Adviesvraag
Abacus
voor de Kiesraad

Document: Kiesraad-Adviesvraag-Abacus-RAPPORT-v1.0.pdf
Auteurs:
dr.ing. Matthijs Koot
Tom Tervoort, MSc.
Projectmanager: Marc van den Berg
Ontvanger: Fleur van Leusden

Versie 1.0
28 juli 2025

Waarom Bureau Veritas Cybersecurity

Bureau Veritas Cybersecurity is uw specialist op het gebied van digitale veiligheid. Wij ondersteunen organisaties bij het in kaart brengen van risico's, het verbeteren van hun verdediging en het naleven van wet- en regelgeving. Onze dienstverlening omvat mens, proces en technologie: van awareness-trainingen en social engineering tot advies, compliance en technische securitytests.

We werken in IT-, OT- en IoT-omgevingen en ondersteunen zowel digitale systemen als connected producten. Met ruim 300 cybersecurity-specialisten wereldwijd combineren we diepgaande technische kennis met internationaal bereik. Bureau Veritas Cybersecurity is onderdeel van Bureau Veritas, wereldwijd actief in testen, inspectie en certificering.

Bureau Veritas Cybersecurity
E: cybersecurity@bureauveritas.com
W: cybersecurity.bureauveritas.com

Documentmanagement

Reviewers

Naam	Functie	Datum	Versie
Tom Tervoort	Security Specialist	2025-07-29	0.2

Wijzigingen

Versie	Datum	Initialen	Aanpassingen
0.1	2025-07-17	MK	Initiële conceptversie
0.2	2025-07-24	MK	Tweede conceptversie
1.0	2025-07-29	TT	Intern gereviewde definitieve versie

Inhoudsopgave

1. Managementsamenvatting

1.1. Uw vraag

U heeft een adviesvraag gesteld aan Bureau Veritas Cybersecurity over de Abacus-uitslagprogrammatuur. Deze software zal tijdens de gemeenteraadsverkiezingen van 2026 voor het eerst worden ingezet. Het onderzoek was bedoeld om uw hoofdvragen te beantwoorden, namelijk:

- “Als we Abacus laten pentesten, gezien de huidige opzet, waar moeten we dan in elk geval nog aan werken met welke prioriteit? Welk laaghangend fruit zien jullie nu al?”
- “Welke beveiligingsrisico’s zien jullie als meest relevant om afgedekt te hebben op technische wijze binnen Abacus, gezien de context waarbinnen Abacus gebruikt wordt?”

1.2. Het onderzoek

In de periode van 3 juli 2025 tot en met 11 juli 2025 heeft Bureau Veritas Cybersecurity onderzoek uitgevoerd ter beantwoording van de adviesvraag.

Het antwoord op de adviesvraag wordt gegeven in de vorm van verslaglegging van observaties en een aantal aanbevelingen. Op verzoek van de onderzoekers is besloten om daar in dit rapport geen risico-inschattingen aan te koppelen, omdat het niet goed mogelijk is om voldoende rekening te houden met de specifieke omstandigheden waarin Abacus wordt gebruikt. Het toevoegen van risico-inschattingen zou in dat geval kunnen resulteren in vertekende beeldvorming.

1.3. Samenvatting van de resultaten

Het verkiezingsproces wordt door de Kiesraad ondersteund met digitale middelen, waaronder uitslagprogrammatuur. Deze programmatuur draagt bij aan het kunnen vaststellen van een uitslag binnen de geldende termijnen.

Momenteel ontwikkelt de Kiesraad de software Abacus als (open-source) uitslagprogrammatuur, ter vervanging van de bestaande (closed-source) programmatuur OSV2020. Abacus wordt tijdens een verkiezing geïnstalleerd en gebruikt op Gemeentelijk Stembureau (GSB)-locaties, Hoofd Stembureau (HSB)-locaties en het Centraal Stembureau (CSB). Compromittering of ontregeling van Abacus zou kunnen resulteren in reputatieschade voor de Kiesraad en ondermijning van het publieke vertrouwen in het verkiezingsproces.

Softwarekwetsbaarheden

Tijdens het onderzoek zijn in het ontwerp of de (kern-)broncode van de ontwikkelversie van Abacus zelf geen ernstige tekortkomingen geconstateerd. Bij een steekproefsgewijze[1] controle van invoervalidatie, autorisatiecontroles en veelvoorkomende softwarekwetsbaarheden bleek Abacus robuust.

Het antwoord op de vraag of er ‘laaghangend fruit’ is aangetroffen luidt dus ontkennend: er zijn geen kwetsbaarheden geïdentificeerd die, binnen de specifieke context waarin Abacus wordt gebruikt, een hoog of kritiek risico opleveren.

[1] Een volledige ‘pentest’ maakte geen deel uit van dit onderzoek.

Wel zijn er observaties gedaan die volgens de onderzoekers aandacht verdienen. Een deel hiervan betreft inzichten die bij de Kiesraad reeds bekend zijn, maar in dit rapport niet achterwege kunnen blijven.

HTTPS

De belangrijkste observatie is dat Abacus geen gebruik maakt van HTTPS: zie Aandachtspunt 10 op pagina 31. HTTPS zorgt ervoor dat gegevens die gebruiker invoeren versleuteld over het netwerk worden verstuurd, in plaats van onversleuteld. Er zijn legitieme redenen voor Abacus om geen HTTPS te gebruiken, en het *kan* passend zijn binnen de context waarin Abacus wordt gebruikt. Het gevolg van afwezigheid van HTTPS is evenwel dat de mogelijkheid (niet te verwarren met *waarschijnlijkheid*) bestaat dat een aanvaller die kortstondig toegang heeft tot het lokale netwerk bij een stembureau waar Abacus wordt gebruikt, wachtwoorden van Abacus-gebruikers kan onderscheppen, de werking van de Abacus-frontend kan verstoren, of andere ongewenste activiteit te veroorzaken. Dit kan met laagdrempelig (hacker)gereedschap en vergt geen diepgaande kennis.

De aansluit- en gebruiksvoorschriften van Abacus bevatten bepalingen die, mits ze worden nageleefd, de kans beperken dat dit zich in de praktijk voordoet. Het naleven van die voorschriften is de verantwoordelijkheid van de organisatie die Abacus gebruikt, zoals een gemeente. Ook is de impact in beginsel beperkt tot (de rol van het) stembureau waar dit zich voordoet: voor een GSB zal de impact dus beperkt zijn tot het lokale niveau en een beperkt aantal personen. Dat neemt niet weg dat Abacus-gebruikers in het algemeen de verwachting kunnen hebben dat gegevens zoals hun wachtwoord niet onversleuteld over een netwerk worden verstuurd, ook als het netwerk lokaal en gescheiden is. Er moet rekening worden gehouden met de realiteit dat een deel van de gebruikers een wachtwoord zullen instellen dat zij ook al gebruiken voor privé- en/of werksystemen, en dat het onderscheppen van dat wachtwoord voor hen consequenties kan hebben buiten Abacus om. Dat maakt afwezigheid van HTTPS een lastig vraagstuk. Er is geen simpele volledige oplossing voor.

Toeleveringsketen

Een ontregeling of compromittering van Abacus bij één gemeentelijk stembureau, bijvoorbeeld door een kwaadwillende die kortstondig fysieke toegang heeft tot de lokale apparatuur waarop Abacus wordt gebruikt, is indien gedetecteerd goed af te handelen. Desnoods kunnen de fysieke stembiljetten handmatig opnieuw worden geteld en de originele papieren processen-verbaal worden gebruikt of opnieuw worden ingevoerd op andere infrastructuur. De impact van een incident bij één gemeentelijk stembureau zal dan ook beperkt zijn.

De apparaten waarop gemeenten Abacus installeren worden niet landelijk centraal ingekocht: gemeenten zijn zelf verantwoordelijk voor inkoop van hun IT-middelen. Een aanvaller kan dus niet via compromittering van een enkele hardware-toeleverketen een landelijke verstoring van Abacus op gemeentelijke apparaten veroorzaken.

Wel zou een aanvaller die grote impact wil veroorzaken, zoals het (tijdelijk) verstoren van de vaststelling van een verkiezingsuitslag, zich kunnen richten op het ontwikkel-, bouw- en distributieproces van Abacus. Daaronder vallen bijvoorbeeld de Abacus-repository op het GitHub-account van de Kiesraad zelf, de toeleverketens van externe software-afhankelijkheden (‘dependency’s’) waarvan Abacus gebruikmaakt[2], en de omgeving waarop de Kiesraad de Abacus-broncode omzet in een uitvoerbaar programma dat vervolgens beschikbaar wordt gesteld aan gebruikers (zoals gemeenten).

Een inventarisatie van externe software-afhankelijkheden van de frontend en backend van Abacus resulteerde in een lijst van 105 unieke GitHub-accounts waarvan de eigenaar externe code kan aanpassen die door Abacus wordt gebruikt. Als een aanvaller één van deze accounts zou overnemen, zou deze daarmee in principe heimelijk code kunnen proberen toe te voegen aan Abacus. De kans dat zulk misbruik zich voordoet wordt is vermoedelijk zeer

[2]De Kiesraad houdt hiervan een Software Bill Of Materials (SBOM) bij van Abacus.

laag, maar de impact zou zeer groot zijn: om die reden is waakzaamheid geboden. Specifieke aanbevelingen hierover staan beschreven in Aandachtspunt 7 op pagina 23.

Andere aanbevelingen

Enkele aanbevelingen zien op functionaliteit die in de ontwikkelversie van Abacus nog niet aanwezig was, maar door de Kiesraad al wordt onderkend als ‘todo’. Een voorbeeld hiervan is de afwezigheid van airgapdetectie in de frontend, die al wel plaatsvindt in de backend: zie Aandachtspunt 3 op pagina 16. Het toevoegen van airgapdetectie in de frontend verkleint de kans nog verder dat zich gebruik van Abacus voordoet terwijl een internetverbinding actief is, om strijd met de aansluit- en gebruiksvoorschriften. Een ander voorbeeld is de afwezigheid van bescherming tegen herhaaldelijk (geautomatiseerd) raden van wachtwoorden: zie Aandachtspunt 11 op pagina 34. Het toevoegen van een (getrapte) blokkade helpt voorkomen dat een aanvaller die zich op het lokale/gescheiden netwerk bevindt, via het raden van wachtwoorden toegang kan krijgen tot Abacus-accounts.

Andere aanbevelingen zien op productieversies van Abacus die de Kiesraad voor een verkiezing maakt en distribueert. Zo is de ontwikkelversie van Abacus niet digitaal ondertekend met een code-signing certificaat, waardoor een maatregel onbenut blijft die de gebruiker kan helpen vaststellen dat deze beschikt over de authentieke versie van de software: zie Aandachtspunt 2 op pagina 15. Voor de productieversie is zulke ondertekening zeer gewenst. Daarnaast zou (op termijn) eventueel kunnen worden gestreefd naar ‘reproducible builds’ van productieversies van Abacus: zie Aandachtspunt 6 op pagina 19. Dit zou het voor derden mogelijk maken om zelfstandig en onafhankelijk te controleren of de versie van de Abacus-software die de Kiesraad distribueert tijdens een verkiezing, overeenkomt met de openbaar gepubliceerde broncode van die versie. Het realiseren van ‘reproducible builds’ is echter technisch ingewikkeld, en als het eenmaal is geslaagd en openbaar wordt gecommuniceerd, kan het bij derden de verwachting scheppen dat toekomstige Kiesraad-software ook altijd als ‘reproducible build’ beschikbaar komt.

Afzenderauthenticatie

Tot slot bevat dit rapport ook suggesties voor een eventueel ondertekenmechanisme voor uitslagbestanden om ‘afzenderauthenticatie’ te realiseren: zie Aandachtspunt 9 op pagina 27. Afzenderauthenticatie is niet relevant voor de gemeenteraadsverkiezingen waarvoor de eerste versie van Abacus wordt ontwikkeld, maar kan dat wel zijn voor landelijke verkiezingen. Dat mechanisme kan een aanvullende waarborg bieden tegen scenario’s waarin uitslagbestanden worden gemanipuleerd alvorens ze te importeren naar het HSB of CSB. De waarschijnlijkheid van die scenario’s is vermoedelijk beperkt, maar het adresseren ervan kan bijdragen aan het publieke vertrouwen in het verkiezingsproces, of tenminste een gelegenheid voor ondermijning daarvan wegnemen.

2. Technische samenvatting

Dit hoofdstuk biedt een overzicht van de bevindingen die zijn opgenomen in dit rapport en welke risico's hieraan zijn verbonden. Daarnaast wordt uitgelicht welke bevindingen de hoogste prioriteit hebben om te worden opgelost. Voor meer informatie zie de daadwerkelijke bevindingen verder in dit rapport.

2.1. Overzicht van bevindingen

De onderstaande tabel bevat een overzicht van alle bevindingen in dit rapport.

Bevinding	Onderwerp	Referentie
Aandachtspunt 1	Overweeg technisch te controleren dat de server niet als invoer-station wordt gebruikt	Pagina 14
Aandachtspunt 2	Pas identiteitsgebonden code-signing toe op binary's	Pagina 15
Aandachtspunt 3	Voeg airgapdetectie toe aan de frontend	Pagina 16
Aandachtspunt 4	Overweeg een ICMP-test toe te voegen aan de airgapdetectie op de backend	Pagina 17
Aandachtspunt 5	Zorg dat de releaseversie van Abacus geen test-secrets bevat in broncode en/of binary builds	Pagina 18
Aandachtspunt 6	Overweeg 'reproducible builds' van Abacus als punt op de horizon voor de langere termijn	Pagina 19
Aandachtspunt 7	Supply chain: dreiging van compromittering van een dependency-ontwikkelaar	Pagina 23
Aandachtspunt 8	Twee transitieve dependency's van backend hebben status 'unmaintained'	Pagina 26
Aandachtspunt 9	Suggesties voor ondertekenmechanisme uitslagbestanden	Pagina 27
Aandachtspunt 10	Aanbevelingen inzake onderscheppen netwerkverkeer	Pagina 31
Aandachtspunt 11	Adviezen met betrekking tot wachtwoordbeleid	Pagina 34

Tabel 2.1: Overzicht van bevindingen

3. Beschrijving van de opdracht

Dit onderzoek is gebaseerd op de offerte met referentie ‘10656656.02-SOW-Kiesraad-Adviesvraag Abacus’ waarin de volgende beschrijving van de opdracht is opgenomen:

> *Adviesvragen*
>
> - *Als we Abacus laten pentesten, gezien de huidige opzet, waar moeten we dan in elk geval nog aan werken met welke prioriteit? Welk laaghangend fruit zien jullie nu al?*
> - *Welke beveiligingsrisico’s zien jullie als meest relevant om afgedekt te hebben op technische wijze binnen Abacus, gezien de context waarbinnen Abacus gebruikt wordt?*
>
> *Uitgangspunten*
>
> *Wij verwachten een advies welke past bij de rol die Abacus heeft in het verkiezingsproces en binnen de juiste context. Daarbij verwachten wij dat de opdrachtnemer let op zaken als wat realistische risico’s zijn en tevens de rol die andere partijen hebben binnen het verkiezingsproces.*
>
> *Abacus is een onderdeel van het verkiezingsproces, maar zou geen single point of failure moeten zijn. Daarnaast word Abacus bestuurd door veelal vrijwilligers met verschillende smartphones welke niet managed zijn door gemeenten.*
>
> *Wij verwachten dat de opdrachtnemer hiervan op de hoogte is en eventueel vragen stelt als zaken onduidelijk zijn. Wij verwachten een advies te ontvangen waarbinnen bovenstaande is meegenomen.*

3.1. Scope van het onderzoek

De scope van het onderzoek omvatte de Abacus-software, binnen de specifieke context waarin deze wordt gebruikt.

3.2. Vooraf verstrekte informatie

Kiesraad heeft Bureau Veritas Cybersecurity vooraf de informatie in tabel 3.1 op de pagina hierna verstrekt.

