FOX IT
part of nccgroup

CLASSIFICATIE
PUBLIC

Beveiligingsonderzoek OSV-2020

Rapportage

Datum	9 maart 2022
Referentie	PR-210537
Opdrachtgever	Kiesraad
Versie	1.1

FOR A MORE SECURE SOCIETY

DOCUMENTCLASSIFICATIE

Dit document is geclassificeerd als PUBLIC.

Enig misbruik van dit document of de informatie in het document is niet toegestaan. Fox-IT aanvaardt geen aansprakelijkheid voor enig ongeautoriseerd gebruik of misbruik van voorliggend document door een derde partij of schade ontstaan door de inhoud van het document.

Fox-IT B.V.
Olof Palmestraat 6
2616 LM Delft
Postbus Box 638
2600 AP Delft
Nederland

T +31 (0)15 284 79 99
F +31 (0)15 284 79 90
fox@fox-it.com
www.fox-it.com

Copyright © 2022 Fox-IT B.V.
Alle rechten voorbehouden. Niets uit deze uitgave mag worden verveelvoudigd, opgeslagen in een geautomatiseerd gegevensbestand, of openbaar gemaakt, in enige vorm of op enigerlei wijze, hetzij elektronisch, mechanisch, door fotokopieën, opnamen of op enige andere manier, zonder voorafgaande schriftelijke toestemming van Fox-IT BV.

Handelsmerk
Fox-IT en het logo van Fox-IT zijn handelsmerken van Fox-IT BV.
Alle andere in dit document opgenomen handelsmerken zijn eigendom van de genoemde organisaties.

Managementsamenvatting

In opdracht van de Kiesraad heeft Fox-IT een technisch beveiligingsonderzoek (penetratietest) uitgevoerd op de OSV-2020 software. OSV-2020 is de software die wordt gebruikt ter ondersteuning van het verkiezingsproces. Het doel van het onderzoek was het identificeren van kwetsbaarheden in de OSV-2020 software. Het onderzoek heeft zich gericht op het deel van de software dat ondersteunend is aan het vaststellen van de verkiezingsuitslag en de zetelverdeling voor de gemeenteraadsverkiezingen.

Tijdens dit onderzoek is de applicatie volgens de handleiding geïnstalleerd. Vervolgens zijn de applicatie en de op het systeem aanwezige bestanden en programma's onderzocht op kwetsbaarheden. De veiligheid van de applicatie is door de specialisten van Fox-IT beoordeeld en mede getoetst aan enkele door de Kiesraad geselecteerde publieke standaarden voor (web)applicatiebeveiliging.

De OSV-2020 software behoort volgens de procedurele voorschriften te worden ingezet binnen het geheel van individuele stembureaus op verschillende niveaus, waarbij de software behoort te worden gebruikt in een geïsoleerd netwerk in een afgesloten ruimte. Mits zodanig toegepast, verlaagt dit de risico's van alle aangetroffen kwetsbaarheden.

In totaal zijn tijdens het onderzoek 18 kwetsbaarheden gevonden waaraan naar het oordeel van Fox-IT een gemiddeld of laag risico is verbonden. Het onderzoek heeft geen bevindingen met een hoog of kritiek risico opgeleverd. De bevindingen als volgt worden samengevat:

- De bescherming van de integriteit en authenticiteit van gegevens is niet toereikend;
- Het identiteits-, toegangs- en sessiebeheer van de webapplicatie bevat kwetsbaarheden;
- Het aanvalsoppervlak van de server en de clients en de verbinding daartussen kan worden verkleind.

De geïdentificeerde kwetsbaarheden kunnen als gevolg hebben dat een kwaadwillende gebruiker, die toegang krijgt tot de afgesloten ruimte waarin de software wordt gebruikt, de mogelijkheid heeft om stemtotalen aan te passen. Omdat de logging van de applicatie beïnvloedbaar is, wordt dit mogelijk niet tijdig opgemerkt. Dit risico is niet aanwezig bij de gemeenteraadsverkiezingen, omdat daar geen digitale overdracht van gegevens plaatsvindt.

