk een biometrische eigenschap is die moeilijk na te maken is. Ook is de kans dat iemand een identieke vingerafdruk heeft waarmee hij de scanner kan misleiden 1 : 50.000. Als het als extra inlogmiddel dient, zoals bij 2FA, heb je ook nog een wachtwoord code of telefoon nodig waardoor het nog veiliger is.

[Bronnen: <https://www.appletips.nl/face-id/>, <https://www.kaspersky.nl/blog/new-fingerprint-reading-technologies/23850/>]



Toch werd vroeger veel gediscussieerd over of het veilig is om vingerscan te gebruiken. Dit komt doordat je vingerafdruk een biometrische eigenschap is die je niet kan veranderen. Het is dus uitermate belangrijk dat het bedrijf zorgt dat je vingerafdruk veilig is opgeslagen. Als je vingerafdruk namelijk “gestolen wordt” kan diegene die jouw afdruk heeft, fraude plegen.

Ook laat je zelf overal jouw vingerafdruk achter, dus met veel moeite is het mogelijk jouw vingerafdruk na te maken en te gebruiken.

Tegenwoordig kan je ook betalen met je telefoon via Apple pay. Hierbij wordt ook een code en vingerscan gebruikt. Iedere keer als je wilt betalen, moet je jouw vingerafdruk tonen of je code invoeren om te kunnen betalen. Hierdoor is het nog belangrijker dat de vingerafdruk veilig bewaard wordt. Als oplossing hiervoor moet je bij bijvoorbeeld smartphones na een aantal mislukte pogingen om met vingerscan in te loggen of als je je telefoon opnieuw opstart je code invullen. Hierdoor blijft de telefoon zeker weten dat jij de juiste persoon bent die de telefoon beheert.

[Bron: <https://tweakers.net/reviews/5623/all/biometrie-vloek-of-zegen.html>]

3.5 Gezichtsherkenning:

Gezichtsherkenning is een van de nieuwste biometrische technologieën op het gebied van veilig inloggen. De technologie was al langer uitgevonden, maar nog nooit gebruikt als authenticatie middel. Tegenwoordig zie je dit steeds vaker. De grootste verandering was Apple, met haar nieuwe iPhone met Face ID. Deze Face ID is een extra manier voor de gebruiker om in te loggen op zijn telefoon waardoor de telefoon de gebruiker kan authenticeren. Het doel van Apple met Face ID is, net als bij de vingerscan, om gemakkelijker in te kunnen loggen en het nog veiliger te maken. Voor Face ID gebruikt Apple een TrueDepth-camera, dat is een combinatie van een normale camera, infraroodcamera, dieptesensor en projector. Door de TrueDepth-camera wordt infraroodstraling uitgezonden die 30.000 puntjes op het gezicht meet en vervolgens foto's van het gezicht maakt om een 3D beeld te krijgen van het gezicht. Wat het voordeel van het gebruik van infrarood is dat infrarood ook door materiaal heen gaat. Hierdoor heeft de Face ID geen problemen met het scannen van jouw gezicht als je bijvoorbeeld een zonnebril ophebt.

En net zoals bij de vingerscan, wordt bij Face ID na een aantal mislukte pogingen of als je je telefoon opnieuw opstart je code gevraagd om in je telefoon te komen. Wat Face ID veilig maakt is dat als je Face ID wilt vervalsen je een 3D beeld van je hoofd moet gebruiken, een foto zal dus niet werken. Ook is de kans dat iemand dezelfde Face ID heeft 1 : 1.000.000, dat is nog 20 keer zo veilig als de vingerscan. Daarnaast blijven alle scans op de telefoon waardoor deze moeilijk gestolen of gehackt kunnen worden.

[Bron: <https://www.appletips.nl/face-id/>]

4 DEELVRAAG 4: DE INLOGGEGEVENS VAN GEBRUIKERS MOETEN WORDEN OPGESLAGEN IN EEN DATABASE, HOE GAAT DIT VEILIG?

Bij een inlogstelsysteem is het van belang dat de gegevens van de ingelogde gebruiker bewaard worden. Hiermee kan de gebruiker later als hij/zij terugkeert weer met dezelfde inloggegevens inloggen op de website. Dit moet natuurlijk wel beveiligd zijn. In dit hoofdstuk bekijken we:

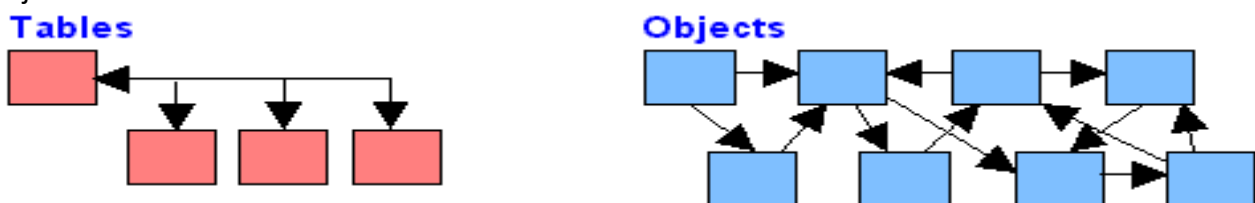
- Wat een database is;
- Hoe veilig databases zijn;
- Wie de data in de databases kan zien.

4.1 Wat is een database?

Hoe zorgen we dat deze gegevens veilig bewaard kunnen worden zonder dat er andere mensen bij kunnen? Dit doen we door een veilige database te maken. Hierin worden de gegevens van de gebruiker opgeslagen, zoals het emailadres en wachtwoord. Een database is dus een verzameling gegevens die digitaal opgeslagen zijn. Deze database kan bijvoorbeeld gemaakt worden met MySQL. Dit is een veel gebruikt database type, omdat deze in combinatie met PHP gratis is en op elk platform uitgevoerd kan worden. Ook wij zullen voor onze database gebruik maken van PHP en MySQL.