Identificatie	Beschrijving
`https://github.com/kiesraad/abacus/`	Abacus: broncode, binary's en documentatie
`Bijlagen I tm III rapport risico analyse Abacus 1.0.pdf`	Document: 'Bijlagen I t/m III Risico Analyse Abacus'
`RA Abacus 2024-2025 rapportage.pdf`	Presentatie: 'Rapportage risico analyse - Abacus'
`06573 Kiesraad Aansluit- en gebruiksvoorschriften_v3.pdf`	Aansluit- en gebruiksvoorschriften voor stembureaus
`06573 Kiesraad Samenwerkingsprotocol_v3.pdf`	Samenwerkingsprotocol Wet programmatuur verkiezingsuitslagen
`CLAbot 28 oct 2024.md`	Notitie over CLA bot
`onboarding.md`	Instructies nieuwe medewerkers
`plan.md`	Plan van aanpak Abacus
`wachtwoord-werkwijze.md`	Geplande wachtwoordwerkwijze

Tabel 3.1: Verstrekte informatie

Daarnaast is lopende het onderzoek aanvullende informatie beschikbaar gesteld. De lokaal bij Bureau Veritas Cybersecurity opgeslagen informatie zal na afronding van dit project door Bureau Veritas Cybersecurity op een veilige wijze worden vernietigd.

3.3. Doel van het onderzoek

Het doel van dit onderzoek was het beantwoorden van de adviesvragen van de Kiesraad en het aandragen van mogelijke verbeteringen inzake Abacus.

3.4. Rapportage

De managementsamenvattingen en technische samenvattingen dienen als overzicht van de onderzoeksresultaten voor respectievelijk het algemeen en technisch management. In de daaropvolgende hoofdstukken worden de gedetailleerde resultaten beschreven, ondersteund door reproduceerbare bevindingen. Deze hoofdstukken zijn bedoeld om technisch personeel te begeleiden in het reproduceren en mitigeren van de gevonden problemen.

3.5. Onderzoeksmethode

Het onderzoek is uitgevoerd als maatwerk op basis van bureauonderzoek, technische inspecties en communicatie met betrokkenen van de Kiesraad.

3.6. Wijzigingen van systemen

Er zijn door de Kiesraad geen systeemwijzigingen doorgevoerd voor dit onderzoek.

3.7. Beperkingen

Een beveiligingsonderzoek biedt waardevol inzicht met betrekking tot de IT-beveiliging van het doelsysteem of -applicatie. Een dergelijk onderzoek is echter slechts een momentopname en biedt geen garantie voor de veiligheid van de IT-omgeving en data. Voortdurend worden er nieuwe aanvalstechnieken ontwikkeld en ontdekt. Daarnaast kan een geringe aanpassing aan de IT-omgeving eenvoudig nieuwe kwetsbaarheden introduceren. Minstens zo belangrijk als het technologische aspect is de rol van processen, procedures en de menselijke factor in informatiebeveiliging. Dit rapport geeft slechts een overzicht van gevonden kwetsbaarheden en is daarmee niet bedoeld als een garantie.

4. Abacus: uitslagprogrammatuur

Dit hoofdstuk beschrijft de context van Abacus en de observaties en aanbevelingen die tijdens dit onderzoek zijn gedaan.

Er is gebruikgemaakt van Abacus-versie 0.1.0-alpha.20250706.38fe459[1] van 6 juli 2025. Dit betreft een ontwikkelversie. Abacus wordt nog actief ontwikkeld en zal voor de Gemeenteraadsverkiezingen 2026 (‘GR2026’) voor de eerste keer een 1.0-release bereiken.

4.1. Adviesvraag

De adviesvraag die de Kiesraad aan Bureau Veritas Cybersecurity heeft voorgelegd bestaat uit twee deelvragen:

- Als we Abacus laten pentesten, gezien de huidige opzet, waar moeten we dan in elk geval nog aan werken met welke prioriteit? Welk laaghangend fruit zien jullie nu al?
- Welke beveiligingsrisico’s zien jullie als meest relevant om afgedekt te hebben op technische wijze binnen Abacus, gezien de context waarbinnen Abacus gebruikt wordt?

Daarbij waren de volgende uitgangspunten aangedragen:

> *Wij verwachten een advies welke past bij de rol die Abacus heeft in het verkiezingsproces en binnen de juiste context. Daarbij verwachten wij dat de opdrachtnemer let op zaken als wat realistische risico’s zijn en tevens de rol die andere partijen hebben binnen het verkiezingsproces.*
>
> *Abacus is een onderdeel van het verkiezingsproces, maar zou geen single point of failure moeten zijn. Daarnaast wordt Abacus bestuurd door veelal vrijwilligers met verschillende smartphones welke niet managed zijn door gemeenten.*

Dit hoofdstuk beschrijft de context en uitkomsten.

4.2. Context van Abacus

In Nederland wordt tijdens verkiezingen met papieren stembiljetten gestemd. Na de stemopname worden de stemmen geteld, waarna papieren processen-verbaal worden opgesteld van de uitkomsten. Dit papieren spoor vormt de ‘single source of truth’ (grondwaarheid) voor verkiezingsuitslagen.

Het verkiezingsproces wordt door de Kiesraad ondersteund met digitale middelen, waaronder uitslagprogrammatuur. Dit draagt bij aan het kunnen vaststellen van een uitslag binnen de geldende termijnen.

De Kiesraad ontwikkelt momenteel in eigen beheer de (open-source) uitslagprogrammatuur[2] Abacus, die op termijn het bestaande (closed-source) pakket OSV-U zal vervangen. De Kiesraad stelt de uitslagprogrammatuur beschikbaar aan gebruikers (stembureaus).

[1] Bron: `https://github.com/kiesraad/abacus/releases/tag/v0.1.0-alpha.20250706.38fe459`

[2] Dit moet niet worden verward met het niet-omomstreden concept ‘online stemmen’, ofwel via internet op afstand stemmen. Het actuele standpunt van de regering over online stemmen, vastgelegd in Handelingen 2021/22, nr. 3408, is dat online stemmen onaanvaardbare (cyber)risico’s voor de integriteit van het stemproces met zich meebrengt, en de regering de verkiezingen hier niet aan wil blootstellen. De auteurs van dit rapport onderschrijven dat standpunt.

De broncode en documentatie van Abacus worden door de Kiesraad openbaar gepubliceerd, zodat een ieder deze kan inspecteren.

Bij twijfels over de betrouwbaarheid van de uitslagprogrammatuur of de uitslagen kan worden teruggegrepen op het papieren spoor, en zou — ultimo — een volledige hertelling kunnen worden gedaan. De processen-verbaal daarvan kunnen in beginsel ook zonder gebruikmaking van de uitslagprogrammatuur worden gecommuniceerd.

Op de uitslagprogrammatuur is wet- en regelgeving van toepassing, waaronder de ‘Wet programmatuur verkiezingsuitslagen’ die tot doel heeft ‘dat het gebruik van uitslagprogrammatuur bij de vaststelling van de verkiezingsuitslag betrouwbaar, veilig, transparant en controleerbaar is’. Op 1 juli 2025 is een wijziging van het Kiesbesluit gepubliceerd in het Staatsblad: ‘Besluit van 23 juni 2025 tot wijziging van het Kiesbesluit in verband met de Wet programmatuur verkiezingsuitslagen’ (voorts: Besluit). Die wijziging codificeert de volgende nieuwe transparantie-eisen:

> *a.de programmatuur wordt als open source ontwikkeld en maakt gebruik van open standaarden;*
>
> *b.de standaardprogrammatuur waarvan gebruik wordt gemaakt is vrij verkrijgbaar;*
>
> *c.het intellectueel eigendom van de maatwerkprogrammatuur berust bij de Staat;*
>
> *d.de uitslagprogrammatuur is geschreven in een programmeertaal, waarvoor een door een actieve gemeenschap onderhouden open source programma om broncode naar machinecode te vertalen beschikbaar is; en*
>
> *e.de functies die in de uitslagprogrammatuur voorzien in het optellen van stemtotalen en het bepalen van de zetelverdeling zijn als zodanig herkenbaar en kunnen zelfstandig getest worden.*

Gebruikers van de programmatuur, zoals tijdens GR26 de gemeenten, dragen een eigen verantwoordelijkheid voor het correcte en voldoende veilige gebruik van uitslagprogrammatuur op hun eigen apparaten. Die verantwoordelijkheid vloeit voort uit de ‘Wet programmatuur verkiezingsuitslagen’ en het bijbehorende ‘Besluit programmatuur verkiezingsuitslagen’, op basis waarvan de Kiesraad in overleg met belanghebbenden een samenwerkingsprotocol heeft opgesteld.

Het samenwerkingsprotocol omvat aansluit- en gebruiksvoorwaarden waaraan de gebruikers dienen te voldoen voor een betrouwbare werking en beveiliging van de uitslagprogrammatuur. Deze voorschriften gelden voor de decentrale locaties en apparaten waar de gebruikers Abacus installeren en gebruiken, en omvatten ook technische vereisten aan de configuratie van systemen. Het is belang van deze voorschriften actueel zijn: in het voornoemde Besluit is daartoe een evaluatiebepaling opgenomen die regelt dat de Kiesraad deze voorschriften na elke verkiezing evalueert.

Het naleven van deze voorschriften is de verantwoordelijkheid van de gebruiker. Het college van burgemeester en wethouders van elke gemeente moet voorafgaand aan de daadwerkelijk stemming een verklaring afgeven aan de Kiesraad over het voldoen aan de aansluitvoorwaarden.

Abacus *ondersteunt* het verkiezingsproces, maar gegevens die via Abacus worden verwerkt zijn *niet* de grondwaarheid voor de uitkomsten: de fysieke stembiljetten en fysieke processen-verbaal zijn de grondwaarheid (‘single source of truth’). Mocht er sprake zijn van onregelmatigheden in invoer of uitvoer van Abacus dan is, in het ultieme geval, handmatig (her)tellen mogelijk. De intentie van de Kiesraad is om bepaalde (beperkte) detectie van onregelmatigheden te laten doen door Abacus zelf, bijvoorbeeld om (menselijke) invoerfouten tijdens de initiële (eerste) invoer van stemtotalen te signaleren. Daarnaast vindt bij elke verkiezingen een audit plaats op basis van steekproeven.

4.3. Beschikbare informatie

De Kiesraad publiceert broncode en documentatie van Abacus openbaar op GitHub[3]:

```
https://github.com/kiesraad/abacus
https://kiesraad.github.io/abacus-documentatie/
```

De Kiesraad heeft daarnaast een visuele weergave van het ontwerp van Abacus openbaar gepubliceerd op Figma:

```
https://www.figma.com/design/xHDfsv69Nhmk3IrWC0303B/Public---Kiesraad---Abacus-Software-voor-verkiezingsuitslagen-en-zetelverdeling
```

Ook heeft de Kiesraad voorafgaand aan het onderzoek een risicoanalyse beschikbaar gesteld die (nog) niet openbaar is gepubliceerd. Uit bestudering van de risicoanalyse ontstond het beeld dat de Kiesraad reeds uitvoerig, en in samenspraak met andere partijen, aandacht heeft besteed aan het inventariseren van mogelijke dreigingen, risico-inschattingen, en welke maatregelen wel of niet mogelijk (en proportioneel) zijn.

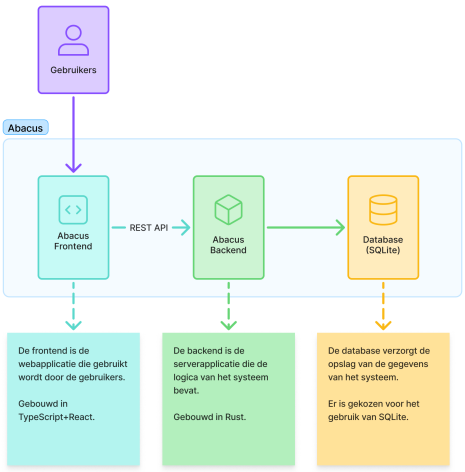
Bureau Veritas Cybersecurity heeft ervoor gekozen om de risicoanalyse niet als afbakening te gebruiken, maar om met open vizier naar Abacus te kijken en zelf keuzes te maken in wat in dit rapport wordt beschreven, aansluitend bij de adviesvraag. Dit rapport bevat daardoor (ook) inzichten die door de Kiesraad reeds in haar eigen risicoanalyse waren onderkend. In die gevallen biedt dit rapport aanvullende aanknopingspunten bij die inzichten.

Tot slot heeft de Kiesraad lopende het onderzoek vragen beantwoord, en aanvullende documentatie beschikbaar gesteld: zie tabel 3.1 op pagina 6.

4.4. Opzet van Abacus op hoofdlijnen

Abacus bestaat uit een backend die primair is geschreven in Rust, en een frontend die primair is geschreven in TypeScript/React. De backend wordt bij een stembureau op een gescheiden lokaal netwerk geïnstalleerd op een apparaat dat als server is aangewezen, en maakt een webinterface beschikbaar aan het lokale netwerk. Die webinterface is de frontend, die vanaf apparaten die als client (zoals invoerstation) zijn aangewezen wordt gebruik via een normale webbrowser. Dit is terug te zien in Figuur 4.1 op de pagina hierna.

[3] Over het gebruik van GitHub, dat eigendom is van Microsoft, als bron voor de Kiesraad-code bevat dit rapport geen commentaar.



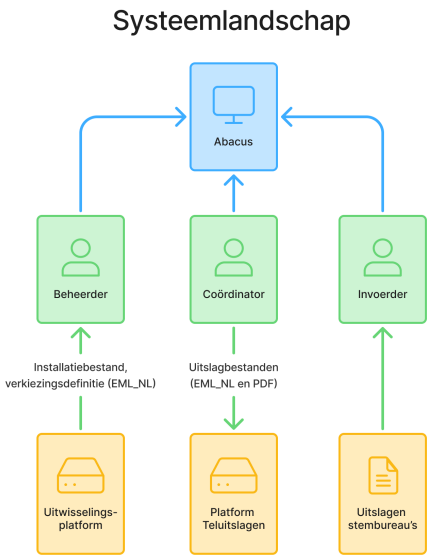
Figuur 4.1: Architectuur van Abacus. Bron: Kiesraad

Er zijn drie rollen in Abacus:

- Beheerder
- Coördinator
- Invoerder

Dit is ook terug te zien in het systeemlandschap[4] dat is weergegeven in Figuur4.2 op de volgende pagina.

[4] Bron (bezocht op 3 juli 2025): `https://github.com/kiesraad/abacus/blob/main/documentatie/softwarearchitectuur/Overzicht.md`



Figuur 4.2: Systeemlandschap van Abacus. Bron: Kiesraad

De Beheerder verkrijgt van het uitwisselingsplatform (op Diginetwerk) van de Kiesraad het installatiebestand van Abacus, de verkiezingsdefinitie (in EML-formaat) en kandidatenlijst (idem). Na de stemopname voert een Invoerder de gegevens van het papieren processen-verbaal met de uitslagen van stembureaus in, waarna een tweede Invoerder hetzelfde (doet ter voorkoming van invoerfouten). De Coördinator exporteert vervolgens het uitslagbestand in EML- en PDF-formaat, en importeert deze in het Platform Teluitslagen (op Diginetwerk) van de Kiesraad.

De Invoerder gebruikt Abacus alleen op het moment dat uitslagen worden ingevoerd, tijdens zittingen van het GSB, HSB of CSB. De Beheerder en Coördinator gebruiken Abacus meer frequent, over een langere periode, en hebben relatief gevoeligere toegangsrechten dan de Invoerder.

4.5. Aanpak van onderzoek

De documentatie en delen van de broncode zijn bestudeerd, en er zijn steekproefsgewijs enkele technische inspecties uitgevoerd om conform de adviesvraag eventueel ‘laaghangend fruit’ te identificeren dat mogelijk zou kunnen terugkomen in een pentest die later wordt uitgevoerd. Een pentest maakte geen deel uit van het huidige onderzoek.

De Abacus-software is door de onderzoekers ingebruikgenomen op een Linux-, macOS- en Windows-omgeving. De webinterface van de software is kort verkend met een proxy (Burp Suite Professional) tussen de browser en de backend om verkeer te kunnen observeren en steekproefsgewijs enkele tests uit te voeren op invoervalidatie (cross-site-scripting, external entity expansion, ...) en autorisatiecontroles[5]. De logging van Abacus is visueel

[5]Zie hierbij ook ‘Autorisatiematrix Gemeenteraadsverkiezingen’: `https://github.com/kiesraad/abacus/blob/main/documentatie/`

geïnspecteerd.