De juiste implementatie van de software volgens de richtlijnen die betrekking hebben op de infrastructuur waarop applicatie wordt ingezet, is van significant belang. De manier waarop OSV-2020 nu wordt ingezet – decentraal en geïsoleerd – vormt een belangrijke mitigerende factor in de kwetsbaarheid van het aggregatieproces voor doelbewuste manipulaties.

Op basis van dit onderzoek geeft Fox-IT de volgende aanbevelingen:

- Verbeter de bescherming van de integriteit en authenticiteit van gegevens door een juiste toepassing van cryptografie;
- Verbeter het identiteits-, toegangs- en sessiebeheer van de webapplicatie;
- Verklein het aanvalsoppervlak.
- Voorzie in voldoende tijd en ruimte om ook de andere programma's die deel uitmaken van de OSV-2020 software (KS- en PP-applicaties) te onderzoeken;
- Bezie de uitkomsten van dit onderzoek en het volledige onderzoek naar de code van de applicaties – uitgevoerd door een andere partij – nadrukkelijk in samenhang;
- Voorzie erin dat de OSV-2020 software zoveel mogelijk decentraal en geïsoleerd gebruikt wordt om zo het risico op manipulatie van het aggregatieproces te beperken.

Inhoudsopgave

DOCUMENTCLASSIFICATIE **2**

Managementsamenvatting **3**

Inhoudsopgave **4**

1 Opdracht details **6**
1.1 Onderzoeksvraag 6
1.2 Aanpak 7
1.3 Kaders 8
1.3.1 Scope 8
1.3.2 Toepasselijke beperkingen 9

2 Conclusies en aanbevelingen **10**
2.1 Conclusies 10
2.2 Aanbevelingen 12
2.2.1 Verbeter de bescherming van de integriteit en authenticiteit van gegevens door een juiste toepassing van cryptografie 13
2.2.2 Verbeter het identiteits-, toegangs- en sessiebeheer van de webapplicatie 13
2.2.3 Verklein het aanvalsoppervlak 13

3 Bijlagen **14**
3.1 Risicomatrix 14
3.2 Risicocorrelatie 15
3.3 CVSS 3.1 Scores 16
3.4 Bevindingen 17
Bevinding 1: Sessies blijven geldig na wachtwoordwijziging 18
Bevinding 2: Gelijktijdige sessies met enkel account mogelijk 19
Bevinding 3: Stemdata wordt niet door middel van digitale handtekeningen gecontroleerd 20
Bevinding 4: Digitale ondertekening van meerdere installatiebestanden ontbreekt 23
Bevinding 5: Zwakke cryptografie in gebruik voor encryptie databasewachtwoord 26
Bevinding 6: Gebruik van standaard sleutel voor encryptie databasewachtwoord 27
Bevinding 7: Least privilege principe niet geïmplementeerd bij installatie 28
Bevinding 8: Logregelinjectie mogelijk zonder authenticatie 29
Bevinding 9: Niet-ondersteunde software in gebruik 31
Bevinding 10: Databasewachtwoord kan door elke gebruiker op de server worden achterhaald 32
Bevinding 11: Niet alle wachtwoorden opgeslagen in een versleutelde key store 34
Bevinding 12: Niet genoeg PBKDF2-iteraties 36
Bevinding 13: Het huidige wachtwoordbeleid is niet veilig genoeg 38
Bevinding 14: Melding van het aantal foutieve inlogpogingen ontbreekt 39
Bevinding 15: Geen tweede factor benodigd bij inloggen 40
Bevinding 16: Clientside air-gap controle ontbreekt 41
Bevinding 17: SSL/TLS-configuratie onvoldoende veilig 42
Bevinding 18: Gebruik van correct servercertificaat wordt niet afgedwongen 43