De database type die we hierboven besproken hebben is een 'relationele' database. Dit houdt in dat de tabellen een relatie met elkaar hebben (een verband). De relaties worden gemaakt door middel van sleutels. De items in deze relationele database zijn georganiseerd als een set met tabellen met kolommen en rijen.

Andere soorten databases zijn de object en flat file databases. Deze worden minder gebruikt en hebben hun voor- en nadelen. Bij een relationele database maak je gebruik van een-op-veel relaties. Bij een object database is dit anders. Hier zijn veel-op-veel relaties mogelijk. Zo worden bij applicaties die complexe relaties vereisen, vaak object databases gebruikt. De gegevens kunnen hier makkelijker in worden opgeslagen. Onderstaande figuur laat het verschil tussen relationele en object databases zien:



Een ander type database is een flat file database. Dit is een database die simpel in een bestandje wordt uitgedrukt. Dit is een voordeel, omdat de andere twee databases soms groot en complex zijn. Echter het nadeel aan een flat file database is dat er geen structuren voor het indexeren of herkennen van relaties aan gebonden is. Je moet dus zelf met het blote oog relaties leggen tussen informatie. Dit doet de database niet uit zichzelf.

Er bestaat dan ook nog een distributed database. Dit is een erg groot en uitgebreide database voor bedrijven, zoals Google. Zij slaan data op, in verschillende plekken. De database is een verzameling van meerdere verbonden databases, verspreid over verschillende locaties. De laatste 2 overige



databases zijn de NoSQL database, wat eigenlijk inhoudt dat er geen gebruik gemaakt wordt van SQL en makkelijk te beheren is. En de Graphic database die geeft data in grafieken weer, wat voor sommige bedrijven handig kan zijn om zo hun data in te kunnen zien in percentages.

Wij zullen ons houden aan de relationele database type, omdat deze keuze het handigst voor ons project is. We gebruiken dit type, omdat dit type gratis is en omdat we de database goed kunnen beheren met PHP en MySQLi. We zullen dan in onze PHP-code query's schrijven die zorgen dat we data in de database kunnen toevoegen of data kunnen ophalen.

Wij zullen met ons praktisch gedeelte gebruik maken van PHP en MySQL, maar hoe werkt dit? Hoe komt de data in de database in de webapplicatie? In de code die wij gaan schrijven, koppelen we de applicatie met de database, door middel van PHP. Zo kunnen we vanuit de webapplicatie data invoeren dat rechtstreeks de database ingeschreven wordt. Hier wordt onze data bewaart. Ook haal je zo de data op in de webapplicatie die in de database staat.

Een database gebruikt meestal een databasesoftware programma (DBMS). Dit dient als interface tussen de database en het programma. Wij gebruiken PHP en MySQL. Hierbij is MySQL de databasebeheersysteem. Met een DBMS kan de informatie worden beheert, georganiseerd en geoptimaliseerd. Er kunnen ook administratieve bewerkingen mogelijk zijn, zoals prestatiebewaking en/of een backup.

[Bron: <https://autoriteitpersoonsgegevens.nl/nl/onderwerpen/avg-europese-privacywetgeving>]

[Bron: <https://www.killersites.com/translations/dutch/wat-zijn-databases.html>]

[Bron: <https://www.oracle.com/be-nl/database/what-is-database.html>]

4.2 Hoe maken we deze database veilig?

We hebben in deelvraag 2 al behandeld wat de manieren zijn om voor een hacker een account of database in te komen. In deze deelvraag behandelen we de vraag hoe we onze database veilig kunnen maken. Er zijn een aantal manieren om een database in te komen. Vaak wordt dit gedaan door een hacker die gaat proberen het administrator-wachtwoord te achterhalen. Ze doen dit bijvoorbeeld via een SQL-injecties. In deelvraag 2 hebben we besproken hoe deze methode werkt. De hacker maakt namelijk via een SQL-query connectie met de database en krijgt zo toegang. Wat zijn de oplossingen om dit tegen te gaan?

4.2.1 Database firewall

Een database firewall installeer je voor de beveiliging van je database. Het helpt je met de bescherming van bijvoorbeeld SQL-injecties en buffer overflows. De firewall kan bijvoorbeeld SQL-injecties herkennen doordat er een aantal SQL-injecties gedefinieerd zijn in de firewall op een blacklist. Ook slaat de firewall oude SQL-injecties op, waardoor de firewall steeds meer injecties herkent. Als er toch een andere SQL-injecties wordt uitgevoerd heeft de firewall een whitelist, hierdoor weet de firewall dat hij deze code moet controleren. Als er een hacker in de database is gekomen herkent de firewall dat ook. Als dit het geval is kan de firewall actie nemen om te zorgen dat er geen data uit de database gaat en er geen data lek ontstaat.

Sommige firewalls kijken ook naar het IP-adres en locatie van de gebruiker. Hierdoor kan de firewall onderscheid maken tussen hackers en gebruikers.