Daarnaast is met `cargo audit` en `npm audit` een vluchtige controle uitgevoerd op eventueel gebruik van kwetsbare software-afhankelijkheden (voorts: ‘dependency’s’) door de backend (Rust) en frontend (TypeScript/React).

Ook zijn delen van de broncode handmatig geïnspecteerd.

4.6. Dreigingsmodel

Een vraag is wat het dreigingsmodel is waar de Abacus weerstand tegen zou moeten (helpen) bieden: wat kan wel en niet van Abacus worden verwacht? Welke dreigingen worden geadresseerd door de aansluit- en gebruiksvoorschriften die gebruikers van Abacus, zoals gemeenten, *zelf* moeten naleven?

Evident *buiten scope* is bescherming tegen installatie en/of gebruik op gecompromitteerde apparaten bij stembureaus: hier kan Abacus zelf niet technisch tegen beveiligen.

Ook buiten scope is bescherming tegen een geautoriseerde systeembeheerder die te kwader trouw handelt tijdens installatie van de Abacus-server, of een geautoriseerde Abacus-beheerder die te kwader trouw handelt tijdens het gebruik van de functionaliteit die in Abacus alleen toegankelijk is voor de beheerder-rol (zoals het importeren van `.eml`-bestanden van verkiezingsdefinities en kandidatenlijsten).

Evident *binnen scope* is het waarborgen van de rollenscheiding van gebruikers van Abacus voor zover dat technisch kan, bijvoorbeeld om gestand te doen aan de integriteitswaarborg van tweevoudige invoer van processen-verbaal: deze dienen door twee verschillende personen (invoerders) te worden ingevoerd.

In algemene zin mag van Abacus worden verwacht dat deze geen ernstige kwetsbaarheden bevat, ongeacht de vraag of er voor die kwetsbaarheden een plausibel scenario is waarin uitbuiting zou kunnen leiden tot beïnvloeding van een (lokale) verkiezingsuitslag of (tijdelijke) verstoring van het digitaal invoeren van processen-verbaal. Die verwachting vloeit voort uit het feit dat Abacus nu eenmaal een rol speelt in het verkiezingsproces, en dat het waarborgen van publiek vertrouwen in dat proces van groot belang is.

Daarnaast *kan* Abacus een rol spelen bij naleving van bepaalde aansluit- en gebruiksvoorschriften, ook al ligt de verantwoordelijkheid voor naleving daarvan bij de gebruiker (zoals een gemeente). Zo stelt één van de voorschriften dat Abacus dient te worden gebruikt op een netwerk dat geen koppeling heeft met reguliere IT-infrastructuur of het internet (het netwerk moet ‘airgapped’ zijn):

> *Gebruik een fysiek gescheiden netwerk zonder koppeling met reguliere IT-infrastructuur of het internet.*

De ontwikkelversie van Abacus bevat reeds een vorm van detectie van internetconnectiviteit — ‘airgapdetectie’ (zie ook paragraaf 4.7.3 op pagina 15):

```
2025-07-09T11:37:24.515611Z ERROR abacus::airgap::detect: Airgap violation detected, abacus is
 connected to the internet!
2025-07-09T11:37:36.192295Z ERROR abacus::airgap::middleware: Blocking request due to airgap
violation: /api/user/whoami
[...]
2025-07-11T08:32:03.883535Z  INFO abacus::airgap::detect: Airgap violation resolved, abacus is
 no longer connected to the internet.
2025-07-11T08:33:03.931329Z ERROR abacus::airgap::detect: Airgap violation detected, abacus is
 connected to the internet!
```

---

use-cases/autorisatiematrix.md

```
2025-07-11T13:37:48.113451Z  WARN abacus::airgap::detect: Airgap violation detected, abacus is
 still connected to the internet.
```

Hiermee ondersteunt Abacus de gebruikers met naleving van een van de gebruikersvoorschriften.

4.7. Observaties

In deze sectie worden observaties beschreven die zijn gedaan tijdens het onderzoek ter beantwoording van de adviesvraag.

Belangrijk is dat deze observaties moeten worden beschouwd binnen de context van het gebruik van Abacus, waaronder de in dit hoofdstuk al eerder benoemde aansluit- en gebruiksvoorschriften.

Aangezien de adviesvraag (ook) ziet op de vraag welke bevindingen mogelijk naar voren zullen worden gebracht tijdens een pentest van Abacus, zijn ook observaties aanwezig die een risico (kunnen) vormen dat — indachtig die context — mogelijk als restrisico zou kunnen worden geaccepteerd.

Ten eerste wordt opgemerkt dat tijdens het onderzoek geen ernstige tekortkomingen ontdekt in het ontwerp of de broncode van Abacus. Bij steekproefsgewijze — maar dus niet uitputtende — tests op invoervalidatie en autorisatiecontroles bleek Abacus correct te functioneren.

Voor XML-parsing van EML-documenten gebruikt de backend de Rust-package `quick-xml`. Deze parser biedt geen ondersteuning voor DTD, waardoor geen kwetsbaarheid kan bestaan voor bijvoorbeeld XXE-aanvallen.

Voor gegevensopslag in een lokale SQLite-database gebruikt de backend de Rust-package `sqlx`. De gegevens worden opgeslagen in het bestand `db.sqlite3`, in de directory waarin de Abacus-binary zich bevindt op het apparaat dat als server dient. Dit bestand bevat dan onder meer de gebruikers en verkiezingsuitslagen.

Als hashingalgoritme voor wachtwoorden van Abacus-gebruikers wordt Argon2id gebruikt, dat (ruimschoots) volstaat voor het doel.

De volgende subsecties beschrijven observaties waarbij een aanbeveling is opgenomen.

4.7.1. Server mag niet worden gebruikt als invoerstation: technisch controleren?

Een gebruiksvoorschrift luidt:

> *Gebruik de servercomputer niet als invoerstation. Gebruik clientcomputers voor het invoeren van de gegevens.*

Dit voorschrift dient voor scheiding van server en clients, wat in het algemeen een ‘good practice’ is om toegang tot, en daarmee blootstelling van, de server te beperken. In de ontwikkelversie is geen controle aanwezig die dit technisch afdwingt. Dit is eenvoudig te realiseren door inlogpogingen van niet-beheerders op de Abacus-frontend te blokkeren indien deze afkomstig zijn van `127.0.0.1`, `::1` of het IP-adres van de netwerkinterface van de server. Dit zou kunnen worden overwogen.

AANDACHTSPUNT 1	10656656.01-R1
Onderwerp	Overweeg technisch te controleren dat de server niet als invoerstation wordt gebruikt
Van toepassing op	Abacus (backend)

Wordt vervolgd op de volgende pagina…

AANDACHTSPUNT 1 (vervolg)	10656656.01-R1
Beschrijving	Een van de gebruiksvoorschriften luidt ‘Gebruik de servercomputer niet als invoerstation. Gebruik clientcomputers voor het invoeren van de gegevens’. Dit wordt in de ontwikkelversie van Abacus niet technisch afgedwongen. Een controle op naleving van dat voorschrift is technisch (redelijk) goed mogelijk.
Aanbeveling	Overweeg in de backend een controle toe te voegen die voorkomt dat de server waarop de backend actief is, zelf als invoerstation (client) wordt gebruikt. Als een harde blokkade om praktische redenen onwenselijk is, zou tenminste een waarschuwing kunnen worden getoond aan de gebruiker en een regel worden weggeschreven naar het auditlog.

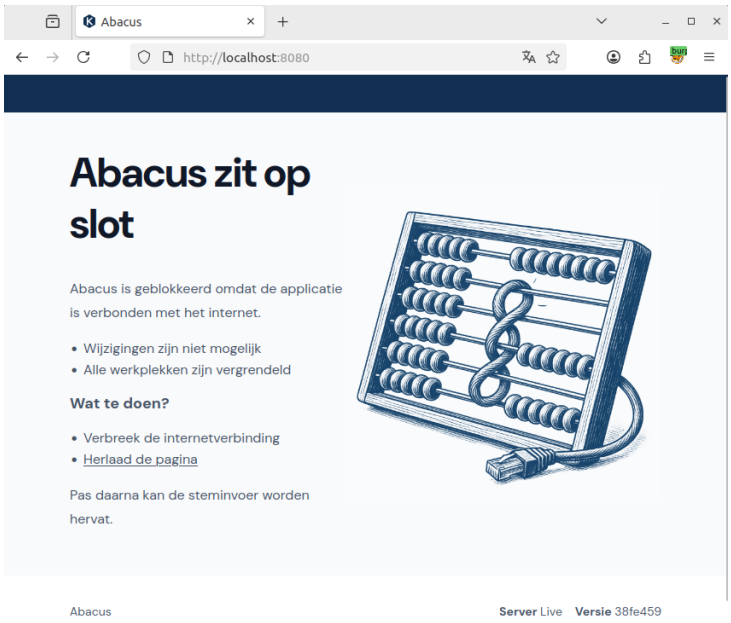
4.7.2. Identiteitsgebonden code-signing van binary builds

De binary’s van Windows en Linux zijn nog niet cryptografisch ondertekend, en de binary voor macOS bevat alleen een ‘adhoc’-ondertekening die niet is gebonden aan een identiteit van de ontwikkelaar. Het is goed gebruik om binary builds te ondertekenen met een code-signing certificaat dat is gebonden aan de identiteit van de uitgever van de software.

AANDACHTSPUNT 2	10656656.01-R2
Onderwerp	Pas identiteitsgebonden code-signing toe op binary’s
Van toepassing op	Abacus (binary builds)
Beschrijving	De binary builds van de ontwikkelversie van Abacus voor Windows, Linux en macOS zijn niet cryptografisch ondertekend met een identiteitsgebonden certificaat. Voor de releaseversie van Abacus wordt dit aanbevolen, zodat de gebruiker meer zekerheid heeft dat de authentieke software wordt geïnstalleerd. Ook kan dit installatieproblemen of beveiligingswaarschuwingen voorkomen als de beheerder er voor kiest om hardeningsmaatregelen op het gebied van code signing te implementeren.
Aanbeveling	Overweeg code-signing toe te passen op de binary builds van releaseversies van Abacus.

4.7.3. Airgapdetectie in frontend

In de ontwikkelversie kan de blokkerende controle van airgapdetectie handmatig worden ingeschakeld gedaan door Abacus te starten met de parameter `--airgap-detection` of `-a`. Zonder deze parameter alleen een waarschuwing naar het auditlog geschreven. In de release-versie van Abacus zal deze controle standaard in blokkerende modus zijn. Indien een schending van de airgap wordt gedetecteerd, zet Abacus zichzelf op slot en zien gebruikers van de webinterface het scherm dat is weergegeven in Figuur 4.3 op de volgende pagina



Figuur 4.3: Veiligheidsklep: Abacus zet zichzelf op slot na airgapdetectie

De airgapdetectie-controle wordt momenteel alleen uitgevoerd in de backend. In de frontend vindt geen airgapdetectie plaats. Dat is wel wenselijk, en blijkt[6] door de Kiesraad al te worden onderkend als ‘should have’ voor versie 1.0:

```
Versie 1.0: zeer gewenst (should have)
[...]
De Abacus-clients kunnen de afwezigheid van een internetverbinding detecteren (airgapdetectie)
.
```

De onderzoekers onderschrijven dat deze functie tenminste als ‘SHOULD HAVE’-eis moet worden beschouwd.

AANDACHTSPUNT 3	10656656.01-R3
Onderwerp	Voeg airgapdetectie toe aan de frontend
Van toepassing op	Abacus (frontend)
Beschrijving	De onderzochte ontwikkelversie van Abacus bevat nog geen airgapdetectie in de frontend. Wanneer de server correct van het internet is afgesloten, maar één van de clientsystemen dat niet is, worden gebruikers hier niet op geattendeerd. Dat is wel wenselijk, zoals door de Kiesraad ook al is onderkend.
Aanbeveling	Voeg airgapdetectie toe aan de frontend conform de ‘SHOULD HAVE’-eis.

De airgapdetectie in de backend is gedefinieerd in `./backend/src/airgap/detect.rs`. De detectie bestaat uit meerdere controles, en wordt elke 60 seconden herhaald. De controles betreffen het uitvoeren van een DNS-lookup

[6] Bron (bezocht 3 juli 2025): `https://github.com/kiesraad/abacus/blob/main/documentatie/functionaliteit/versie-1.0-gr.md`

van een internetdomein en het opzetten van uitgaande TCP-verbindingen naar poort 443 op twee hardgecodeerde IPv4-adressen en twee hardgecodeerde IPv6-adressen. De volgende snippet uit `./backend/src/airgap/detect.rs` illustreert deze configuratie:

```
const SECURE_PORT: u16 = 443; // Default secure port for HTTPS
[...]
const IPV4: [Ipv4Addr; 2] = [
    Ipv4Addr::new(104, 26, 1, 225), // Cloudflare (informatiebeveiligingsdienst.nl)
    Ipv4Addr::new(145, 100, 190, 243), // SURFnet
];

const IPV6: [Ipv6Addr; 2] = [
    Ipv6Addr::new(0x2606, 0x4700, 0x20, 0x0, 0x0, 0x0, 0x681a, 0xe1), // Cloudflare (
informatiebeveiligingsdienst.nl)
    Ipv6Addr::new(0x2001, 0x610, 0x188, 0x410, 0x145, 0x100, 0x190, 0x243), // SURFnet
];

const DOMAINS: [&str; 3] = [
    "kiesraad.nl",
    "informatiebeveiligingsdienst.nl",
    "surfnet.nl",
];

pub const AIRGAP_DETECTION_INTERVAL: u64 = 60; // interval in seconds
```

Dit is een prima werkwijze. Er zou nog kunnen worden overwogen om ook een ICMP-test toe te voegen in de backend. Net als DNS en TCP is immers ook ICMP een alomtegenwoordig protocol dat geschikt is voor Command & Control (C2)-verkeer, en het is denkbaar dat een apparaat dat geen DNS-lookups en geen uitgaande TCP-verbindingen kan doen, wel ICMP-pakketten naar internet kan sturen. Omdat een ICMP-test ofwel toegang tot een ‘raw socket’ vereist, ofwel de mogelijkheid om het systeemcommando `ping` uit te voeren, is een ICMP-test niet mogelijk in de frontend: normale webbrowsers staan zulke toegang niet toe.

De meerwaarde van een ICMP-test ten opzichte van de reeds aanwezige DNS- en TCP-test is overigens naar verwachting beperkt, maar het is een overweging waard.

AANDACHTSPUNT 4	10656656.01-R4
Onderwerp	Overweeg een ICMP-test toe te voegen aan de airgapdetectie op de backend
Van toepassing op	Abacus (backend)
Beschrijving	De airgapdetectie in de backend voert een DNS- en TCP-test uit. Er zou een ICMP-test kunnen worden toegevoegd, zodat ook is uitgesloten dat internetcommunicatie mogelijk is op een server die slaagt voor de DNS-test en TLS-test maar die via ICMP alsnog met internet kan communiceren.
Aanbeveling	Overweeg een ICMP-test toe te voegen aan de airgapdetectie op de backend. Dit kan ofwel via een raw socket, ofwel door het systeemcommando `ping` te gebruiken. Een ICMP-test vormt een extra waarborg tegen schending van het airgap-voorschrift. Opgemerkt wordt echter wel dat de meerwaarde ten opzichte van de reeds aanwezige DNS- en TCP-tests naar verwachting beperkt is.

4.7.4. Omgang met (test-)secrets in broncode

De broncode en git-history zijn met onder meer `bearer`, `gitleaks`, `grep` en handmatige inspectie onderzocht op aanwezigheid van secrets, zoals tokens of wachtwoorden. Er zijn geen secrets aangetroffen die gevoelig (kunnen) zijn.