4 Richtlijnen **45**
4.1 TLS van NCSC 45
4.2 Webapplicaties van NCSC 47
4.3 ASVS van OWASP 50

Documentbeheer **65**

1 Opdracht details

1.1 Onderzoeksvraag

De Kiesraad heeft Fox-IT gevraagd om de beveiliging van Ondersteunende Software Verkiezingen 2020, hierna te noemen OSV-2020, te onderzoeken. OSV-2020 wordt gebruikt ter ondersteuning van het verkiezingsproces. In OSV-2020 zijn, ten opzichte van OSV, veranderingen doorgevoerd in de software op het vlak van beveiliging en door gebruik te maken van actuelere software-onderdelen. De software wordt als volgt onderverdeeld:

- Applicaties voor kandidaatstelling voor politieke partijen en het centraal stembureau;
- Een applicatie voor de vaststelling van de uitslag en zetelverdeling voor onder andere gemeenten en centraal stembureau.

De scope van dit onderzoek betrof OSV-2020 en bevat de drie applicaties waaruit OSV-2020 bestaat, te weten:

- Politieke partijen (PP)
- Kandidaatstelling (KS)
- Uitslagvaststelling (U)

Hoewel de volledige software in scope was, heeft de Kiesraad aangegeven dat de nadruk diende te liggen op de applicatie die ondersteunend is aan het vaststellen van de verkiezingsuitslag en de zetelverdeling (U). Het onderzoek heeft zich gericht op de installatie zoals gebruikt voor de gemeenteraadsverkiezingen. Binnen de tijd die beschikbaar was voor dit onderzoek kon alleen dit software-onderdeel onderzocht worden. Het uitvoeren van een code review maakte geen onderdeel uit van deze opdracht.

Het uitgevoerde onderzoek diende om de volgende onderzoeksvragen te beantwoorden:

- Welke kwetsbaarheden en risico’s op het gebied van informatiebeveiliging zijn te onderkennen in het testobject?
- Welke maatregelen kunnen worden getroffen om de geconstateerde risico’s te mitigeren?
- Hoe groot is het risico dat ontstaat door het gebruik van de end-of-life software Thorntail als onderdeel van OSV-2020?

1.2 Aanpak

Bij het beantwoorden van de in paragraaf 1.1 weergegeven onderzoeksvragen is de volgende aanpak gehanteerd.

De U-applicatie is onderzocht als volgt:

- De installatiebestanden:
  - zoals die zijn aangeleverd;
  - na decomplilatie.
- De applicatie na installatie:
  - Bestanden op het bestandssysteem;
  - De webapplicatie:
    - Verwerking van externe gegevens door OSV-2020:
      - Afhandeling van externe invoer in de vorm van bestanden;
      - Afhandeling van externe invoer in de vorm van interactie met de gebruiker.
    - De in OSV-2020 aanwezige maatregelen ter voorkoming van manipulatie, waaronder
      - Integriteitswaarborgen van de gegevens wanneer deze geëxporteerd worden;

Ten behoeve van het onderzoek zijn geen accounts aangeleverd; de gebruikte accounts zijn alle lokaal aangemaakt en gebruikt. De applicatie is op verschillende virtuele omgevingen geïnstalleerd, waarbij elke omgeving in ieder geval een server en mogelijk een client bevatte. Deze virtuele omgeving was van buitenaf niet bereikbaar. De applicatie van de server is ten behoeve van het onderzoek geïnstalleerd op de besturingssystemen Ubuntu Linux 20.04, Windows 10 en macOS Catalina. De clients maakten gebruik van Ubuntu Linux 20.04 en Windows 10. In sommige gevallen is de server tevens gebruikt als client om het scenario te ensceneren waarbij één computer gebruikt wordt voor zowel de server als de client. Gedurende de installatie en gebruik van de applicatie is gebruik gemaakt van de meegeleverde documentatie. De gebruikte documentatie voor zowel de installatie als het gebruik van de applicatie werd meegeleverd in de in de technische scope genoemde zip-bestanden, in de map ‘handleidingen’.

Het onderzoek is uitgevoerd tussen 3 januari en 21 januari 2022.

1.3 Kaders

Deze paragraaf schetst de kaders van het onderzoek zoals deze door de Kiesraad en Fox-IT overeengekomen zijn.