[Bronnen: <https://executive-people.nl/582339/databases-tips-om-de-security-op-scherp-te-zetten.html>, <https://www.sciencedirect.com/topics/computer-science/database-firewall>, <https://excitingip.com/1933/what-are-database-firewalls-why-are-they-required-how-do-they-protect-databases/>]

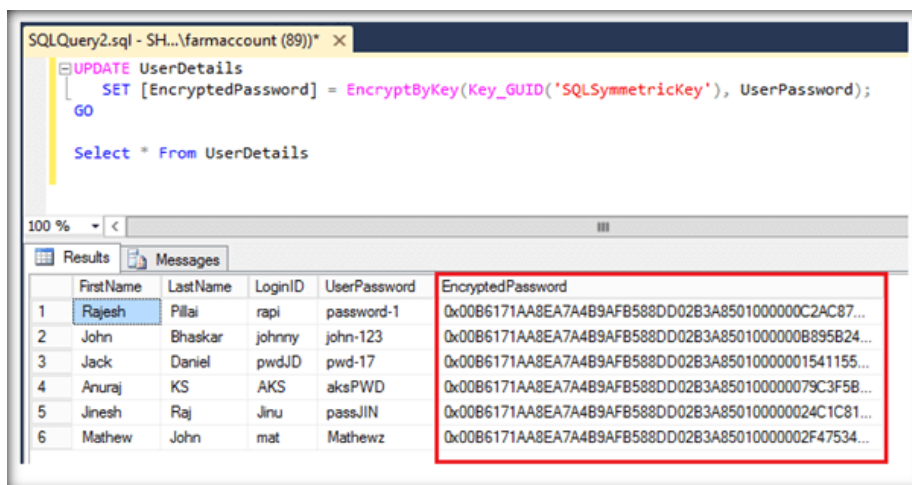
4.2.2 Encryptie

De belangrijkste oplossing om een database veilig te maken is het encrypten (versleutelen, coderen) van data. Dit gebeurt meestal bij het wachtwoord van de gebruiker, maar kan ook voor persoonsgegevens. Het nadeel hiervan is dat jijzelf meestal deze data ook niet kan zien, mits je een symmetrische cryptografie gebruikt. Dit houdt in dat de informatie gedecodeerd wordt met een sleutel. Coderen en decoderen gebeurt dan met dezelfde sleutel.

Ook heb je asymmetrische cryptografie. Hier worden twee verschillende sleutels gebruikt. Een openbare sleutel en een privésleutel. De privésleutel moet geheim blijven en kan gebruikt worden om de data in te zien. De openbare sleutel wordt met iedereen gedeeld. Hiermee kan de data encrypted worden opgeslagen.

Wij kunnen geen beschikking krijgen tot de asymmetrische cryptografie aangezien wij onze applicatie met PHP en sql maken. Dit maakt gebruik van symmetrische cryptografie.

Een versleuteld wachtwoord ziet er in de database uit als een lange niet te lezen code. Zie afbeelding hieronder.



The screenshot shows a SQL query editor window titled 'SQLQuery2.sql - SH...\farmaccount (89))' with the following SQL code:

```
UPDATE UserDetails
SET [EncryptedPassword] = EncryptByKey(Key_GUID('SQLSymmetricKey'), UserPassword);
GO

Select * From UserDetails
```

Below the query, a 'Results' tab is active, displaying a table with 6 rows and 6 columns: FirstName, LastName, LoginID, UserPassword, and EncryptedPassword. The 'EncryptedPassword' column is highlighted with a red box. The data in the table is as follows:

	FirstName	LastName	LoginID	UserPassword	EncryptedPassword
1	Rajesh	Pillai	rapi	password-1	0x00B6171AA8EA7A4B9AFB588DD02B3A850100000C2AC87...
2	John	Bhaskar	johnny	john-123	0x00B6171AA8EA7A4B9AFB588DD02B3A850100000B895B24...
3	Jack	Daniel	pwdJD	pwd-17	0x00B6171AA8EA7A4B9AFB588DD02B3A8501000001541155...
4	Anuraj	KS	AKS	aksPWD	0x00B6171AA8EA7A4B9AFB588DD02B3A85010000079C3F5B...
5	Jinesh	Raj	Jinu	passJIN	0x00B6171AA8EA7A4B9AFB588DD02B3A85010000024C1C81...
6	Mathew	John	mat	Mathewz	0x00B6171AA8EA7A4B9AFB588DD02B3A8501000002F47534...

Dit zorgt ervoor dat data niet eenvoudig uit te lezen is wanneer de database gehackt zou worden.

[Bron: <https://nl.wikipedia.org/wiki/Encryptie>]

[Bron: <https://www.kaspersky.nl/resource-center/definitions/encryption>]

4.2.3 Speciale tekens

Er is nog een handige manier om de database te beveiligen tegen SQL-injecties. Dat is het verbieden van speciale tekens en het instellen van een minimum en maximum van het aantal tekens.

Wanneer je speciale tekens, zoals komma's, uitroeptekens, vraagtekens etc... verbiedt in de input's van de PHP kunnen er geen gekke tekens in de database terecht komen. Deze tekens heb je over



het algemeen ook niet nodig aangezien er niemand een voornaam heeft met een dollarteken erin. Dit soort tekens willen hackers nog wel eens gebruiken als manier om te hacken. Dit zorgt dus voor een lastigere manier voor de hacker om in de database terecht te komen. Wel wordt er gebruik gemaakt van speciale tekens in wachtwoorden om die zo veilig mogelijk te houden. Daarom is het niet altijd even handig om speciale tekens te verbieden. Een oplossing hiervoor is een `mysqli_real_escape_string()` te gebruiken. Dit is een functie in PHP die de waarde van het input veld omzet naar een string waarbij de speciale tekens geen werking meer hebben. Dus als de gebruiker een apostrof invult zal deze niet meer de string beëindigen als de `mysqli_real_escape_string()` functie is uitgevoerd op die waarde.

[Bron: <https://www.hacksplaining.com/prevention/sql-injection>]

[Bron: <https://www.php.net/manual/en/mysqli.real-escape-string.php>]

4.2.4 *Prepare statements*

Een goeie bestrijding van sql-injecties, is een prepare statement. Een prepare statement biedt de programmeur de mogelijkheid om zijn of haar query veiliger te maken. Zo kan deze op een efficiënte manier vele malen uitgevoerd worden. Bij een prepare statement wordt er eerst een template gestuurd naar de server. Deze wordt gecheckt op syntax fouten en of de query logisch is. Daarna worden deze gegevens opgeslagen. Prepare statement is een bestrijding tegen SQL-injectie omdat de waarde pas later verzonden wordt en gecheckt wordt op tekens die er niet horen te staan. Deze methode gaan wij ook gebruiken in ons praktisch gedeelte.