Er zijn wel test-secrets aangetroffen die niet gevoelig zijn en alleen dienen om in de ontwikkelversie van Abacus het ontwikkel- en testwerk te faciliteren. Daarbij viel dan nog wel op dat een deel van die test-secrets zich niet alleen bevindt in broncodebestanden die als testcode herkenbaar zijn (zoals in `./frontend/e2e-tests/[...]` en `./backend/tests/[...]`), maar ook verweven zijn in 'normale' broncodebestanden. In de Rust-code bevinden deze zich in functies die expliciet zijn aangemerkt met het attribuut `#[test]`, dus als unit-test die de Rust-compiler weglaat uit non-test builds.

Voorbeelden van deze (niet gevoelige) test-secrets in 'normale' broncodebestanden van de backend:

```
./backend/src/authentication/mod.rs:
  password: "TotallyValidNewP4ssW0rd".to_string(),
  temp_password: "TotallyValidP4ssW0rd".to_string(),

./backend/src/authentication/user.rs:
  let old_password = "TotallyValidP4ssW0rd";
  let new_password = "TotallyValidNewP4ssW0rd";

./backend/src/authentication/password.rs
  let unhashed = "TotallyValidP4ssW0rd";

./backend/fixtures/users.sql
-- admin: 'AdminPassword01'
-- coordinator: 'CoordinatorPassword01'
-- typist: 'TypistPassword01'
-- typist2: 'Typist2Password01'
```

Voorbeeld van (niet gevoelige) test-secrets in de frontend, dienend voor de automatische login om tijdens het testen/ontwikkelen gemakkelijk tussen rollen te kunnen schakelen:

```
./frontend/src/app/DevHomePage.tsx:
  void login("admin", "AdminPassword01");
  void login("coordinator", "CoordinatorPassword01");
  void login("typist", "TypistPassword01");
  void login("typist2", "Typist2Password01");
```

In de frontend is verder een (niet gevoelige) test-secret aanwezig in een broncodebestand dat wel direct herkenbaar als testcode, omdat het zich in een bestandspad onder `e2e-tests` bevindt:

```
./frontend/e2e-tests/fixtures.ts:
  export const FIXTURE_TYPIST_TEMP_PASSWORD: string = "temp_password_9876";
```

De aanwezigheid van deze test-secrets is in de ontwikkelfase een non-issue. Evenwel is het verstandig om zorgvuldig en bij voorkeur consistent om te gaan met test-secrets, en er zeker van te zijn dat deze afwezig zijn in zowel de broncode als binary builds van de releaseversie.

AANDACHTSPUNT 5	10656656.01-R5
Onderwerp	Zorg dat de releaseversie van Abacus geen test-secrets bevat in broncode en/of binary builds
Van toepassing op	Abacus
Beschrijving	In de broncode en binary builds van de ontwikkelversie van Abacus zijn test-secrets aanwezig. Dit is een non-issue. Echter, als (sommige) test-secrets ook aanwezig zijn in releaseversies, zou dat tot verwarring of misverstanden kunnen leiden, zoals een goed- of kwaadwillende misvatting dat sprake is van een 'backdoor'-account.
Aanbeveling	Zorg dat de releaseversie van Abacus geen test-secrets bevat in broncode en/of binary builds.

4.7.5. Reproducible builds

Een heilige graal van integriteit en transparantie van software is 'reproducible builds'[7]. Daarmee kan een derde, zoals in deze context een burger, de Abacus-software zelf uit broncode compileren tot een binary build die *exact identiek* is aan de binary build die de Kiesraad beschikbaar stelt voor een verkiezing.

Dat schept de mogelijkheid voor derden om onafhankelijk te controleren of de binary build die door de Kiesraad beschikbaar wordt gesteld voor een verkiezing, vrij is van onverklaarde afwijkingen. Daarmee kunnen 'reproducible builds' een extra waarborg vormen die het publieke vertrouwen in het verkiezingsproces ontsteunt. De 'reproducible build' zou bijvoorbeeld de vorm kunnen hebben van een publiek beschikbaar gemaakte Docker-image met getagde versies van de Abacus-broncode en getagde versies van alle componenten die deel uitmaken van het bouwproces, zoals specifieke versies van de gebruikte compiler(s).

De (on)mogelijkheid om 'reproducible builds' te realiseren is sterk afhankelijk van technische keuzes in het ontwerp en de implementatie van software. Met Rust is het thans[8] nog niet goed mogelijk om tot 'reproducible builds' te komen.

Het realiseren van 'reproducible builds' is geen noodzaak, maar zou een punt op de horizon kunnen zijn voor de langere termijn.

AANDACHTSPUNT 6	10656656.01-R6
Onderwerp	Overweeg 'reproducible builds' van Abacus als punt op de horizon voor de langere termijn
Van toepassing op	Abacus (binary builds)
Beschrijving	Momenteel biedt Abacus geen ondersteuning voor 'reproducible builds'. Met 'reproducible builds' kan de mogelijkheid worden geschept voor derden om onafhankelijk te controleren of de binary build die door de Kiesraad beschikbaar wordt gesteld tijdens een verkiezing, vrij is van onverklaarde afwijkingen. Dit kan een extra waarborg geven voor de integriteit en controleerbaarheid van het verkiezingsproces.

Wordt vervolgd op de volgende pagina…

[7] `https://reproducible-builds.org/`
[8] Zie ook: 'Tracking Issue for Reproducible Build bugs and challenges #129080', `https://github.com/rust-lang/rust/issues/129080`

AANDACHTSPUNT 6 (vervolg)	10656656.01-R6
Aanbeveling	Overweeg ‘reproducible builds’ van Abacus als punt op de horizon voor de langere termijn. Houdt daarbij ook rekening met nadelen ervan, zoals het (onbedoeld) creëren van verwachtingen dat ook van alle toekomstige versies van Kiesraad-software een ‘reproducible build’ beschikbaar wordt gesteld: een verwachting die om technische of praktische redenen misschien niet altijd waar te maken is.

4.8. Supply chain: aanvallen via dependency-toeleverketen

4.8.1. Scenario

Een effectieve methode om grootschalige schade aan te richten is het meegeleverd krijgen van een ‘backdoor’ in de uiteindelijk gedistribueerde Abacus-software. Dit kan bijvoorbeeld door ontwikkelaars van Abacus zelf aan te vallen, of de distributiemethode. Het uitvoeren van zo’n aanval zonder gedetecteerd te worden is echter zeer complex, omdat het een beperkte groep doelwitten betreft met een hoog volwassenheidsniveau en er verschillende controls omzeild moeten worden.

Het doel kan echter ook bereikt worden door een ontwikkelaar van één van de dependency’s aan te passen. Bijvoorbeeld door diens GitHub-account te hacken, of door aan te bieden het beheer van een (klein, open-source) project over te nemen. Wanneer dit account ‘owner’-rechten heeft op `crates.io` of schrijfrechten op NPM kan hiermee eigenhandig een pakket vervangen worden voor respectievelijk een backend- of frontend-dependency.

Een malafide upload naar `crates.io` of NPM is moeilijk te detecteren, omdat de wijzigingen die aan het pakket zijn gemaakt niet terugkomen in de Git-geschiedenis en GitHub-pagina van het project.

4.8.2. Ontwikkelen en verbergen van backdoor

Vanwege de airgap kan een aanvaller met een backdoor geen kanaal opzetten voor Command & Control (C2). De malafide code zou dus offline en zonder directe aansturing moeten kunnen functioneren. Omdat Abacus echter open source is en een representatieve testomgeving gemakkelijk kan worden opgezet, kan een aanvaller wel uitgebreid diens backdoor testen zodat deze met hoge waarschijnlijkheid zou werken in een offline omgeving.

Om detectie door Abacus-testers te voorkomen kan een aanvaller een ‘tijdbom’ inbouwen: bijvoorbeeld enkel de backdoor laten activeren op of na de verkiezingsdag. Zo lang de klok op het uitvoerende systeem enigszins accuraat is, is dit een betrouwbare methode voor selectieve uitvoer. Daarnaast kan de aanvaller ook een controle inbouwen dat de backdoor-code inderdaad wordt uitgevoerd als onderdeel van Abacus, en niet als deel van ongerelateerde software met dezelfde dependency.

Om de detectiekans te verkleinen kan een aanvaller de package-update proberen zo laat mogelijk te publiceren, en obfuscatietechnieken gebruiken om te pogen de aanwezigheid van de backdoor verborgen te houden in de periode tussen publicatie en de verkiezingen.

4.8.3. Mogelijke impact

Een backdoor in zowel een frontend- als een backend-dependency zou de opgeslagen tellingen kunnen manipuleren, of dergelijke manipulaties pas kunnen toepassen bij het exporten van het telbestand en/of proces

verbaal. Hiermee daadwerkelijk een verkiezingsuitslag veranderen is in theorie mogelijk, maar een aanvaller zou deze mutaties zo subtiel moeten laten zijn dat niemand gemotiveerd is deze te vergelijken met de papieren resultaten, en op een schaal toepassen die klein genoeg is dat het gemuteerde resultaat zich waarschijnlijk niet in de steekproef van de audit bevindt. Een dergelijke aanval is zeer complex en onwaarschijnlijk. De kans dat dit ongedetecteerd kan gebeuren wordt bovendien ook procedureel tegengegaan door (procedurele) controleprotocollen die worden gebruikt voor de vaststelling van verschillende verkiezingsuitslagen.

Als het doel van de aanvaller echter is om chaos te creëren en vertrouwen in het proces te ondermijnen, dan kan die ook (grote) wijzigingen aan teldata aanbrengen die waarschijnlijk wel snel gedetecteerd worden. Ook kan de aanvaller Abacus onbeschikbaar maken en daarbij eventueel een boodschap op het scherm van de invullers tonen.

4.8.4. Aanvalsoppervlak

4.8.4.1. Directe dependency's

De dependency's van de frontend zijn als volgt:

frontend/package.json:

```
[...]
  "dependency's": {
    "browserslist": "^4.25.1",
    "lightningcss": "^1.30.1",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-router": "^7.6.3",
    "vite": "^6.3.5"
  },
[...]
```

En van de backend als volgt:

backend/Cargo.toml:

```
[dependency's]
axum = { version = "0.8", features = ["macros", "tracing", "tokio"] }
axum-extra = { version = "0.10", default-features = false, features = [
    "attachment",
    "cookie",
    "typed-header",
    "query",
] }
chrono = { version = "0.4", features = ["alloc", "std", "serde"] }
clap = { version = "4.5", default-features = false, features = [
    "derive",
    "std",
    "help",
] }
hyper = { version = "1", features = ["full"] }
tokio = { version = "1", features = ["full"] }
memory-serve = { version = "1.2.1", optional = true, features = [
```

```
    "force-embed",
] }
utoipa = { version = "5.4.0", features = ["axum_extras"] }
utoipa-axum = "0.2.0"
utoipa-swagger-ui = { version = "9.0.2", features = ["axum"], optional = true }
serde = { version = "1.0", features = ["derive"] }
serde_json = "1.0"
sqlx = { version = "0.8", features = ["runtime-tokio", "sqlite", "chrono"] }
tower = "0.5"
tracing-subscriber = { version = "0.3.19", features = ["env-filter"] }
tracing = "0.1.41"
tower-http = { version = "0.6", features = ["set-header", "trace"] }
typst = { version = "0.13.1", optional = true }
typst-pdf = { version = "0.13.1", optional = true }
quick-xml = { version = "0.38.0", features = ["serialize"] }
sha2 = "0.10.9"
argon2 = "0.5.3"
password-hash = { version = "0.5.0", features = ["getrandom"] }
rand = "0.9.0"
cookie = { version = "0.18.1", features = ["percent-encode"] }
zip = { version = "4.2.0", default-features = false, features = [
    "deflate",
    "chrono",
] }
strum = { version = "0.27", features = ["derive"] }
```

Er is geïnventariseerd hoeveel en welke GitHub-accounts het privilege hebben om één van deze pakketten op `crates.io` of NPM te updaten. Hiervoor is de volgende methode gebruikt:

1. Via de API `https://crates.io/api/v1/crates/<pakket>/owner_user` worden de ‘owners’ van de Rust-pakketten opgevraagd.
2. Via `https://crates.io/api/v1/crates/<pakket>/owner_team` worden de ‘team owners’ van de pakketten opgevraagd. Dit zijn GitHub-teams waarvan elk lid (ongeacht autorisaties binnen de GitHub-organisatie) publiceerrechten heeft[9].
3. Met `npm access ls-collaborators <pakket>` worden ACL’s voor de JavaScript/Typescript-pakketten opgevraagd; vervolgens is gefilterd op degenen die toegangsniveau `"read-write"` hebben.
4. De resulterende accountnamen worden gecombineerd en duplicaten verwijderd. Via `https://api.github.com/users/<gebruiker>` wordt informatie van publieke GitHub-profielpagina’s aangevraagd.

Dit resulteert in een lijst van 105 unieke GitHub-accounts waarvan de eigenaar gerechtigd is een directe Abacus-dependency aan te passen. Deze lijst is opgenomen in bijlageparagraaf A.1 op pagina 35.

4.8.4.2. Transitieve dependency’s

Bij het uitvoeren van een bouwcommando met Cargo of NPM worden echter niet alleen directe dependency’s opgehaald. Ook dependency’s van dependency’s (en dependency’s daarvan enz.) worden geïnstalleerd. Een aanvaller kan daardoor ook een backdoor plaatsen door enkel een transitieve dependency aan te vallen, wat het aanvalsoppervlak vergroot en detectiekans verkleind.

[9] `https://doc.rust-lang.org/cargo/reference/publishing.html#cargo-owner`

Met behulp van de GitHub-functionaliteit ‘dependency graph’ en de tool `https://npmgraph.js.org`, worden 451 Rust-pakketten en 27 JavaScript/TypeScript-pakketten geïdentificeerd die ofwel direct ofwel transitief een dependency zijn. Dezelfde methode wordt gebruikt om voor de pakketten beheerders te identificeren, wat resulteerde in een lijst van 472 unieke GitHub-accounts. Deze is opgenomen in bijlageparagraaf A.2 op pagina 36.

4.8.4.3. Ontwikkel-dependency’s

Abacus definieert ook `devDependencies`: externe dependency’s die enkel tijdens het bouw- of testproces worden binnengehaald. Deze worden niet meegedistribueerd met een productie-build, maar kunnen nog steeds een doelwit voor een supply-chainaanval zijn. In dit geval kan de aanvaller niet direct de backdoor plaatsen maar wel malafide code uitvoeren in een build pipeline. Daarbinnen zou eventueel de geproduceerde binary kunnen worden aangepast zodat deze een backdoor bevat. Deze aanval is complexer, maar kan wel via een groter aantal pakketten worden uitgevoerd.

De analyse uit paragraaf 4.8.4.2 op de pagina hiervoor wordt herhaald, maar dan ook voor `devDependencies`. Dit resulteert in een lijst van 829 unieke GitHub-accounts. Zie bijlageparagraaf A.3 op pagina 39.

4.8.4.4. Conclusie

AANDACHTSPUNT 7	10656656.01-R7
Onderwerp	Supply chain: dreiging van compromittering van een dependency-ontwikkelaar
Van toepassing op	Abacus (ontwikkelproces)
Beschrijving	Er zijn 472 GitHub-accounts gemachtigd om een software-dependency van Abacus aan te passen in de package repositories `crates.io` of NPM. Een aanvaller die één van deze accounts weet te compromitteren (bijvoorbeeld middels een gelekt wachtwoord bij een account zonder 2FA, of een door een infostealer verzamelde sessiecookie) kan een pakket publiceren met een backdoor. Zie ook paragraaf 4.8.4.2 op de pagina hiervoor. De aanvaller kan een dergelijke backdoor m.b.v. een representatieve testomgeving ontwikkelen en zo programmeren om enkel binnen Abacus-software te activeren op de verkiezingsdag. Als de backdoor onopgemerkt blijft van het moment van publicatie tot deze dag, kan de aanvaller tegelijkertijd alle Abacus-instanties aanvallen. Hoewel het ongemerkt wijzigen van een verkiezingsuitslag onwaarschijnlijk is, kan een aanvaller zo wel chaos veroorzaken en twijfel zaaien over de integriteit van het proces. Zie ook paragraaf 4.8.3 op pagina 20.