1.3.1 Scope

De volgende door de Kiesraad aangeleverde bestanden zijn gebruikt voor het onderzoek:

	Bestandsnaam	SHA256-hash
1	osv2020-u-installer-gr-1.4.1.zip	1791 25AF BCE5 E75C 4E0E 141C FA0E F230 90C1 9D8E 91CB 03F0 33C8 6E20 3164 84C7
2	GR1-13zetels.zip	CA90 B135 6B00 8A4C 99D5 37D0 0D69 DB07 24E8 5335 4943 AB5C 8182 0B1B 5C77 29E2

Tabel 1 – De onderzochte bestanden

Bestand 1 bevat de installatiebestanden die het mogelijk maken om de U-applicatie te installeren op een daarvoor bedoeld systeem. Onderstaand is een overzicht te zien van de meegeleverde bestanden in het bestand `osv2020-u-installer-gr-1.4.1.zip`. Dit bestand bevat veel ingebedde zip-bestanden, daarom zijn hier enkel de eerste twee lagen afgebeeld. De bevindingen in paragraaf 3.4 refereren aan deze tabel.

Bestand 3 bevat stemdata die gebruikt is als invoer tijdens het testen van de applicatie. Op deze manier kan een verkiezingssituatie worden geënsceneerd waarbij het mogelijk is het aantal uitgebrachte stemmen te tellen.

Het aangeleverde bestand met de U-applicatie had de volgende mappenstructuur en bestanden:

- `handleidingen`
  - `handleiding_installer_OSV2020-U-GR-v1.4.1.pdf`
  - `handleiding_kort_OSV2020-U-GR-v1.4.1.pdf`
  - `handleiding_OSV2020-U-GR-v1.4.1.pdf`
- `installatiebestanden`
  - `bin`
  - `izpack`
  - `jdk-linux`
  - `jdk-windows`
  - `vapp-metadata.zip`
  - `vapp.properties`
- `installer-linux-hdpi.sh`
- `installer-linux.sh`
- `installer-mac-package.zip`
- `Installer-windows.exe`
- `__MACOSX`
  - `Installatiebestanden`

Tabel 2 – Overzicht van de bestanden in osv2020-u-installer-gr-1.4.1.zip

De webapplicatie is geschreven in Java en is na installatie te benaderen via de webbrowser. Via de webbrowser kunnen gebruikers inloggen met een gebruikersaccount dat aangemaakt is gedurende de installatie. De webapplicatie maakt gebruik van een database waar de versleutelde wachtwoorden van gebruikersaccounts in staan. Daarnaast bevat de database de uitgebrachte stemmen.

1.3.2 Toepasselijke beperkingen

De volgende beperkingen zijn van toepassing op het onderzoek:

- Het onderzoek richtte zich op de applicatie Uitslagvaststelling (OSV2020-U);
- De applicaties OSV2020-PP en OSV2020-KS zijn niet onderzocht;
- Het onderzoek was puur technisch van aard: het onderzoeken van de wettelijke kaders om na te gaan of deze afdoende zijn om een adequaat beveiligingsniveau te bereiken viel niet binnen dit onderzoek;
- Een onderzoek van de broncode van de applicaties viel niet binnen de scope van dit onderzoek.
- Bij de uitvoer van het onderzoek is de aanname gedaan dat het systeem ingericht wordt zoals voorgeschreven en daarmee als enige functie heeft de OSV-2020-applicatie te draaien, in een geïsoleerd netwerk in een afgesloten ruimte.

2 Conclusies en aanbevelingen

Dit onderzoek heeft tot doel om de volgende onderzoeksvragen te beantwoorden:
- Welke kwetsbaarheden en risico's op het gebied van informatiebeveiliging zijn te onderkennen in het testobject?
- Welke maatregelen kunnen worden getroffen om de geconstateerde risico's te mitigeren?
- Hoe groot is het risico dat ontstaat door het gebruik van de end-of-life software Thorntail als onderdeel van OSV-2020?

De onderzoeksvragen worden in paragraaf 1.1 beschreven op basis van de in paragraaf 1.2 beschreven aanpak.