[Bron: <https://www.phphulp.nl/php/tutorial/overig/pdo-verbinden-met-verschillende-databases/534/prepared-statements/1368/>]

4.3 Wie kan de data in de database zien?

We willen er dus voor zorgen dat niet iedereen onze database in kan. Dit is ook wettelijk verplicht in Nederland. In Nederland heb je als website of webshop met inlogstelsysteem veel te maken met privacywetgeving (AVG). Het is noodzakelijk dat de gegevens van de gebruiker niet zomaar op straat terecht komen. Daarom is deze wetgeving voor de versterking en uitbreiding van privacy rechten. Ook zorgt dit voor meer verantwoordelijkheid bij organisaties, oftewel het inlogstelsysteem moet veilig zijn. We zijn dus noodzakelijk dat alleen wij tot deze data beschikken en geen andere mensen.

Wie kan deze data nou precies inzien? Om mensen de data in databases te laten bekijken hebben ze daar een privilege voor nodig. Zonder dat privilege mag je die data niet bekijken en/of aanpassen. We hadden al een stukje besproken in het kopje 'encryptie' waar we uitlegden dat je met symmetrische cryptografie een sleutel nodig hebt om de data te kunnen lezen. Zo'n sleutel zal je dan alleen krijgen als daar het privilege voor hebt.

[Bron: <https://autoriteitpersoonsgegevens.nl/nl/onderwerpen/beveiliging/beveiliging-van-persoonsgegevens>]

5 DEELVRAAG 5: KUN JE MET ÉÉN INLOGMIDDEL OP MEERDERE SITES INLOGGEN EN IS DAT PRIVACYGEVOELIG?

Veel sites vragen als jij wilt inloggen, of je wilt inloggen met bijvoorbeeld jouw Facebook- of Google-account in plaats van een nieuw account maken op hun website. Deze manier werkt gemakkelijk, omdat je bijvoorbeeld niet talloze wachtwoorden hoeft te onthouden. Ook voor de websitebouwer is het handig omdat die niet zelf een inlogstelsel hoeft te maken. Maar krijgt bijvoorbeeld Google dan ook alle informatie van die website en is dit wel veilig?

5.1 Hoe werkt inloggen met een account van Google?

Een gebruiker gaat naar een website waar hij wil inloggen.

Als hij kan inloggen met zijn google of Facebook-account, dan klikt hij op de knop “inloggen met mijn Google-account”.

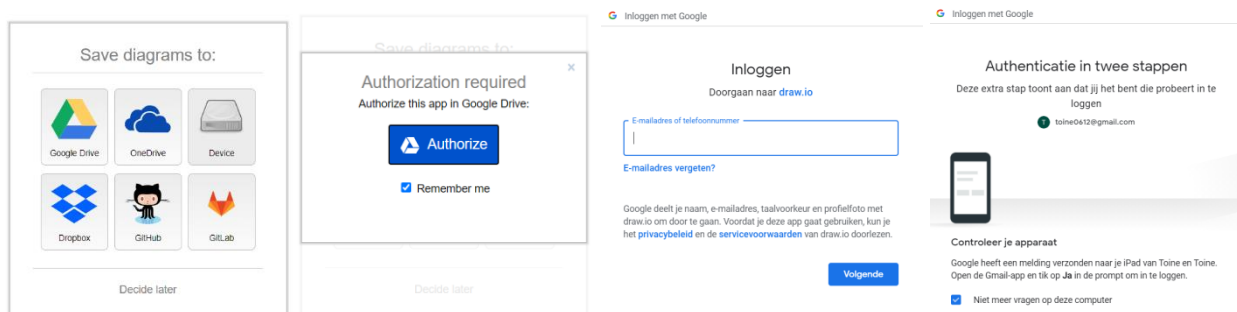
De website maakt een verbinding met Google. De gebruiker logt in op de website van Google.

Google geeft vervolgens de gegevens van de gebruiker aan de website die op die manier weet wie er wil inloggen (identificatie).

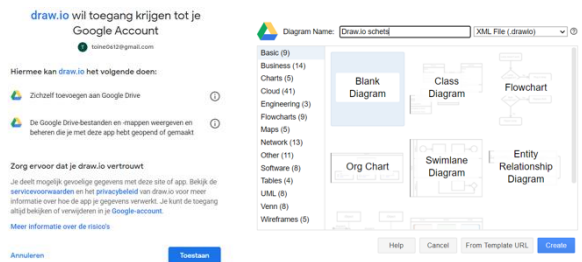
De website kan vervolgens zelf in een eigen database zien wat deze gebruiker mag en wat hij interessant vindt (autorisatie en personalisatie).

Het is ook mogelijk dat de website toegang krijgt tot bijvoorbeeld Google-drive.

Als voorbeeld kijken we naar hoe het werkt door een Google-account te koppelen aan de website draw.io. Als je jouw Google-account koppelt aan draw.io kan draw.io projecten die jij maakt direct opslaan in de Google-drive. Vervolgens kan jij via Google-drive het project overal openen. Door dit gemak kiezen veel mensen om dit te doen.