Wordt vervolgd op de volgende pagina…

AANDACHTSPUNT 7 (vervolg)	10656656.01-R7
Aanbeveling	In de risico-analyse van de Kiesraad is het risico op een dependency met een ernstige kwetsbaarheid, waardoor op afstand kwaadaardige code worden toegevoegd, reeds meegenomen. Hoewel het beschreven risico niet het gevolg is van een softwarekwetsbaarheid, is het vergelijkbaar en voorgestelde mitigerende maatregelen (zoals een beperking op en periodiek reviewproces van dependency's) zijn ook voor een supplychain-aanval via een package repository relevant. Daar bovenop worden het volgende aanbevolen: • Zorg dat bij het testen van de (pre)productie-release van Abacus een systeemdatum en -tijd wordt gebruikt die overeenkomen met de periode van de beoogde verkiezing waarin Abacus gaat worden gebruikt. Als zich een 'tijdbom' in de code bevindt dan wordt uitgevoerd, zou deze al tijdens het testen kunnen worden opgemerkt. • Kijk bij een dependencyreview ook specifiek naar de publicatiegeschiedenis op `crates.io` en NPM voor verdachte wijzigingen. Let er op dat de backdoor in het beschreven aanvalspad niet op de GitHub-pagina van de software-library zichtbaar is. Geef extra aandacht aan publicaties van een gebruikersaccount dat anders is dan het account dat gebruikelijk een specifiek pakket update. • "Bevries"wanneer mogelijk de productiebuild enige tijd voor de verkiezing. De aanvaller kan dan niet last-minute het malafide pakket publiceren, waardoor de detectiekans (ook aan de kant van de legitieme eigenaren en de gehele software-gemeenschap) wordt vergroot. • Overweeg om voor een wat langere periode voor de verkiezing versienummers van dependency's vast te pinnen. Mocht Dependabot geautoriseerd zijn om zonder goedkeuring automatische updates door te voeren, pauzeer dan dit proces en verifieer tijdens deze periode updates handmatig.

4.8.5. Naslag op gebruik van dependency's met bekende kwetsbaarheden

Er is korte naslag gedaan op gebruik van (versies van) dependency's met bekende kwetsbaarheden. Voor de frontend is `npm audit` uitgevoerd: dit leverde geen enkele dependency met een bekende kwetsbaarheid op.

Voor de backend is `cargo audit` uitgevoerd: dit leverde het overzicht op dat is weergegeven in Figuur 4.4 op pagina 26.

Het overzicht bevat geen *directe* dependency's van Abacus, maar *transitieve* dependency's die worden opgehaald door twee (wel-)directe dependency's: `sqlx` en `typst`.

Er werd één dependency met een bekende kwetsbaarheid aangetroffen, afkomstig van de Abacus-dependency `sqlx`: `rsa`-versie 0.9.8 (RUSTSEC-2023-0071), 'Marvin Attack: potential key recovery through timing sidechannels', als gevolg van het niet gebruiken van een constant-time implementatie voor cryptografische (RSA-)operaties. In de onderzochte ontwikkelversie van Abacus is deze kwetsbaarheid per definitie irrelevant omdat niet wordt gebruikgemaakt van RSA-operaties.

Verder werden twee 'unmaintained' dependency's aangetroffen, afkomstig van de Abacus-dependency `typst`: `paste` (RUSTSEC-2024-0436) en `yaml-rust` (RUSTSEC-2024-0320). Deze kunnen in potentie interessant zijn voor aanvallers die de supplychain van Abacus willen manipuleren, zoals nader beschreven in paragraaf 4.8 op pagina 20. Dat een crate de status 'unmaintained' heeft, resulteert overigens niet zomaar in een risico. Voor de crate `paste` geldt bijvoorbeeld dat deze is ontwikkeld door David Tolnay, een bekende Rust-contributor die meerdere

crates beheert: de advisory RUSTSEC-2024-0436 vloeit voort uit het feit dat de crate is gearchiveerd nadat de ontwikkelaar besloot dat de crate ‘af’ is. In dit specifieke geval ligt het niet voor de hand dat een aanvaller zich op deze crate zou richten.

```
bash-3.2$ cargo audit
    Fetching advisory database from `https://github.com/RustSec/advisory-db.git`
      Loaded 787 security advisories (from /Users/m/.cargo/advisory-db)
    Updating crates.io index
    Scanning Cargo.lock for vulnerabilities (489 crate dependencies)
Crate:     rsa
Version:   0.9.8
Title:     Marvin Attack: potential key recovery through timing sidechannels
Date:      2023-11-22
ID:        RUSTSEC-2023-0071
URL:       https://rustsec.org/advisories/RUSTSEC-2023-0071
Severity:  5.9 (medium)
Solution:  No fixed upgrade is available!
Dependency tree:
rsa 0.9.8
└── sqlx-mysql 0.8.6
    ├── sqlx-macros-core 0.8.6
    │   └── sqlx-macros 0.8.6
    │       └── sqlx 0.8.6
    │           └── abacus 0.1.0
    └── sqlx 0.8.6

Crate:     paste
Version:   1.0.15
Warning:   unmaintained
Title:     paste - no longer maintained
Date:      2024-10-07
ID:        RUSTSEC-2024-0436
URL:       https://rustsec.org/advisories/RUSTSEC-2024-0436
Dependency tree:
paste 1.0.15
├── utoipa-axum 0.2.0
│   └── abacus 0.1.0
├── hayagriva 0.8.1
│   └── typst-library 0.13.1
│       ├── typst-svg 0.13.1
│       │   └── typst-html 0.13.1
│       │       └── typst 0.13.1
│       │           └── abacus 0.1.0
│       ├── typst-realize 0.13.1
│       │   └── typst 0.13.1
│       ├── typst-pdf 0.13.1
│       │   └── abacus 0.1.0
│       ├── typst-layout 0.13.1
│       │   └── typst 0.13.1
│       ├── typst-html 0.13.1
│       ├── typst-eval 0.13.1
│       │   └── typst 0.13.1
│       └── typst 0.13.1
└── biblatex 0.10.0
    └── hayagriva 0.8.1

Crate:     yaml-rust
Version:   0.4.5
Warning:   unmaintained
Title:     yaml-rust is unmaintained.
Date:      2024-03-20
ID:        RUSTSEC-2024-0320
URL:       https://rustsec.org/advisories/RUSTSEC-2024-0320
Dependency tree:
yaml-rust 0.4.5
└── syntect 5.2.0
    ├── typst-library 0.13.1
    │   ├── typst-svg 0.13.1
    │   │   └── typst-html 0.13.1
    │   │       └── typst 0.13.1
    │   │           └── abacus 0.1.0
    │   ├── typst-realize 0.13.1
    │   │   └── typst 0.13.1
    │   ├── typst-pdf 0.13.1
    │   │   └── abacus 0.1.0
    │   ├── typst-layout 0.13.1
    │   │   └── typst 0.13.1
    │   ├── typst-html 0.13.1
    │   ├── typst-eval 0.13.1
    │   │   └── typst 0.13.1
    │   └── typst 0.13.1
    └── two-face 0.4.3
        └── typst-library 0.13.1

error: 1 vulnerability found!
warning: 2 allowed warnings found
```

Figuur 4.4: Uitvoer van `cargo audit` op backend-code

AANDACHTSPUNT 8	10656656.01-R8
Onderwerp	Twee transitieve dependency's van backend hebben status 'unmaintained'
Van toepassing op	Abacus (backend)
Beschrijving	De backend heeft `typst` als directe dependency, die twee transitieve dependency's heeft die de status 'unmaintained' hebben: `paste` en `yaml-rust`. Deze kunnen in potentie interessant zijn voor aanvallers die de supplychain van Abacus willen manipuleren, zoals nader beschreven in paragraaf 4.8 op pagina 20. Dat een crate de status 'unmaintained' heeft, resulteert overigens niet zomaar in een risico. Voor de crate `paste` geldt bijvoorbeeld dat deze is ontwikkeld door David Tolnay, een bekende Rust-contributor die meerdere crates beheert: de advisory RUSTSEC-2024-0436 vloeit voort uit het feit dat de crate is gearchiveerd nadat de ontwikkelaar besloot dat de crate 'af' is. In dit specifieke geval ligt het niet voor de hand dat een aanvaller zich op deze crate zou richten.
Aanbeveling	Wees waakzaam op de integriteit van de transitieve dependency's `paste` en `yaml-rust`, ook als deze (thans) geen deel uitmaken maken van de binary builds van Abacus.

4.9. Afzenderauthenticatie: authenticiteit uitslagbestanden

De Kiesraad heeft de wens om op termijn uitslagen uit Abacus te voorzien van cryptografische digitale handtekeningen, om de authenticiteit ervan aan te tonen. Er is echter nog geen concreet mechanisme gespecificeerd hoe deze handtekeningen worden gezet of geverifieerd. In deze paragraaf zijn enkele suggesties opgenomen voor een oplossingsrichting.

AANDACHTSPUNT 9	10656656.01-R9
Onderwerp	Suggesties voor ondertekenmechanisme uitslagbestanden
Van toepassing op	Abacus (ontwerp)
Beschrijving	De Kiesraad heeft de wens om op termijn uitslagen uit Abacus te voorzien van cryptografische digitale handtekeningen, om de authenticiteit ervan aan te tonen en aanvallen op het transport tussen stembureaus te mitigeren. Er is echter nog geen concreet mechanisme gespecificeerd hoe deze handtekeningen worden gezet of geverifieerd. In paragraaf 4.9 zijn enkele suggesties opgenomen voor oplossingsrichtingen voor de verschillende uitdagingen van een dergelijk systeem.
Aanbeveling	Neem de suggesties in paragraaf 4.9 door en evalueer of (onderdelen hiervan) bruikbaar zijn voor de uiteindelijke oplossing.

4.9.1. Dreigingsmodel

Opgemerkt wordt dat een digitale handtekening die wordt geproduceerd door de Abacus-software geen effectief middel is om aanvallen te voorkomen die het gevolg zijn van compromittering van de software of de systemen waarop deze wordt uitgevoerd. In deze situatie kan een aanvaller namelijk de input van de handtekening-functie vervangen door een vervalste waarde, en deze van een vertrouwde handtekening laten zien.

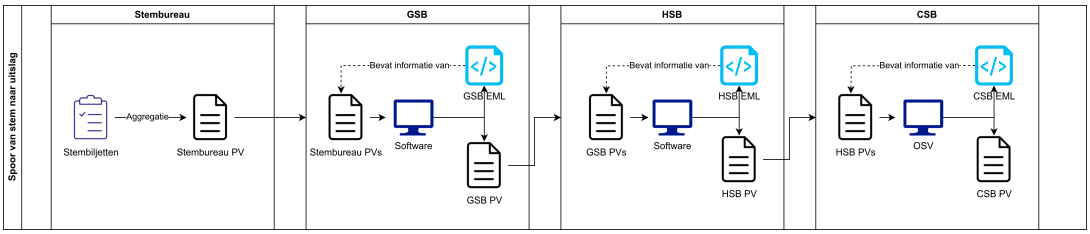
Na het produceren van een geldig EML-bestand (telbestand) en PDF-bestand (proces verbaal), kunnen deze echter wel tijdens het transport (via papier of USB-stick) worden aangepast of vervangen. In de praktijk worden maatregelen genomen om dit transport te beveiligen, en wanneer deze als afdoende worden beschouwd heeft een

ondertekeningsmechanisme in de praktijk weinig meerwaarde. Het hanteren van een dreigingsmodel waarin het transport niet automatisch als volledig vertrouwd wordt beschouwd is desalniettemin nuttig om het vertrouwen in het proces te versterken, en te anticiperen op aanvallen op (of fouten in) dit transportmechanisme.

In onderstaande suggesties is om deze reden uitgegaan van het dreigingsmodel waarbij een aanvaller bestanden kan manipuleren die getransporteerd worden van een GSB naar een HSB, of van een HSB naar het CSB. Deze aanvaller kan echter niet de Abacus-software of systemen binnen de stembureaus manipuleren.

4.9.2. Verificatie proces-verbaal

In het huidige proces zijn het de processen-verbaal (geëxporteerd als PDF door Abacus en vervolgens uitgeprint) die worden gebruikt om resultaten van GBS naar HSB of van HSB naar CSB te communiceren (zie ook figuur 4.5). Dat betekent dat de authenticiteit van deze documenten ook geverifieerd zou moeten worden. Een digitale handtekening op een PDF-bestand kan echter niet worden gevalideerd als het is uitgeprint.



Figuur 4.5: Flow van stemresultaten bij landelijke verkiezingen. Bron: Kiesraad

Het proces-verbaal bevat reeds een hash van het telbestand. Echter heeft een aanvaller in het in paragraaf 4.9.1 op de pagina hiervoor beschreven dreigingsmodel de mogelijkheid om tijdens het transport de inhoud van het proces-verbaal aan te passen. In die situatie kan die aanvaller ook de hash aanpassen; of de hash intact laten maar de overige inhoud van de PDF manipuleren.

Om deze reden wordt geadviseerd om de digitale handtekening enkel op het telbestand te plaatsen, en na verificatie daarvan de optellingen die uit het proces-verbaal zijn overgetypt te vergelijken met het telbestand. Als deze gegevens overeenkomen impliceert een geldige handtekening op het telbestand ook dat het proces-verbaal (of in ieder geval de belangrijkste gegevens daarin) authentiek is.

Dit proces is vergelijkbaar met het controleprotocol optellingen, maar wordt geautomatiseerd door uitgevoerd met enkel als doel om handtekening-verificatie mogelijk te maken. Dit voorgestelde mechanisme geen vervanging voor het controleprotocol, want dat dient proces dient onafhankelijk te zijn van de software.

Deze oplossing zou vereisen dat de USB-stick met het telbestand mee wordt getransporteerd met het proces-verbaal, en dat Abacus een EML-importfunctie krijgt waarin het telbestand wordt ingelezen en de digitale handtekening daarop wordt geverifieerd.

4.9.3. Ondertekenen en verifiëren telbestanden

Wat een digitaal handtekening-systeem complex maakt in het geval van Abacus is dat er een mechanisme moet zijn om public keys uit te wisselen over een airgap heen, dat de authenticiteit en het geassocieerde stembureau van een public keys (bij voorkeur door het algemene publiek) gecontroleerd moet kunnen worden en dat men moet kunnen vertrouwen dat alleen bij goedkeuring van een stembureauvoorzitter een signature wordt gezet.

Er zijn allerlei oplossingen voor deze problemen mogelijk, met verschillende trade-offs tussen veilighed en gebruiksgemak. Eén oplossingsrichting is bijvoorbeeld om losstaande software (zoals een telefoon-app) of hardware te gebruiken voor signing, maar dat introduceert veel extra administratieve complexiteit inclusief omgang met risico's op diefstal, verlies of defecten van die hardware.

Hierna volgen een aantal suggesties over mogelijke oplossingsrichtingen waarbij geen externe apparaten nodig zijn tijdens het tellen van de stemmen. Deze zijn als volgt:

4.9.3.1. Sleutelopslag en -generatie

Gezien in het in paragraaf 4.9.1 op pagina 27 beschreven dreigingsmodel de Abacus-software zelf wordt vertrouwd, is het acceptabel om de sleutelgeneratie-, sign- en verificatie-operaties ook door Abacus zelf te laten uitvoeren. De private key wordt dan tijdens de voorbereidingsfase gegenereerd op de ge-airgapte Abacus-server en zal deze niet verlaten. De airgap dient daardoor meteen als een belangrijke beveiligingsmaatregel tegen sleutellekken.

Een inherent risico is wel dat een Abacus-beheerder zelf de private key kan extraheren. Eventueel kan de Trusted Platform Module (TPM) van het systeem worden gebruikt om dat te voorkomen, maar dat heeft beperkte toegevoegde waarde omdat de beheerder dan alsnog onbeperkt handtekeningen om vervalste EML-bestanden kan laten zetten. Dit risico kan echter als acceptabel worden beschouwd, ook als geen TPM wordt gebruikt: zelfs als een malafide Abacus-beheerder niet bij de sleutels kan komen dan zou die immers alsnog de software kunnen manipuleren om EML-bestanden te vervalsen voordat deze ondertekent worden.