2.1 Conclusies

Tijdens de penetratietest op het deel van de OSV-2020 software dat ondersteunend is aan het vaststellen van de verkiezingsuitslag en de zetelverdeling voor de gemeenteraadsverkiezingen zijn 18 kwetsbaarheden aangetroffen waaraan naar het oordeel van Fox-IT een gemiddeld of laag risico is verbonden.

De geïdentificeerde kwetsbaarheden kunnen als gevolg hebben dat een kwaadwillende gebruiker, die toegang krijgt tot de afgesloten ruimte waarin de software wordt gebruikt, de mogelijkheid heeft om stemtotalen aan te passen. Omdat de logging van de applicatie beïnvloedbaar is, wordt dit mogelijk niet tijdig opgemerkt. Dit risico is niet aanwezig bij de gemeenteraadsverkiezingen, omdat daar geen digitale overdracht van gegevens plaatsvindt.

De geïdentificeerde kwetsbaarheden zijn in detail gedocumenteerd in paragraaf 3.4. Op basis van de verschillende kwetsbaarheden die gevonden zijn gedurende het onderzoek heeft Fox-IT de volgende conclusies:

De bescherming van de integriteit en authenticiteit van gegevens is niet toereikend. Deze conclusie wordt ondersteund door de volgende bevindingen:
- Stemdata wordt niet door middel van digitale handtekeningen gecontroleerd;
- Digitale ondertekening van meerdere installatiebestanden ontbreekt;
- Zwakke cryptografie in gebruik voor encryptie databasewachtwoord;
- Gebruik van standaard sleutel voor encryptie databasewachtwoord;
- Niet alle wachtwoorden opgeslagen in een versleutelde key store;
- Niet genoeg PBKDF2-iteraties.

Het identiteits-, toegangs- en sessiebeheer van de webapplicatie bevat kwetsbaarheden. Deze conclusie wordt ondersteund door de volgende bevindingen:
- Sessies blijven geldig na wachtwoordwijziging;
- Gelijktijdige sessies met enkel account mogelijk;
- Het huidige wachtwoordbeleid is niet veilig genoeg;
- Melding van het aantal foutieve inlogpogingen ontbreekt;
- Geen tweede factor benodigd bij inloggen;
- Logregelinjectie mogelijk zonder authenticatie.

Het aanvalsoppervlak van de server en de clients en de verbinding daartussen kan worden verkleind. Deze conclusie wordt ondersteund door de volgende bevindingen:

- Niet-ondersteunde software in gebruik;
- Databasewachtwoord kan door elke gebruiker op de server worden achterhaald;
- Clientside air-gap controle ontbreekt;
- Least privilege principe niet geïmplementeerd bij installatie;
- SSL/TLS-configuratie onvoldoende veilig;
- Gebruik van correct servercertificaat wordt niet afgedwongen.

De manier waarop OSV-2020 nu wordt ingezet – decentraal en geïsoleerd – vormt een belangrijke mitigerende factor in de kwetsbaarheid van het aggregatieproces voor doelbewuste manipulaties. De OSV-2020 software behoort volgens de procedurele voorschriften te worden ingezet binnen het geheel van individuele stembureaus op verschillende niveaus, waarbij de software behoort te worden gebruikt in een geïsoleerd netwerk in een afgesloten ruimte. Mits zodanig toegepast, verlaagt dit de risico's van verlaagt de risico's voor alle aangetroffen kwetsbaarheden. De juiste implementatie van de software volgens de richtlijnen die betrekking hebben op de infrastructuur waarop applicatie wordt ingezet, is daarom van significant belang.

2.2 Aanbevelingen

Dit hoofdstuk beschrijft de strategische aanbevelingen die de Kiesraad kunnen helpen om beveiliging van OSV-2020 verhogen. Aanbevelingen voor individuele technische bevindingen staan beschreven bij de corresponderende bevinding in paragraaf 3.4. Fox-IT raadt aan om een overkoepelde roadmap op de stellen waarin individuele bevindingen toegewezen worden aan verantwoordelijke personen of groepen. Deze dragen de verantwoordelijkheid voor het oplossen van de bevindingen binnen een geplande tijdspanne. Fox-IT constateert dat varianten van een aantal bevindingen uit het onderhavige onderzoek reeds in eerdere beveiligingsonderzoeken zijn vastgesteld. Voor het oplossen van individuele bevindingen moet altijd gekeken worden naar de volledige omvang van een probleem omdat de kans bestaat dat niet alle gevallen van het probleem zijn geïdentificeerd gedurende het onderzoek.