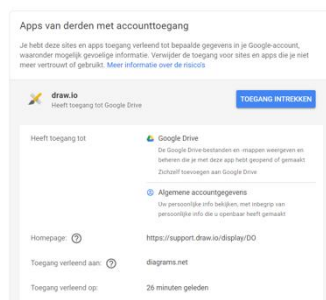


Draw.io vraagt of ik mijn Google-drive wil koppelen door in te loggen via Google zelf. Dit betekent dat je jezelf moet authenticeren bij Google zelf waarna je draw.io toegang kan geven tot jouw gegevens.



Als je draw.io toegang hebt gegeven tot jouw Google-account en Google-drive kan draw.io projecten direct opslaan in jouw Drive.

Op de supportpagina van Google staat dat derde partijen waar jij met jouw Google-account bent ingelogd toegang kunnen krijgen tot je basisprofiel (naam, profielfoto en e-mailadres), informatie kunnen bekijken en/of ophalen of content in jouw Google-account kunnen bewerken, uploaden, maken en verwijderen. Google zal nooit het wachtwoord van jouw Google-account delen voor veiligheid. In dit voorbeeld kan draw.io gegevens van mijn Drive uploaden.



Als de gebruiker niet meer wil dat een website toegang heeft tot zijn Google-gegevens kan hij altijd de toegang intrekken. De gebruiker moet dan naar zijn Google-account > Beveiliging > Apps van derden met accounttoegang. Op deze manier kan de gebruiker altijd nog beheren welke websites welke informatie heeft.

[bron: https://support.google.com/accounts/answer/10130420?hl=nl&ref_topic=7188760]

5.2 Voor- en nadelen

Het is handig als je overal kunt inloggen met bijvoorbeeld je Google of Facebook-account. Je hoeft niet voor alle websites wachtwoorden te onthouden.

Ook voor de websitebouwer is het handig om gebruikt te kunnen maken van Google of Facebook-accounts. Hij hoeft dan niet zelf een inlogstelsysteem te bouwen.

Er zijn ook nadelen.

Als het google of Facebook-account gehackt is (bijvoorbeeld via Fishing), dan kan de hacker ook bij jouw gegevens. Ook is het privacygevoelig. Facebook bijvoorbeeld wil graag informatie weten over hun gebruikers, zodat ze reclames kunnen tonen die bij de gebruiker horen. Als gebruikers met hun Facebook-account inloggen, dan weet Facebook welke websites je bezoekt, hoe vaak je dat doet en wanneer je dat doet.



Wel heeft Facebook op 20 augustus 2019 een nieuwe functie aangekondigd: 'off-Facebook activity'. Hiermee kan de gebruiker inzage krijgen wat Facebook over hem weet en dit aanpassen. De gebruiker kan toekomstige activiteiten van die derde partij uitschakelen. Hieronder een overzicht wat de invloed daarvan is:

Toekomstige activiteit beheren

×

Toekomstige activiteit van ikea.com uitschakelen?

Dit zijn een aantal dingen die je moet weten:

- **Als je dit uitschakelt, wordt de koppeling met je toekomstige activiteit verwijderd.** Het kan 48 uur duren voordat de koppeling met je account volledig is verwijderd.
- **Als je met Facebook bent aangemeld bij ikea.com, word je mogelijk afgemeld als je je activiteit uitschakelt.** Je kunt je in de toekomst ook niet meer met Facebook aanmelden bij ikea.com.
- **We ontvangen nog steeds activiteit van ikea.com.** Deze kan worden gebruikt voor metingen en om onze advertentiesystemen te verbeteren, maar wordt losgekoppeld van je account.
- **Je kunt nog steeds advertenties zien van ikea.com.** Je advertentievoorkeuren en acties die je uitvoert op Facebook, worden gebruikt om relevante advertenties aan je te tonen.

Annuleren

Uitschakelen

[Bron: <https://www.seniorweb.nl/tip/is-inloggen-met-een-facebookaccount-veilig>]

[Bron: <https://www.marketingfacts.nl/berichten/off-facebook-activity-voor-bedrijven>]

[Bron: <https://www.hln.be/ihln/facebookers-kunnen-voortaan-informatie-die-derde-partijen-over-hen-verzamelen-wissen~a90108c9/?referrer=https%3A%2F%2Fwww.google.com%2F>]

[Bron: <https://www.facebook.com>]

5.3 Inloggen met IRMA

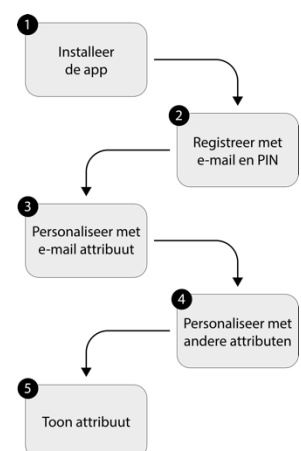
Het bedrijf IRMA is misschien wel een oplossing voor de privacy-gevoeligheid bij het inloggen bij derde partijen.

Het verschil tussen IRMA en inloggen via Google of Facebook-account is dat met IRMA jouw gegevens alleen op je eigen telefoon staan. Wat IRMA zo privacy-vriendelijk houdt, is het feit dat IRMA en de controleur (de derde partij) van de gegevens geen contact met elkaar hebben.

Om IRMA te gebruiken moet je de app downloaden. In de app kan je persoonlijke informatie, ook wel attributen genoemd, opslaan om die later te gebruiken voor authenticatie. Je kunt bijvoorbeeld attributen ophalen bij de gemeente door in te loggen met je DigiD. De uitgever zet ook een blinde digitale handtekening bij die attributen, zodat niemand de gegevens kan veranderen.

Als je ergens met IRMA inlogt met die attributen, dan weet de website ook zeker wie er inlogt.