Het is wel nuttig om de vereisen dat de private key vernietigd wordt na afloop van het stemproces, zodat deze daarna niet meer misbruikt kunnen worden om achteraf handtekeningen te vervalsen.

Bij deze aanpak is een sleutelpaar geassocieerd met een Abacus-instantie, en bij het opnieuw installeren van de software zal ook dit sleutelpaar vervangen worden (zie ook paragraaf 4.9.3.6 op de pagina hierna). Om een Abacus-sleutelpaar vervolgens met een stembureau te associëren is het in paragraaf 4.9.3.3 beschreven mechanisme nodig.

4.9.3.2. Extractie public key

Als Abacus het sleutelpaar genereert, dan moet de public key wel eenmalig door de beheerder geëxtraheerd worden over de air gap heen. Eén optie is het gebruik van een USB-stick, maar omdat een EdDSA/ECDSA public key slechts 32 bytes lang is (en bij een ander algoritme kan ook een hash van deze lengte over de public key worden berekend), zijn er ook verschillende andere opties: zoals een QR-code die gescand kan worden en/of een redelijk gemakkelijk over te schrijven codering (zoals base32 met wat foutdetectie-bits, of conversie naar een woordenreeks).

4.9.3.3. Distributie en verificatie public keys

Als er per verkiezing slechts één public key geassocieerd is met elk stembureau (GSB/HSB/CSB), dan is het niet nodig om hier een complexe PKI-infrastructuur voor op te zetten. De Kiesraad kan immers ook simpelweg een enkel bestand publiceren met daarop een lijst van alle stembureaus en de bijbehorende public keys. Wanneer deze lijst over HTTPS gedownload wordt, wordt deze impliciet geauthenticeerd via de WebPKI. Daarnaast kan per verkiezing ook een hash van deze lijst worden aangekondigd via verschillende kanalen, zodat de integriteit ook zo gecontroleerd kan worden.

De Kiesraad is bij dit systeem verantwoordelijk voor het authenticeren van public keys. Voor transport van een stembureau naar de Kiesraad zou ook Diginetwerk gebruikt kunnen worden, maar omdat de sleutels slechts 32 bytes lang kunnen ook alternatieve (‘analoge’) kanalen worden gebruikt

Om te controleren of dit proces correct is verlopen, kan in de procedure worden opgenomen dat elk stembureau controleert of de public key op de publieke lijst overeen komt met de public key die de Abacus-software toont.

Eventueel kunnen gemeenten ook zelf een kopie van hun public key publiceren op hun eigen website, zodat derden deze verificatie ook kunnen uitvoeren.

Een nadeel van deze aanpak is dat public keys niet snel kunnen worden vervangen of ingetrokken. Wanneer een stembureau tijdig aangeeft dat ze bijvoorbeeld de Abacus-server opnieuw moesten vervangen, dan zou mogelijk nog een revisie gepubliceerd kunnen worden, maar dat kan niet wanneer dit op het laatste moment gebeurt. Dit zou afgevangen kunnen worden door deze situatie hetzelfde te behandelen als andere gevallen waarin signature-verificatie mislukt, en voor die uitslagen een extra controleprotocol op te stellen. Zie ook paragraaf 4.9.3.6.

4.9.3.4. Signing

Als de Abacus-software de private key opslaat, kan de signing uitgevoerd worden door de software zelf. Dit kan automatisch gebeuren bij het exporten van het EML-bestand, maar indien wenselijk kan hiervoor ook een stap aan de UI worden toegevoegd waarbij de inhoud nogmaals kan worden geïnspecteerd en op een ‘onderteken-knop’ kan worden geklikt. Elke signature die wordt gezet zou opgenomen moeten worden in het auditlog van Abacus.

De signature zou over alle bytes van het EML-bestand berekend moeten worden, en zou kunnen worden opgenomen in een apart bestand, of aan het EML-bestand worden toegevoegd. Bureau Veritas Cybersecurity raadt af om gebruik te maken van de ‘XML Signature’-standaard, gezien tekortkomingen in het ontwerp van deze standaard regelmatig tot kwetsbaarheden leiden[10][11][12].

4.9.3.5. Verificatie

Bij het signen kan Abacus een kopie van de public key meeleveren in of met het EML-bestand. Vervolgens kan aan de gebruiker gevraagd worden om deze public key te verifiëren. Wanneer dat slaagt, kan de software deze public key gebruiken om de signature op het bestand te valideren.

Voor verificatie van de public key kan dezelfde methode worden gebruikt die reeds wordt gebruikt voor verificatie van verkiezingsdefinities en kandidatenlijsten. Hierbij wordt de public key (of een hash daarvan) gedeeltelijk getoond en moet de gebruiker een aantal tekens hiervan correct overtypen. Om de juiste public key op te zoeken, zou bijvoorbeeld een uitgeprinte versie van de door de Kiesraad gepubliceerde lijst gebruikt kunnen worden.

4.9.3.6. Revocatie, verloren sleutels en verificatiefouten

Het is mogelijk dat er onbedoeld een onschadelijke wijziging aan een EML-bestand wordt gemaakt, zoals een verschil in whitespace of character encoding, waardoor verificatie mislukt. Ook is het mogelijk dat een sleutel op een laat moment moest worden ingetrokken, of dat de software na publicatie van de public key-lijst is gereset waardoor de sleutel niet meer klopt.

[10] https://secops.group/xml-signature-wrapping/
[11] https://securityboulevard.com/2025/04/security-vulnerabilities-in-saml-oauth-2-0-openid-connect-and-jwt/
[12] https://joonas.fi/2021/08/saml-is-insecure-by-design/

Daarom wordt aanbevolen om Abacus het te laten toestaan om de signature-verificatie over te slaan, maar dat daarbij wel een reden moet worden opgegeven waarom dit is gedaan. Daar kan vervolgens tegenover staan dat een extra controlestap moet worden toegepast voor uitslagen waar een geldige signature ontbreekt. Wanneer deze extra controlestap als arbeidsintensiever wordt beschouwd dan het gebruik van het signing-mechanisme, kan dat voorkomen dat signing uit gemakzucht wordt overgeslagen.

Bij een grootschalige verkiezing zullen er waarschijnlijk altijd (enkele) gemeenten zijn waar iets fout gaat in het sign-proces, ongeacht de gekozen oplossing. Achteraf kan voor deze gevallen een verslag worden gepubliceerd met de oorzaken hiervan en de extra uitgevoerde controlestappen. Eventueel kan daarbij ook een geüpdatete lijst met public keys, of een lijst met hashes van handmatig geverifieerde EML-bestanden, worden gepubliceerd.

4.10. Lokale aanvallen

In deze paragraaf worden aanvalsscenario's besproken die betrekking hebben op een aanvaller met een bepaald (beperkt) niveau van toegang tot een ge-airgapt netwerk waarin Abacus wordt gebruikt. Hieronder vallen bijvoorbeeld kwaadwillende invoerders of malware op clientsystemen waarbij om de één of andere reden de airgap is omzeild.

4.10.1. Onderscheppen onversleuteld HTTP-netwerkverkeer

Het opstarten van Abacus resulteert in een HTTP-interface die luistert op `0.0.0.0:8080`. Op apparaten die als client dienen wordt deze interface gebruikt in een normale webbrowser.

De verbinding tussen clients (browsers) en de Abacus-server maakt dus gebruik van onversleuteld HTTP. Er zijn geldige redenen waarom geen HTTPS wordt gebruikt: bij gebruik van een airgap zonder publieke domeinnamen (of publiek routeerbare IP-adressen) kunnen geen WebPKI-certificaten worden aangevraagd; daarentegen wordt installatie van een intern CA-certificaat op alle clientsystemen als onvoldoende gebruikersvriendelijk beschouwd. Het gebruik van een self-signed certificaat kan volstaan, maar biedt nauwelijks een simpelere oplossing, omdat ook dit certificaat dan alsnog handmatig in de browser moet worden geïmporteerd: het simpelweg 'wegklikken' van een certificaat-foutmelding resulteert niet in een persistente trust-relatie, de browser registreert die actie slechts in het werkgeheugen en zal na afsluiten en herstarten dezelfde certificaat-foutmelding weer tonen.

Desalniettemin introduceert het niet gebruiken van HTTPS een (beperkt) risico: een aanvaller met fysieke toegang tot een systeem, router of kabel in het netwerk kan netwerkverkeer naar Abacus onderscheppen. Hierin bevinden zich wachtwoorden en sessionidentifiers, die een aanvaller vervolgens kan gebruiken om iemand zijn account over te nemen. Een actieve aanvaller kan daarnaast ook ingevoerde tellingen muteren wanneer deze van de client naar de server worden gestuurd, of gedownloade bestanden van de server richting de client.

Een 'quick fix' die weerstand biedt tegen een passieve aanvaller, die alleen meeluistert met het netwerkverkeer en het niet (actief) manipuleert, is het toepassen van opportunistische encryptie, waarbij de server een publieke sleutel naar de client stuurt en de client deze gebruikt om het wachtwoord te versleutelen alvorens deze naar de server te sturen. Opportunistische encryptie biedt echter geen bescherming tegen een actieve aanvaller die het netwerkverkeer (wel) manipuleert en is daardoor van beperkte waarde.

AANDACHTSPUNT 10	10656656.01-R10
Onderwerp	Aanbevelingen inzake onderscheppen netwerkverkeer
Van toepassing op	Abacus (backend)

Wordt vervolgd op de volgende pagina…

AANDACHTSPUNT 10 (vervolg)	10656656.01-R10
Beschrijving	Omdat Abacus (om legitieme redenen) geen HTTPS gebruikt, kan een aanvaller met fysieke toegang tot netwerkkabels of -apparatuur, of met hoge rechten op een aangesloten apparaat, onversleuteld netwerkverkeer onderscheppen. Hiermee kunnen vervolgens andermans accounts worden overgenomen, of gegevens onderweg worden gemanipuleerd.
Aanbeveling	Als het installeren van certificaten niet haalbaar is, dan kunnen ook algemene netwerkbeveiligingsmaatregelen worden genomen om te voorkomen dat een aanvaller in de positie komt verkeer te onderscheppen. In de aansluitvoorwaarden zijn bijvoorbeeld al restricties opgenomen op het uitdelen van root- en administrator-accounts. De onderzoekers raden daarnaast ook aan om specifieke adviezen op te nemen over de bescherming van netwerkapparatuur: • Switches of routers dienen niet zonder toezicht toegankelijk te zijn voor niet-beheerders. • Schakel waar mogelijk MitM-beschermingsmaatregelen (zoals ARP-spoofpreventie) in op switches, als deze door de switch worden ondersteund. ARP-inspectie is tegenwoordig ook vaak beschikbaar op goedkopere (consumenten-)switches, maar dat is niet altijd het geval. • Schakel IPv6 uit op clientsystemen, gezien dit extra interceptie-kwetsbaarheden biedt. Hoewel zonder HTTPS geen volledige bescherming tegen MitM-aanvallers mogelijk is, is het wel mogelijk om softwarematige beschermingen te implementeren tegen passieve 'sniffers' die netwerkverkeer kunnen inzien maar niet kunnen modificeren. Namelijk: • Wachtwoorden op de frontend versleutelen met een public key die de server aanlevert ('opportunistische' encryptie). • Sessie-identifiers niet herbruikbaar maken; bijvoorbeeld door deze te binden aan een IP-adres of constant te rouleren. Wachtwoordencryptie kan ook voorkomen dat een beheerder die om legitieme redenen netwerkinspectie doet onbedoeld een gebruikerswachtwoord te zien krijgt.

4.10.2. Voorspelbare wachtwoorden

In het conceptdocument 'Functionaliteit voor Abacus 1.0 - Gemeenteraadsverkiezingen (concept)' staat over authenticatie beschreven:

```
Gebruikers kunnen lokaal worden aangemaakt met eenvoudige authenticatie, dus alleen een
gebruikersnaam en wachtwoord.
```

Gezien de praktijkomstandigheden waarin Abacus wordt gebruikt is dit niet onredelijk. In het document `wachtwoord-werkwijze.md` formuleert de Kiesraad de volgende afwegingen:

- *Vanwege het air-gapped netwerk is gebruik van Single Sign On of oplossingen die leunen op e-mail of SMS niet mogelijk.*
- *Gebruikers kunnen niet met hun vertrouwde werk-account of andere bekende gegevens inloggen. Ook 2FA is niet mogelijk*[13]

[13] Dit geldt ook voor TOTP-gebaseerde methoden, omdat die tijdsynchronisatie vereisen en uit informatie van de Kiesraad blijkt dat niet kan worden aangenomen dat apparaten waarop Abacus bij bijvoorbeeld gemeenten wordt gebruikt een correcte datum- en tijdinstelling hebben.

- *De belangrijkste rol van wachtwoorden in deze applicatie is niet het bieden van veiligheid maar het scheiden van rollen.*
- *Vanwege de rechenstructuur zijn de risico's van misbruik van een Coördinator of Beheerder account groter dan de risico's bij misbruik van een Inoverder account*
- *Coördinatoren en Beheerders loggen tijdens de verkiezingsperiode veelvuldig in in Abacus Invoerders zullen gedurende de invoersessie een aantal keren moeten uit- en inloggen. Bijvoorbeeld als ze hun werkplek verlaten voor een pauze of sanitaire stop*
- *Invoerders die hun wachtwoord vergeten zorgen voor extra workload bij de Coördinator of Beheerder die tijdens de steminvoer al druk is met andere taken*

Daarnaast wordt onderscheid gemaakt tussen de rollen Coordinator en Beheerder enerzijds, en Invoerders anderzijds:

Coördinatoren en Beheerders

- *De Beheerder maakt voor deze gebruikers een account aan en geeft daarbij gebruikersnaam en een tijdelijk wachtwoord op.*
- *Gebruikers ontvangen gebruikersnaam en tijdelijk wachtwoord van een Beheerder. De Beheerder kiest hierbij zelf het distributiekanaal. Gebruikers moeten bij eerste gebruik het wachtwoord aanpassen.*
- *Vanwege het frequentere gebruik over een langere periode en de grotere risico's bij misbruik, hebben we voor deze gebruikers een streng wachtwoordbeleid. Wachtwoorden moeten minimaal 13 karakters lang zijn.*
- *Na 3 mislukte inlogpogingen blokkeren we het account (alternatief: exponentieel oplopende tijd tussen loginpogingen)*

Invoerders

...

- *Vanwege de gebruikscontext en het tijdelijke karakter van het account kiezen we voor minder strenge wachtwoordeisen. Het wachtwoord moet minimaal 8 karakters bevatten. Cijfers of speciale symbolen zijn niet vereist.*

...

- *Na 3 mislukte inlogpogingen blokkeren we het account. De Coördinator of Beheerder kan vervolgens een nieuw tijdelijk wachtwoord instellen, dat de gebruiker na opnieuw inloggen weer moet aanpassen.*

Bureau Veritas Cybersecurity beschouwd dit als redelijke wachtwoordeisen in de context van het systeem. Het blokkeren van een account na drie pogingen (of exponentieel oplopende tijd in het geval van beheerders- en coördinatoraccounts) is een effectieve methode om het (geautomatiseerd) raden van wachtwoorden te voorkomen.

Als toevoeging hierop raadt Bureau Veritas Cybersecurity ook aan om een 'blocklist' van zeer vaak voorkomende wachtwoorden of onderdelen van wachtwoorden te hanteren. In de ervaring van de onderzoekers komen bijvoorbeeld cijferreeksen (bijvoorbeeld `12345678`), toetsenbordpatronen (`1qazxsw2`) en (domein-)specifieke basiswoorden (`welkom01`, `verkiezingen2026`) veel voor. Uit een onderzoek van SplashData uit 2016[14] bleek 10% van de onderzochte accounts één van de 25 meest voorkomende wachtwoorden te hebben.

Zelfs bij accountblokkade na drie foute pogingen zou een aanvaller alsnog een *password spray* kunnen proberen waarbij één of twee zeer populaire wachtwoord voor verschillende accounts worden geprobeerd.