De vastgestelde tijdslijnen voor het gebruik van de applicatie verlangen wellicht dat geïdentificeerde kwetsbaarheden (deels) geaccepteerd moeten en niet (volledig) opgelost kunnen worden. Wanneer dit het geval is dienen deze besluiten duidelijk gedocumenteerd en bewaard te worden in de relevante documentatie.

Tijdens de besluitvorming omtrent het oplossen van een bevinding is het verstandig om ook te proberen om de oorzaak van het probleem op te lossen. De volgende hoofdzaken zijn geïdentificeerd tijdens dit onderzoek:

- Verbeter de bescherming van de integriteit en authenticiteit van gegevens door een juiste toepassing van cryptografie;
- Verbeter het identiteits-, toegangs- en sessiebeheer van de webapplicatie;
- Verklein het aanvalsoppervlak.

De beschreven aanbevelingen zullen nooit het risico volledig kunnen wegnemen, maar dragen bij aan een zo veilig mogelijk aggregatieproces van de stemtotalen. Daarnaast zijn de volgende aanbevelingen van toepassing:

- Tijdens dit en het vorige uitgevoerde onderzoek van OSV-2020 is alleen de U-applicatie onderzocht. Voorzie in voldoende tijd en ruimte om ook de andere programma’s die deel uitmaken van de OSV-2020 software (KS- en PP-applicaties) te onderzoeken;
- Bezie de uitkomsten van dit onderzoek en het volledige onderzoek naar de code van de applicaties – uitgevoerd door een andere partij – nadrukkelijk in samenhang;
- Voorzie erin dat de OSV-2020 software zoveel mogelijk decentraal en geïsoleerd gebruikt wordt om zo het risico op manipulatie van het aggregatieproces te beperken.

De technische aanbevelingen naar aanleiding van de bevindingen worden in de onderstaande paragrafen kort uiteengezet en in paragraaf 3.4 verder toegelicht.

2.2.1 Verbeter de bescherming van de integriteit en authenticiteit van gegevens door een juiste toepassing van cryptografie

De volgende technische aanbevelingen, die zijn afgeleid van de aanbevelingen bij de betreffende bevindingen, zijn van toepassing:

- Maak gebruik van digitale ondertekening van alle bestanden waarbij de integriteit en authenticiteit van de data belang is, en zorg ervoor dat iedere gebruiker een persoonsgebonden certificaat gebruikt.
- Onderteken het Java-bestand met daarin alle bestanden die nodig zijn om de applicatie te installeren, of controleer de hashwaarde van de installatiebestanden vóór installatie.
- Gebruik sterke versleutelingsalgoritmen.
- Maak geen gebruik van een standaard sleutel voor het versleutelen van het databasewachtwoord, maar gebruik een sleutel die per installatie verschilt.
- Gebruik één uniforme veilige manier van opslag van gevoelige data zoals wachtwoorden.
- Gebruik een recentere Key Derivation Function (KDF) zoals Argon2 of zorg voor meer PBKDF2-iteraties.

2.2.2 Verbeter het identiteits-, toegangs- en sessiebeheer van de webapplicatie

De volgende technische aanbevelingen, die zijn afgeleid van de aanbevelingen bij de betreffende bevindingen, zijn van toepassing:

- Sluit alle actieve sessies van het betreffende account af wanneer een wachtwoord gewijzigd wordt.
- Blokkeer nieuwe inlogpogingen wanneer iemand is ingelogd op het gebruikersaccount, invalideer de eerdere actieve sessie en/of laat een waarschuwing zien
- Verbeter het wachtwoordbeleid en stimuleer het gebruik van sterke wachtwoorden.
- Laat een melding zien die de gebruiker attendeert op het feit dat één of meerdere foutieve inlogpogingen gedaan zijn, als dit het geval is.
- Voeg een tweede factor toe aan de authenticatie.
- Filter en valideer alle gebruikersinvoer, en maak hierbij waar mogelijk gebruik van een allowlist.