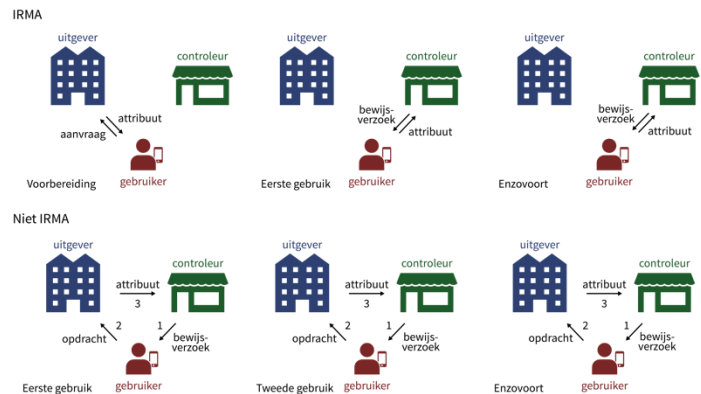
Hiernaast staat een overzicht hoe je IRMA aanmaakt:





Om vervolgens ergens in te loggen via IRMA vraagt de controleur de gegevens aan IRMA en komt er geen derde partij tussen. Op deze manier krijgt de derde partij geen gegevens waar jij inlogt.

Dit is een overzicht hoe je credentials koppelt en toont:



Wat IRMA ook privacy-vriendelijk maakt, is het feit dat IRMA gebruik maakt van data-minimalisatie. Als een gebruiker bij een derde partij wil inloggen of gegevens moet laten zien, laat IRMA alleen de noodzakelijke gegevens zien om het proces voort te zetten. Dit voorkomt dat derde partijen meer informatie krijgen dan nodig en hierdoor kan de gebruiker ook anoniem inloggen.

Als een webshop bijvoorbeeld alleen zeker wil weten dat je 18 jaar of ouder bent, dan kun je met IRMA inloggen en alleen het gegeven "ik ben ouder dan 18" doorgeven

Naast ervoor te zorgen dat door een handtekening de gegevens gemanipuleerd worden, wordt er ook vervaldatum gekoppeld aan de attributen. Dit zorgt ervoor de eens in de zoveel tijd de gegevens verversen worden.

[Bron: <https://privacybydesign.foundation/irma-uitleg/#garanties>]



6 PRAKTISCH GEDEELTE (HET INLOGSYSTEEM ZELF MAKEN)

Na alle kennis die we hebben opgedaan in dit profielwerkstuk werd het tijd om dan ook echt ons eigen veilige inlogstelsel te maken. Dit deden we met behulp van al onze antwoorden die we op de vorige deelvragen kregen.

We begonnen ons praktisch gedeelte met een stappenplan te maken met wat wij belangrijk vonden om in ons stelsel te hebben. We kozen ook om het stelsel te maken met PHP en MySQL, omdat dat voor ons het handigste was. Het is gratis en met school hadden we al eerder met deze talen gewerkt. We kozen voor een relationele database, omdat dit te creëren was met MySQL en we hier al eerder mee gewerkt hadden dus voor ons ideaal was. We hebben een taakverdeling gemaakt en zijn aan de slag gegaan. Jesse zou de aanmeld- en inlogformulieren maken, zorgen dat de gegevens veilig werden opgeslagen in de database, reCAPTCHA koppelen en het wachtwoord vergeten traject maken. Toine ging aan de slag om de authenticator app te koppelen en de website te hosten, zodat ie klaar voor gebruik was.

In hoofdstuk 2 staat dat een gebruiker zich moet registreren.

We begonnen ons stelsel met het maken van een script om een gebruiker een account te laten maken en te kunnen inloggen doormiddel van PHP. De gebruiker moest zijn naam, emailadres, telefoonnummer en zijn zelfgemaakte wachtwoord opgeven.

Deze gegevens moesten we opslaan, zodat de gebruiker de volgende keer weer terug kon keren met dezelfde gegevens. Deze gegevens sloegen we op in een relationele database, gemaakt met MySQL. Alleen wij hadden toegang tot deze database en daarmee tot de gegevens van de gebruiker.

In hoofdstuk 3 staat dat de gegevens veilig opgeslagen moeten worden.

In het script dat we gemaakt hebben, hebben we het wachtwoord gehasht waardoor we ook zelf niet konden inzien wat het wachtwoord van de gebruiker is. Bij iedere gebruiker werd een eigen unieke lange code toegewezen die alleen de computer kan decoderen.

wachtwoord

\$2y\$10\$twdHFqgSmnxwqKaSEQTPbu2d7/2wfaHaKFLFXyElvTy...

\$2y\$10\$1r9ca7PzUvr/U1wgFxYwunkdjg/KPpbQMv9VWoli.B...

\$2y\$10\$NJAjkk5R/jilD2dD5VoLw.eb4IKdAcLjKLe0/S4i2dB...

\$2y\$10\$cN.MfCJtip.DcabMLAaaNeIYpLH9VpgKIHHlw6b91c8...

(zo ziet de unieke lange code eruit die de gebruiker in de database toegewezen krijgt.)

Ook was het belangrijk in een veilig inlogstelsel om hackers tegen te gaan die gebruik maken van SQL Injecties. We hebben hiervoor code geschreven, zodat hackers niet zomaar informatie uit de database kunnen opvragen via de input velden. We maakten hier gebruik van prepared statements en `mysqli_real_escape_string()`.



In hoofdstuk 4 staan oplossingen om inloggen veiliger te maken.