[14] `https://time.com/4639791/worst-passwords-2016/`

In de versie van Abacus (0.1.0-alpha.20250706.38fe459) die beschikbaar was tijdens het onderzoek, waren de in het document `wachtwoord-werkwijze.md` genoemde maatregelen nog niet volledig geïmplementeerd. De wachtwoordrestricties in deze versie waren geïmplementeerd in `./backend/src/authentication/password.rs`, en als volgt:

- Wachtwoorden dienen tenminste 13 tekens lang te zijn.
- Het wachtwoord mag niet gelijk zijn aan de gebruikersnaam.
- Het wachtwoord mag niet gelijk zijn aan het vorige (potentieel op papier aangeleverde) wachtwoord.

Wachtwoorden van 8 karakters waren in deze versie nog niet toegestaan voor invoerders. Ook werd er nog geen limiet afgedwongen op een maximaal aantal mislukte inlogpogingen. Dat laatste beschouwd Bureau Veritas Cybersecurity als een belangrijke maatregel: hoewel login-requests enigszins worden vertraagd omdat de backend een dure wachtwoordhashfunctie (Argon2) moet uitvoeren, zou een ‘online’ brute-forceaanval met zeer veelgebruikte wachtwoorden nog steeds een goede slagingskans kunnen hebben.

Bureau Veritas Cybersecurity merkt daarnaast op mislukte inlogpogingen niet in de activiteitenlog werden getoond, waardoor een brute-forceaanval onder de radar kan blijven.

AANDACHTSPUNT 11	10656656.01-R11
Onderwerp	Adviezen met betrekking tot wachtwoordbeleid
Van toepassing op	Abacus (backend)
Beschrijving	Bureau Veritas Cybersecurity is het eens met de afwegingen die worden gemaakt in het document `wachtwoord-werkwijze.md`, met betrekking tot wachtwoordvereisten. De werkwijze in dit document was echter nog niet volledig geïmplementeerd in de versie van Abacus die tijdens het onderzoek beschikbaar was. Ook werd opgemerkt dat mislukte inlogpogingen niet in de log van Abacus werden opgenomen. Hierdoor zouden malafide pogingen om andermans wachtwoord te raden onopgemerkt kunnen blijven. Als aanvullende beveiligingsmaatregel wordt daarnaast geadviseerd om ook zeer populaire wachtwoorden (zoals `12345678` of `1qazxsw2`) of onderdelen van wachtwoorden (zoals `wachtwoord` of `welkom`) te blokkeren.
Aanbeveling	Implementeer de in `wachtwoord-werkwijze.md` beschreven werkwijze in Abacus. Met name de accountblokkade na enkele mislukte inlogpogingen is een belangrijke maatregel. Neem mislukte inlogpogingen daarnaast ook op in de zichtbare logs. Bureau Veritas Cybersecurity raadt ook aan om een (bescheiden) blocklist van zeer vaak gebruikte wachtwoorden te handhaven.

A. Externe dependency's: overzicht van GitHub-accounts met schrijfrechten

Als onderdeel van de analyse van het aanvalsoppervlak in paragraaf 4.8 op pagina 20, zijn lijsten verzameld van GitHub-accounts die individueel wijzigingen in een externe dependency van Abacus kunnen updaten via de package repositories `crates.io` of NPM.

A.1. Directe dependencies

Onderstaande accounts (geïdentificeerd op basis van GitHub-gebruikersnaam) hebben schrijfrechten op directe dependencies. Zie ook paragraaf 4.8.4.1 op pagina 21.

1. `0xPoe`
2. `Aaron1011`
3. `aatxe`
4. `abonander`
5. `acfoltzer`
6. `ADD-\acs{SP}`
7. `agayev`
8. `aidanhs`
9. `alce`
10. `alecmocatta`
11. `alexcrichton`
12. `alexheretic`
13. `ashleygwilliams`
14. `AzureMarker`
15. `baloo`
16. `battesonb`
17. `benjamin-lieser`
18. `bluejekyll`
19. `bryangarza`
20. `BurntSushi`
21. `Byron`
22. `calebcartwright`
23. `carllerche`
24. `carols10cents`
25. `ChrisDenton`
26. `codesections`
27. `compiler-errors`
28. `cratelyn`
29. `csmoe`
30. `Darksonn`
31. `davidbarsky`
32. `davidpdrsn`
33. `davidtwco`
34. `d-e-s-o`
35. `detrumi`
36. `dhardy`
37. `Diggsey`
38. `dignifiedquire`
39. `djc`
40. `dralley`
41. `dswij`
42. `dtolnay`
43. `Dylan-DPC`
44. `eddyb`
45. `edunham`
46. `ELD`
47. `epage`
48. `erickt`
49. `fmease`
50. `gardnervickers`
51. `gaurikholkar-zz`
52. `gruberb`
53. `GuillaumeGomez`
54. `hawkw`
55. `hrvolapeter`
56. `huonw`
57. `ipetkov`
58. `jlizen`
59. `josephlr`
60. `jplatte`
61. `juhaku`
62. `jxs`
63. `kbknapp`
64. `laurmaedje`
65. `ldm0`
66. `lifthrasiir`
67. `LucioFranco`
68. `maminrayej`
69. `marlonbaeten`
70. `matthiasbeyer`
71. `mehcode`
72. `MichaelOwenDyer`
73. `Milo123459`
74. `Mingun`
75. `mox692`
76. `newpavlov`
77. `nickray`
78. `oli-obk`
79. `olix0r`
80. `Peternator7`
81. `pksunkara`
82. `Plecra`
83. `Pr0methean`
84. `quininer`
85. `satakuma`
86. `seanmonstar`
87. `SergioBenitez`
88. `tafia`
89. `taiki-e`
90. `tarcieri`
91. `tobz`
92. `vladikoff`
93. `vorot93`
94. `warner`
95. `ai`
96. `antfu`
97. `brophdawg11`
98. `devongovett`
99. `fb`
100. `mjackson`
101. `patak`
102. `react`
103. `timdorr`
104. `vitebot`
105. `yyx990803`

A.2. Transitieve dependencies

Onderstaande GitHub-accounts hebben schrijfrechten op directe ofwel transitieve dependencies. Zie ook paragraaf 4.8.4.2 op pagina 22.

1. 0xPoe
2. 132ikl
3. 197g
4. 1c3t3a
5. 4lDO2
6. 7rulnik
7. Aaron1011
8. aatxe
9. abonander
10. abrown
11. acfoltzer
12. adamgreig
13. adamreichold
14. ADD-\acs{SP}
15. aDotInTheVoid
16. adrq
17. agayev
18. ai
19. aidanhs
20. akhilles
21. albertlarsan68
22. alce
23. AldaronLau
24. alecmocatta
25. alerque
26. alex
27. alexcrichton
28. Alexendoo
29. AlexEne
30. alexeyraspopov
31. alexheretic
32. alex-semenyuk
33. AlexTMjugador
34. alibektas
35. alicemaz
36. aliemjay
37. allan2
38. Amanieu
39. amanjeev
40. amtoine
41. andrasio
42. andreiltd
43. andre-richter
44. andrewpollack
45. AngelOnFira
46. anp
47. antfu
48. anthrotype
49. aphillips
50. arazabishov
51. arora-aman
52. asajeffrey
53. ashleygwilliams
54. astraw
55. atbrakhi
56. aturon
57. autozimu
58. avanhatt
59. awelkie
60. awygle
61. AzureMarker
62. badboy
63. Bahex
64. baloo
65. baoyachi
66. BartMassey
67. battesonb
68. behnam
69. benbrandt
70. beneb
71. benjamin-lieser
72. berkus
73. birktj
74. blakeembrey
75. bluejekyll
76. bluss
77. bnjbvr
78. brendanzab
79. brophdawg11
80. brson
81. Bruflot
82. bryangarza
83. bt
84. btangmu
85. BurntSushi
86. burrbull
87. Byron
88. calebcartwright
89. carllerche
90. carols10cents
91. cathieyun
92. cbreeden
93. cfallin
94. chicoxyzzy
95. ChrisDenton
96. chrisduerr
97. chyh1990
98. cjchapman
99. cmyr
100. codesections
101. comex
102. compiler-errors
103. Constellation
104. Constellation
105. CosmicHorrorDev
106. cptpiepmatz
107. cramertj
108. cratelyn
109. csmoe
110. cuviper
111. CYBAI
112. danaugrs
113. danez
114. danielrh
115. danwilhelm
116. Darksonn
117. data-pup
118. davidbarsky
119. davidcarlisle
120. David-OConnor
121. davidpdrsn
122. davidtwco
123. deian
124. delan
125. d-e-s-o
126. detrumi
127. devongovett
128. de-vri-es
129. dfrankland
130. dfrg
131. dhardy
132. Diggsey

133. dignifiedquire
134. Dirbaio
135. Disasm
136. divergentdave
137. djc
138. dkhayes117
139. dminor
140. doowb
141. dougwilson
142. drager
143. dralley
144. droundy
145. dswij
146. dtolnay
147. dvc94ch
148. dvdhrm
149. Dylan-DPC
150. Dylan-DPC-zz
151. ebarnard
152. eddyb
153. edunham
154. eggrobin
155. ehuss
156. eira-fransham
157. ELD
158. eldruin
159. elliottt
160. Emilgardis
161. emilio
162. enarxbot
163. Enselic
164. eopb
165. epage
166. erickt
167. esbuild
168. etemesi254
169. evanw
170. eventualbuddha
171. fabalbon
172. faern
173. Farkal
174. fb
175. fenhl
176. fintelia
177. fitzgen
178. fizyk20
179. fmease
180. fogti
181. folkertdev
182. frewsxcv
183. Frommi
184. Gankra
185. gardnervickers
186. gaurikholkar-zz
187. glennw
188. gnzlbg
189. gregtatum
190. grhoten
191. gruberb
192. gterzian
193. GuillaumeGomez
194. gwenn
195. hannobraun
196. haraldh
197. hargoniX
198. hawkw
199. hdoordt
200. Hoverbear
201. hrvolapeter
202. hsivonen
203. huangjj27
204. huonw
205. IanManske
206. ibaryshnikov
207. ibraheemdev
208. ipetkov
209. IsaacWoods
210. isislovecruft
211. ithinuel
212. jackpot51
213. jamesmunns
214. jamiebuilds
215. jannic
216. japaric
217. jdm
218. jedisct1
219. jeehoonkang
220. jefgen
221. jessebraham
222. jhpratt
223. jlizen
224. JohnTitor
225. jonschlinkert
226. josephlr
227. joshlf
228. joshtriplett
229. jpernst
230. jplatte
231. jrmuizel
232. jschwe
233. jswrenn
234. juhaku
235. jwilm
236. jxs
237. Kagami
238. kaj
239. kaksmet
240. kamadak
241. kartva
242. kazcw
243. kbknapp
244. kchibisov
245. kenlunde
246. kennykerr
247. khaledhosny
248. Kijewski
249. killercup
250. Kimundi
251. kinnison
252. KodrAus
253. KokaKiwi
254. kornelski
255. krisprice
256. kubkon
257. kvark
258. kwantam
259. kyren
260. lambda-fairy
261. larsbergstrom
262. LaurenzV
263. laurmaedje
264. ldm0
265. lianghai
266. lifthrasiir
267. lo48576
268. Lokathor
269. loukamb
270. lovell
271. LucioFranco
272. lukastaegert
273. lukeed
274. lunacookies
275. lxfontes
276. lydell
277. MabezDev
278. macchiati
279. magiclen

280. mainrs
281. MajorBreakfast
282. makotokato
283. maminrayej
284. Manishearth
285. marcianx
286. markusicu
287. marlonbaeten
288. marshallpierce
289. martinlindhe
290. mathiasbynens
291. matklad
292. Matthias247
293. matthiasbeyer
294. mbrubeck
295. mcgoo
296. mcountryman
297. mehcode
298. metajack
299. mgeisler
300. MichaelOwenDyer
301. michelleperham
302. MikeCamel
303. mikedilger
304. Milo123459
305. Mingun
306. mjackson
307. Mossaka
308. mox692
309. mrego
310. mrhooray
311. mrmlnc
312. mrobinson
313. Ms2ger
314. mstallmo
315. mukilan
316. mvdnes
317. mystor
318. nabijaczleweli
319. nagisa
320. nastevens
321. newAM
322. newpavlov
323. nfriedly
324. nical
325. nickray
326. nickvidal
327. nicoburns
328. nikomatsakis
329. nordzilla
330. notgull
331. novacrazy
332. nox
333. npmccallum
334. Ogeon
335. ogham
336. ohanar
337. oli-obk
338. olix0r
339. ordian
340. orlp
341. ospencer
342. oyvindln
343. pacman82
344. paholg
345. patak
346. PaulGrandperrin
347. paupino
348. pczarn
349. pepsighan
350. pepyakin
351. peterhuene
352. Peternator7
353. philipc
354. pkgw
355. pksunkara
356. Plecra
357. posborne
358. Pr0methean
359. pyfisch
360. quininer
361. radu-matei
362. raphlinus
363. RazrFalcon
364. react-bot
365. reem
366. reitermarkus
367. reknih
368. retep998
369. richard-uk1
370. Riey
371. ripytide
372. rjzak
373. rkuhn
374. rnijveld
375. robamu
376. Robbepop
377. robertbastian
378. robinst
379. Robzz
380. roubert
381. Roughsketch
382. RReverser
383. RReverser
384. RumovZ
385. rust-lang-owner
386. rvolosatovs
387. rzumer
388. sagudev
389. samchen61661
390. satakuma
391. saulecabrera
392. schets
393. sdroege
394. seanmonstar
395. sebasmagri
396. sebcrozet
397. sendilkumarn
398. SergioBenitez
399. sfackler
400. sffc
401. Shnatsel
402. sholderbach
403. silesmo
404. SimonSapin
405. sirhcel
406. skius
407. smoelius
408. snktd
409. sophiajt
410. Soveu
411. spinda
412. srijs
413. srl295
414. starkat99
415. Storyyeller
416. sujayakar
417. sunfishcode
418. tacdom
419. tafia
420. taiki-e
421. tailhook
422. tarcieri
423. Taym95
424. tbu-
425. TH3CHARLie
426. thalesfragoso

427. thecodrr
428. theotherphil
429. Thomasdezeeuw
430. thomcc
431. TianyiShi2001
432. ticki
433. timdorr
434. tobz
435. torch2424
436. trishume
437. tschneidereit
438. tspiteri
439. tyler
440. TyOverby
441. types
442. udoprog
443. utkarshgupta137
444. valenting
445. Veykril
446. VictorKoenders
447. vitebot
448. vladikoff
449. vorner
450. vorot93
451. Vtec234
452. v-thakkar
453. warner
454. waych
455. waywardmonkeys
456. whitequark
457. withoutboats
458. wusyong
459. xorgy
460. xStrom
461. xtuc
462. yaahc
463. yoshuawuyts
464. ysthakur
465. yurydelendik
466. yyx990803
467. zakarumych
468. zbraniecki
469. zeenix
470. zertosh
471. zesterer
472. ZoeyR

A.3. Alle dependencies

In de onderstaande lijst van GitHub-accounts worden ook accounts meegenomen die rechten hebben op een dependency die enkel tijdens het ontwikkel- of testproces wordt gebruikt. Zie ook paragraaf 4.8.4.2 op pagina 22.