Voor de veiligheid vonden we het belangrijk dat de gebruiker een goed sterk wachtwoord moest aanmaken. Daarom hebben we eisen gesteld aan de gebruiker z'n wachtwoord, zoals dat het wachtwoord minimaal 8 tekens lang moet zijn, minimaal 1 hoofdletter, 1 cijfer en 1 speciaal teken.

```
if($result){
    // Ingevoerde velden in de database zetten, + encrypten van het wachtwoord + tegen sql injecties.
    $result = $link -> prepare(" INSERT INTO 'account' ('voornaam', 'achternaam', 'emailadres', 'telefoonnummer', 'wachtwoord',
    'secretKey') VALUES (?, ?, ?, ?, ?, password_hash($ _SESSION['wachtwoord'], PASSWORD_DEFAULT), ' ' , ?) " or exit(mysql_error());
    $result -> bind_param("sssss", $ _SESSION['voornaam'], $ _SESSION['achternaam'], $ _SESSION['emailadres'],
    $ _SESSION['telefoonnummer'], $ _SESSION['secret']);
    $result->execute();
    $result->store_result();
    $result -> close();
    $link -> close();
    if($result){
        session_unset();
        $ _SESSION['login'] = true;
        header("Location: profiel.php");
    }
} else {
    $warning = "De code is onjuist";
}
```

(Zo ziet onze code eruit die we hebben geschreven tegen de SQL-injecties.)

Naast alle zelfgemaakte codering hebben we reCAPTCHA van het bedrijf Google in ons inlogstelsel geplaatst. We kregen een publieke sleutel en een privé sleutel die we moesten gebruiken in de code om reCAPTCHA te maken. We moesten een script tag toevoegen, waarmee Google dan de reCAPTCHA kon maken:

```
<script src='https://www.google.com/recaptcha/api.js' async defer ></script>
```

Daarna moesten we een div toevoegen waarin de reCAPTCHA opdracht stond:

```
<div class="g-recaptcha" data-sitekey="6LfYnigaAAAAALkaFhJP8h5lierWV1hL_b--OK1S"></div>
```

Als de gebruiker vervolgens de opdracht had voltooid en verder wilde, controleerde we in php of de opdracht goed uit was gevoerd door middel van deze code:

```
//reCAPTCHA
if(isset($_POST['g-recaptcha-response']) && !empty($_POST['g-recaptcha-response']))
{
    $secret = '6LfYnigaAAAAALkaFhJP8h5lierWV1hL_b--OK1S';
    $verifyResponse = file_get_contents('https://www.google.com/recaptcha/api/siteverify?secret='.$secret.'&response='.$_POST['g-recaptcha-response']);
    $responseData = json_decode($verifyResponse);
    if($responseData->success)
    {
    }
```

Om het inloggen nog veiliger te maken, hebben we Two-Factor-Authentication ingebouwd.

Wanneer de gebruiker zich gaat registreren moet zijn account gekoppeld worden aan een authenticator app. Dit kan door middel van de QR-code te scannen die in beeld komt nadat hij zijn gegevens heeft ingevuld. Hierna kan de gebruiker de code in de authenticator app invullen en is de authenticator app gekoppeld:



Authenticator app koppelen
Scan deze QR-code in de authenticator app en vul de code in.



VERZEND CODE

TERUG

(Zo ziet de QR-code eruit die gescand moet worden.)

Als de gebruiker hierna weer wil inloggen moet hij weer de code invullen via de authenticator app die aan zijn account gekoppeld is.

We hebben dit gemaakt door packages te downloaden van github via een php composer.

Deze packages hebben wij gebruikt:

- <https://github.com/antonioribeiro/google2fa> (qrCodeURL)
- <https://github.com/RobThree/TwoFactorAuth> (secret key, verify key)

De composer zet dan een mapje 'vendor', waarin het bestand 'autoload.php' staat, tussen alle bestanden. 'autoload.php' kan vervolgens alle packages lezen. Het is dus belangrijk dat je boven elk bestand waar je deze packages gebruikt de 'autoload.php' include. Ook zetten we 'use PragmaRX\Google2FA\Google2FA;' boven elk bestand om de packages te kunnen lezen. Vervolgens moeten we een secret key aanmaken die de gebruiker met de authenticator koppeld. De authenticator app zal dan voor die bepaalde secret key steeds per 30 seconde een code genereren die de gebruiker kan invullen.

Deze secret key genereren we op deze manier:

```
//Secret key aanmaken
$google2fa = new Google2FA();
$tfa = new RobThree\Auth\TwoFactorAuth('Pws');
$secret = $tfa->createSecret();
```

Voor de gebruikersvriendelijkheid maken wij gebruik van een QR-code die de secret key, de URL van de website en de email van de gebruiker bevat. Als de gebruiker deze scant hoeft hij niet de hele secret key invullen in de authenticator app.

We maken eerst een qrCodeUrl die de informatie bevat die in de QR-code staat:

```
//qrCodeUrl aanmaken met de secret key
$qrCodeUrl = $google2fa->getQRCodeUrl('inlogveiligheid.nl', $_SESSION['emailadres'], $_SESSION['secret']);
```



Vervolgens zetten we deze Url in de src van een image om de url te creëren:

```
<?php echo "<img src='https://chart.googleapis.com/chart?chs=300x300&cht=qr&chl=".$qrCodeUrl."&choe=UTF-8' alt=''>" ?>
```

Op deze manier heeft de gebruiker altijd 2 apparaten nodig om in te loggen en is dit dus een manier van 2FA.

Tenslotte hebben we voor de gebruikersvriendelijkheid ingebouwd dat als de gebruiker zijn wachtwoord vergeten is, kan hij zijn wachtwoord opnieuw kan instellen. Dit doen we door een mail te sturen waarin een URL staat met een code die ook in de database staat. Als de gebruiker naar de URL met de juiste code gaat kan hij zijn wachtwoord aanpassen.

```
function randomstring()
{
    $length = '20';
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charactersLength = strlen($characters);
    $randomstring = '';
    for ($i = 0; $i < $length; $i++) {
        $randomstring .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomstring;
}
```

Met deze code maken we een code die we in de database en de URL zetten. Elke code kan maar 1 keer in de database voorkomen en de code zal ook verwijderd worden uit de database als het wachtwoord is aangepast. We controleren ook nog of de code ouder is dan 3 uur. Zo ja? Kan de code niet gebruikt meer worden en wordt deze ook weer verwijderd uit de database. Het nieuwe wachtwoord moet 2 keer ingetypt worden en aan alle eisen voldoen die eerder behandeld zijn bij het registreren.