1. 0xPoe
2. 132ikl
3. 197g
4. 1c3t3a
5. 3imed-jaberi
6. 3rdeden
7. 43081j
8. 4lDO2
9. 7rulnik
10. aaron
11. Aaron1011
12. aatxe
13. abetomo
14. abonander
15. abrown
16. acfoltzer
17. adamgreig
18. adamreichold
19. ADD-\acs{SP}
20. aDotInTheVoid
21. adrianheine
22. adrq
23. agayev
24. ai
25. aidanhs
26. airhorns
27. akhilles
28. alaguna
29. alangpierce
30. albertlarsan68
31. alce
32. AldaronLau
33. aleclarson
34. alecmocatta
35. alerque
36. alex
37. alexanderGugel
38. alexcrichton
39. Alexendoo
40. AlexEne
41. alexeyraspopov
42. alexgorbatchev
43. alexheretic
44. alex-semenyuk
45. AlexTMjugador
46. alibektas
47. alicemaz
48. aliemjay
49. allan2
50. Amanieu
51. amanjeev
52. amol-anand
53. amtoine
54. Andarist
55. andrasio
56. andreiltd
57. andre-richter
58. andrewbranch
59. andrewpollack
60. AngelOnFira
61. anp
62. antfu
63. anthrotype
64. aphillips
65. arazabishov
66. AriPerkkio
67. arnaud-barre
68. arora-aman
69. asajeffrey
70. aseemk
71. ashleygwilliams
72. ashtuchkin
73. Aslemammad
74. aspro83
75. asthabh23

76. astraw
77. atbrakhi
78. aturon
79. autozimu
80. avanhatt
81. awaterma
82. awelkie
83. awygle
84. ayusharma
85. AzureMarker
86. azz
87. badboy
88. Bahex
89. baloo
90. baoyachi
91. BartMassey
92. bashmish
93. battesonb
94. bcoe
95. behnam
96. benbrandt
97. beneb
98. benjamin-lieser
99. benjie
100. benmosher
101. berkus
102. birktj
103. bjolind
104. bkonkle
105. blakeembrey
106. bluejekyll
107. bluss
108. bnjbvr
109. bnjmnt4n
110. bradzacher
111. brendanzab
112. bret
113. brizental
114. Brooooooklyn
115. brophdawg11
116. brrianalexis
117. brson
118. Bruflot
119. bryangarza
120. bryanmacfarlane
121. bt
122. btangmu
123. BurntSushi
124. burrbull
125. BYK
126. Byron
127. calebcartwright
128. carbonrobot
129. carllerche
130. carols10cents
131. cathieyun
132. cbreeden
133. ccasey
134. ceceppa
135. cfallin
136. chad3814
137. chaijs
138. ChALkeR
139. chenfengyuan
140. chicoxyzzy
141. ChrisDenton
142. chrisduerr
143. chrispat
144. chyh1990
145. cjchapman
146. climba03003
147. CMckinstry
148. cmyr
149. codecov-devops
150. codesections
151. colinhacks
152. comex
153. compiler-errors
154. Constellation
155. Constellation
156. coreyfarrell
157. CosmicHorrorDev
158. cpojer
159. cptpiepmatz
160. cramertj
161. cratelyn
162. cschleiden
163. csmoe
164. cspotcode
165. cuviper
166. CYBAI
167. danaugrs
168. danez
169. danielchatfield
170. danielrh
171. danielroe
172. danwilhelm
173. Darksonn
174. data-pup
175. davidbarsky
176. davidbonnet
177. davidcarlisle
178. davidkpiano
179. David-OConnor
180. davidpdrsn
181. davidtwco
182. dcousens
183. dcpfsdk
184. dead-horse
185. defunctzombie
186. deian
187. delan
188. delvedor
189. d-e-s-o
190. detrumi
191. devongovett
192. de-vri-es
193. dfcook
194. dfrankland
195. dfrg
196. dhardy
197. diego
198. Diggsey
199. dignifiedquire
200. Dirbaio
201. Disasm
202. divergentdave
203. dividstefansvensson
204. djc
205. dkhayes117
206. dminor
207. domenic
208. dominicbarnes
209. doowb
210. Doten
211. dougwilson
212. dpfister
213. DQLabs
214. drager
215. dralley
216. drazisil
217. droundy
218. dswij
219. dtolnay
220. duailibe
221. dvc94ch
222. dvdhrm

223. dylanb
224. dylandepass
225. Dylan-DPC
226. Dylan-DPC-zz
227. ebarnard
228. eddyb
229. EduardoRFS
230. edunham
231. eemeli
232. egeste
233. eggrobin
234. egoist
235. ehuss
236. einaros
237. eira-fransham
238. ejpbruel
239. ELD
240. eldruin
241. elliottt
242. emilbayes
243. Emilgardis
244. emilio
245. enarxbot
246. enisdenjo
247. Enselic
248. Eomm
249. eopb
250. epage
251. eps1lon
252. erickt
253. erieng
254. es128
255. esbuild
256. eslintbot
257. esp
258. etemesi254
259. evanw
260. evcohen
261. eventualbuddha
262. evilebottnawi
263. existentialism
264. fabalbon
265. faddee
266. faern
267. Farkal
268. fb
269. fengmk2
270. fenhl
271. feross
272. fintelia
273. Fishrock123
274. fisker
275. fitzgen
276. fizyk20
277. fmease
278. fmeschbe
279. fogti
280. folkertdev
281. forehalo
282. FormidableLabs
283. fox1t
284. Fraggle
285. frewsxcv
286. Frommi
287. fullcolorcoder
288. gaearon
289. galvez
290. Gankra
291. gar
292. gardnervickers
293. GarthDB
294. garycourt
295. gaurikholkar-zz
296. geon
297. gijs
298. gkz
299. glennw
300. gnzlbg
301. gotwarlost
302. Gpx
303. gr2m
304. gregtatum
305. grhoten
306. gruberb
307. gterzian
308. GuillaumeGomez
309. gurgunday
310. guybedford
311. gwenn
312. hannobraun
313. haraldh
314. hargoniX
315. hashtagchris
316. hawkw
317. hdoordt
318. henbr
319. hiddentao
320. hootener
321. Hoverbear
322. hrvolapeter
323. hsivonen
324. huangjj27
325. huonw
326. hzoo
327. i1g
328. IanManske
329. ibaryshnikov
330. ibraheemdev
331. ikatyang
332. inikulin
333. ipetkov
334. isaacs
335. IsaacWoods
336. isislovecruft
337. ithinuel
338. ivolodin
339. jackpot51
340. jakebailey
341. JaKXz
342. JamesHenry
343. jamesmunns
344. jamiebuilds
345. jannic
346. japaric
347. jaredwray
348. jbgutierrez
349. jdalton
350. jdm
351. jean-michelet
352. jed
353. jedisct1
354. JedWatson
355. jeehoonkang
356. jefgen
357. jensyt
358. jessebeach
359. jessebraham
360. jfmengels
361. jfsiii
362. jhpratt
363. jkoops
364. jkratzer
365. JLHwung
366. jlizen
367. jlongster
368. johankristiansson
369. JohannesKlauss

370. Johkah
371. johno
372. JohnTitor
373. jonaskello
374. jonathanong
375. jonathantneal
376. jonchurch
377. jongleberry
378. jonschlinkert
379. jordanbtucker
380. jorenbroekema
381. Joris-van-der-Wel
382. josephlr
383. joshlf
384. joshmgross
385. joshtriplett
386. JoshuaKGoldberg
387. JounQin
388. jpernst
389. jplatte
390. jrfeenst
391. jridgewell
392. jrmuizel
393. jschwe
394. jsumners
395. jswrenn
396. juhaku
397. juliangruber
398. justmoon
399. jwilm
400. jxs
401. kael
402. Kagami
403. kaj
404. kaksmet
405. kamadak
406. kartva
407. kazcw
408. kbknapp
409. kchan
410. kchibisov
411. kdy1
412. keithluchtel
413. kenlunde
414. kennykerr
415. kentcdodds
416. kettanaito
417. kevva
418. KhaledElAnsari
419. khaledhosny
420. kibertoad
421. Kijewski
422. killercup
423. Kimundi
424. kinnison
425. kmck
426. knownasilya
427. knowtheory
428. KodrAus
429. koichik
430. KokaKiwi
431. komagata
432. konradpabjan
433. kornelski
434. kptdobe
435. krisnye
436. krisprice
437. kubkon
438. kvark
439. kwantam
440. kwonoj
441. kyldvs
442. kylehousley
443. kyren
444. lambda-fairy
445. larsbergstrom
446. LaurenzV
447. laurmaedje
448. lazd
449. lazurski
450. lddubeau
451. ldm0
452. leebyron
453. lencioni
454. leo
455. lianghai
456. lifthrasiir
457. ljharb
458. lo1tuma
459. lo48576
460. loganfsmyth
461. Lokathor
462. loukamb
463. lourd
464. lovell
465. lpinca
466. LucioFranco
467. lukastaegert
468. lukechilds
469. lukeed
470. lunacookies
471. lupomontero
472. lxfontes
473. lydell
474. MabezDev
475. macchiati
476. mafintosh
477. magiclen
478. mainrs
479. MajorBreakfast
480. makotokato
481. maminrayej
482. Manishearth
483. marbec
484. marcianx
485. marcysutton
486. mariano-formidable
487. marijn
488. markusicu
489. marlonbaeten
490. marshallpierce
491. Marsve
492. martinlindhe
493. masiddee
494. matheuss
495. mathias
496. mathiasbynens
497. matklad
498. Matthias247
499. matthiasbeyer
500. mattpauldavies
501. mbrubeck
502. mcg
503. mcgoo
504. mcountryman
505. mdevils
506. mdjastrzebski
507. megawac
508. mehcode
509. metajack
510. metcoder95
511. mgeisler
512. mhaack
513. MichaelDeBoey
514. michaelficarra
515. michaelmerrill
516. MichaelOwenDyer

517. michelleperham
518. mihar-22
519. MikeCamel
520. mikedilger
521. MikeMcl
522. miksu
523. Milo123459
524. Mingun
525. mischah
526. mjackson
527. mjmahone
528. mlifshin
529. MoOx
530. Mossaka
531. mourner
532. mox692
533. mpeyper
534. mrego
535. mrexox
536. mrhooray
537. mrmlnc
538. mrobinson
539. Ms2ger
540. mskelton
541. mstallmo
542. mukilan
543. mvdnes
544. mxschmitt
545. mystor
546. mythmon
547. nabijaczleweli
548. nagisa
549. nastevens
550. NateBaldwin
551. ndelangen
552. newAM
553. newpavlov
554. nexdrew
555. nfriedly
556. nical
557. nicholas-codecov
558. nickfitzgerald
559. nickray
560. nickvidal
561. nicoburns
562. nicolo-ribaudo
563. niftylettuce
564. nikomatsakis
565. nopersonsmodules
566. nordzilla
567. notgull
568. novacrazy
569. nox
570. npmccallum
571. nzakas
572. octokitbot
573. Ogeon
574. ogham
575. ohanar
576. oli-obk
577. olix0r
578. openjsfoundation
579. openjs-operations
580. ordian
581. orlp
582. oskdah
583. ospencer
584. oss-bot
585. overset
586. owlstronaut
587. oyvindln
588. pacman82
589. pago
590. paholg
591. passle
592. patak
593. patrickhulce
594. PaulGrandperrin
595. paulmillr
596. paupino
597. pavelfeldman
598. pczarn
599. pepsighan
600. pepyakin
601. peterhuene
602. Peternator7
603. phated
604. philipc
605. philpl
606. phryneas
607. pi0
608. pieroxy
609. pipobscure
610. pkgw
611. pksunkara
612. Plecra
613. posborne
614. Pr0methean
615. prettier-bot
616. pyfisch
617. qix
618. quininer
619. radubrehar
620. radu-matei
621. raphlinus
622. rashal01
623. RasSva
624. rauchg
625. Raynos
626. RazrFalcon
627. react-bot
628. realityking
629. reem
630. reggi
631. reitermarkus
632. reknih
633. Rem
634. remarkablemark
635. remcohaszing
636. remusao
637. retep998
638. richard-uk1
639. rickhanlonii
640. riderjensen
641. Riey
642. ripytide
643. rjzak
644. rkuhn
645. rnijveld
646. robamu
647. Robbepop
648. robertbastian
649. robinst
650. robrez
651. robwalkerco
652. Robzz
653. rofe
654. romainmenke
655. ronag
656. roubert
657. Roughsketch
658. RReverser
659. RReverser
660. rtsao
661. RumovZ
662. rust-lang-owner
663. rvolosatovs

664. RyanZim
665. rzumer
666. sagudev
667. samchen61661
668. samn
669. SamVerschueren
670. saquibkhan
671. sarahformidable
672. sarmeyer
673. satakuma
674. satazor
675. saulecabrera
676. SBoudrias
677. schets
678. scottianstewart
679. scott-rippey
680. sdroege
681. seanmonstar
682. sebasmagri
683. sebcrozet
684. Sebmaster
685. sendilkumarn
686. SergioBenitez
687. sfackler
688. sffc
689. shadowspawn
690. shazron
691. sheetalkamat
692. shellscape
693. Shnatsel
694. sholderbach
695. silesmo
696. SimenB
697. simoneb
698. SimonSapin
699. sindresorhus
700. sirhcel
701. skius
702. smoelius
703. snktd
704. SomeKittens
705. sophiajt
706. sophiebits
707. sosukesuzuki
708. Soveu
709. spinda
710. srijs
711. srl295
712. starkat99
713. stefan-guggisberg
714. stefanpenner
715. stephenmathieson
716. Storyyeller
717. suchipi
718. sujayakar
719. sundress
720. sunfishcode
721. suryagh
722. Swaagie
723. sxzz
724. tacdom
725. tafia
726. taiki-e
727. tailhook
728. tarcieri
729. Taym95
730. tbu-
731. tejasmanohar
732. terkelg
733. testing-library-bot
734. TH3CHARLie
735. thalesfragoso
736. thboop
737. thecodrr
738. thejameskyle
739. theotherphil
740. thlorenz
741. thomasb
742. Thomasdezeeuw
743. thomcc
744. thorn0
745. thymikee
746. TianyiShi2001
747. ticki
748. tigleym
749. timdeschryver
750. timdorr
751. timneutkens
752. TimothyGu
753. titanism
754. tjholowaychuk
755. tmpvar
756. tobz
757. TooTallNate
758. torch2424
759. toyobayashi
760. trent-codecov
761. trieloff
762. tripod
763. trishume
764. tromey
765. troygoode
766. TrySound
767. tschneidereit
768. tspiteri
769. tunnckoCore
770. tyler
771. TyOverby
772. types
773. typescript-bot
774. udoprog
775. UlisesGascon
776. unshiftio
777. utkarshgupta137
778. Uzlopak
779. v1
780. valenting
781. vercel-release-bot
782. Veykril
783. VictorKoenders
784. Victorystick
785. vitaly
786. vitebot
787. vjeux
788. vkurchatkin
789. vladikoff
790. vorner
791. vorot93
792. Vtec234
793. v-thakkar
794. warner
795. waych
796. waywardmonkeys
797. WebReflection
798. wesleytodd
799. weswigham
800. whitequark
801. WilcoFiers
802. withoutboats
803. wooorm
804. wusyong
805. xorgy
806. xStrom
807. xtuc
808. xyc
809. yaacovCR
810. yaahc

811. yannickcr
812. yocontra
813. yoshuawuyts
814. ysthakur
815. yurydelendik
816. yurys
817. yyx990803
818. zakarumych
819. zbraniecki
820. zdahbi
821. zeenix
822. zeit-bot
823. zekth
824. zensh
825. zertosh
826. zesterer
827. Zirro
828. zlafil
829. ZoeyR

B. Gebruikte afkortingen

In ons vakgebied wordt vaak gebruikgemaakt van afkortingen. Niet altijd is meteen duidelijk waar een bepaalde afkorting voor staat. Vandaar dat we in deze appendix proberen om een overzicht te geven van de in dit rapport gebruikte afkortingen. In veel gevallen zal hier een Engelstalige “verklarende beschrijving” zijn opgenomen.

2FA	Two-Factor Authentication
ACL	Access Control List
API	Application Programming Interface
ARP	Address Resolution Protocol
C2	Command & Control
CA	Certificate Authority
CSB	Centraal Stembureau
DNS	Domain Name System
DTD	Document Type Definition
ECDSA	Elliptic Curve Digital Signature Algorithm
EML	Election Markup Language
GSB	Gemeentelijk Stembureau
HSB	Hoofd Stembureau
HTTP	HyperText Transfer Protocol
HTTPS	Secure HTTP
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPv4	IP version 4
IPv6	IP version 6
IT	Information Technology
MitM	Man-in-the-Middle
OSV	Ondersteunende Software Verkiezingen
PDF	Portable Document Format
PKI	Public Key Infrastructure
QR	Quick Response
RSA	Rivest, Shamir and Adleman
SBOM	Software Bill Of Materials
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module
UI	User Interface
USB	Universal Serial Bus
XML	eXtensible Markup Language
XXE	XML eXternal Entity