7 SAMENVATTING

Een inlogstelsysteem is een systeem waarbij de gebruiker zich identificeert (wie ben ik) en zich authenticereert (bewijzen dat je bent wie je zegt dat je bent).

Als de gebruiker heeft bewezen dat hij is wie hij zegt dat hij is, kan de site de gebruiker autoriseren om bepaalde dingen op de website te doen en kan de site gepersonaliseerd worden met bijvoorbeeld een eigen winkelmandje of speciale aanbiedingen. Om in te kunnen loggen moet een gebruiker zich altijd eerst registreren.

Uit ons onderzoek is gebleken dat een inlogstelsysteem veiliger wordt als gebruikers een sterk wachtwoord gebruiken. Hierdoor wordt het veel moeilijker voor brute force bots om het wachtwoord te kraken. Een andere manier om brute force attacks te voorkomen is reCAPTCHA. ReCAPTCHA herkent bots en zal ervoor zorgen dat deze bots niet door kunnen gaan met het kraken van het wachtwoord. Om inlogsystemen veilig te maken kan er ook gebruik gemaakt worden van biometrische eigenschappen om in te loggen, zoals vingerscan of gezichtsherkenning. Deze gegevens moeten echter wel veilig bewaard worden. En een inlogstelsysteem kan ook nog gebruik maken van Two Factor Authentication. Hierdoor moet de gebruiker nog een extra stap uitvoeren om te bewijzen dat hij echt is wie hij zegt dat hij is.

Naast veilig kunnen inloggen, is het ook belangrijk dat de inloggegevens veilig bewaard worden in de database. Het is belangrijk dat alleen mensen die het privilege hebben om de data te zien, de data te zien krijgen. Ook is het belangrijk dat de database een firewall heeft om bijvoorbeeld SQL-injecties en buffer overflows te voorkomen. Een andere manier om SQL-injecties te voorkomen is door speciale tekens te verbieden in velden waar gebruikers tekst in kunnen vullen. Als je geen speciale tekens wil verbieden, kan je gebruik maken van `mysqli_real_escape_string()` om de speciale tekens om te zetten naar een string, waardoor de speciale tekens geen SQL-injectie kunnen veroorzaken. Voordat je daarna een query uitvoert met de ingevulde data is het veilig om gebruik te maken van een prepared statement die checkt of er geen sprake is van een SQL-injectie. Als dan alles gecheckt is kan de query veilig uitgevoerd worden. Maar als je met een query data in de database wilt zetten is het belangrijk dat belangrijke gegevens zoals het wachtwoord encrypted worden. Hierdoor kan de hacker nog steeds niet meteen het wachtwoord zien als hij in de database is gekomen.

Ook hebben we gezien dat er gebruik gemaakt kan worden van inlogsystemen van bijvoorbeeld Google of Facebook. Dat is handig, maar heeft als nadeel dat Google of Facebook ook weet waar je inlogt. IRMA heeft dat nadeel niet. Dus vinden wij Irma de beste methode als je gebruik wilt maken van zo'n systeem.



8 KLOPT ONZE HYPOTHESE

In onze hypothese hadden wij aangeven hoe wij dachten dat een veilig inlogstelsysteem werkt en dat door 2FA een inlogstelsysteem veiliger zou worden. We zijn er tijdens het onderzoek achter gekomen dat er meer methodes zijn die inlogsystemen veilig kunnen maken, door bijvoorbeeld een veiliger wachtwoord te gebruiken, door de wachtwoorden te hashen of door reCAPTCHA te gebruiken. Ook zeiden we in de hypothese dat de database geen zwakheden mag hebben.

Het is ons gelukt om een inlogstelsysteem te maken, die zo veilig mogelijk is door gebruik te maken van wat we hebben geleerd. We hebben echter geen gebruik gemaakt van 2FA via sms. Daarentegen hebben we een authenticator app gebruikt, omdat dat veiliger bleek te zijn. We hebben ook geen gebruik gemaakt van Yubikey, omdat we die niet konden aanschaffen.

In deelvraag 3 is gebleken dat het van belang is een veilige database te creëren. We hebben in de praktijkopdracht gekozen voor een relationele database, aangezien dit gemakkelijk te gebruiken is en voor onze resultaten de beste optie was. Alleen wij kunnen de gegevens zien en om de gegevens veiliger te bewaren hashden we de wachtwoorden. Ook gebruikte we prepared statements en `mysqli_real_escape_strings` om sql-injecties te voorkomen.

9 CONCLUSIE

De hoofdvraag van ons PWS luidt "Hoe maak je een veilig inlogstelsysteem".

Met het beantwoorden van de deelvragen zijn we erachter gekomen dat er heel veel technieken zijn waarmee een hacker inloggegevens kan stelen, maar er gelukkig ook veel oplossingen zijn die inlogsystemen veiliger maken. Met ons onderzoek denken we de meeste en belangrijkste informatie te hebben gevonden om een veilig inlogstelsysteem te kunnen bouwen. In het praktische gedeelte hebben we daarna een veilig inlogstelsysteem gemaakt die gebruik maakt van technieken die we in de deelvragen hebben gevonden, zoals prepared statements en het tegenhouden van bots door reCAPTCHA. Alles met elkaar zorgde ervoor dat we een naar onze mening veilig inlogstelsysteem hebben neergezet en ons profielwerkstuk over inlogsystemen dus geslaagd is!