2.2.3 Verklein het aanvalsoppervlak

De volgende technische aanbevelingen, die zijn afgeleid van de aanbevelingen bij de betreffende bevindingen, zijn van toepassing:

- Spreek met de leverancier van de software af dat, en controleer of, de meest recente versie van software wordt meegeleverd. Onderzoeken of gebruik gemaakt kan worden van opvolgers van Thorntail: Wildfly en Quarkus.
- Geef het databasewachtwoord niet mee als command line argument.
- Forceer ook het gebruik van een air-gap controle aan de gebruikerskant. Waarschuw de gebruiker indien dit niet het geval is en beperk de toegang tot de applicatie vanaf die client.
- Maak tijdens de installatie een account aan waarmee niet kan worden ingelogd op het systeem, maar waarmee alleen de applicatie kan worden opgestart (bijvoorbeeld met behulp van een service). Dit account moet zo min mogelijk rechten hebben.
- Besteed meer aandacht aan het veilig configureren van SSL/TLS.
- Lever het OSV Root CA certificaat (waarmee het servercertificaat getekend is) mee en vereis procedureel dat dit in de internetbrowser, die voor het gebruik van de OSV-applicatie wordt gebruikt, beschikbaar is, zodat geen waarschuwing over ongeldige TLS-certificaten wordt weergegeven.

3 Bijlagen

Dit hoofdstuk bevat een overzicht van tijdens deze test gedane bevindingen. Daarnaast bevat dit hoofdstuk een risicocorrelatie waarmee een beeld geschetst wordt van hoe de tijdens deze test aangetroffen bevindingen door een aanvaller gebruikt zouden kunnen worden.

3.1 Risicomatrix

Onderstaande tabel geeft een overzicht van alle bevindingen die tijdens deze test zijn gedaan. De bevindingen zijn gesorteerd op het geconstateerde risico van hoog naar laag.

Bevinding	Omschrijving	Risico
1	Sessies blijven geldig na wachtwoordwijziging	GEMIDDELD
2	Gelijktijdige sessies met enkel account mogelijk	GEMIDDELD
3	Stemdata wordt niet door middel van digitale handtekeningen gecontroleerd	GEMIDDELD
4	Digitale ondertekening van meerdere installatiebestanden ontbreekt	GEMIDDELD
5	Zwakke cryptografie in gebruik voor encryptie databasewachtwoord	GEMIDDELD
6	Gebruik van standaard sleutel voor encryptie databasewachtwoord	GEMIDDELD
7	Least privilege principe niet geïmplementeerd bij installatie	LAAG
8	Logregelinjectie mogelijk zonder authenticatie	LAAG
9	Niet-ondersteunde software in gebruik	LAAG
10	Databasewachtwoord kan door elke gebruiker op de server worden achterhaald	LAAG
11	Niet alle wachtwoorden opgeslagen in een versleutelde key store	LAAG
12	Niet genoeg PBKDF2-iteraties	LAAG
13	Het huidige wachtwoordbeleid is niet veilig genoeg	LAAG
14	Melding van het aantal foutieve inlogpogingen ontbreekt	LAAG
15	Geen tweede factor benodigd bij inloggen	LAAG
16	Clientside air-gap controle ontbreekt	LAAG
17	SSL/TLS-configuratie onvoldoende veilig	LAAG
18	Gebruik van correct servercertificaat wordt niet afgedwongen	LAAG

3.2 Risicocorrelatie

OSV-2020 mag alleen worden gebruikt in een geïsoleerd netwerk in een afgesloten ruimte, op een systeem dat uitsluitend voor OSV-2020 wordt gebruikt. Dit verlaagt de kans van optreden voor alle geïdentificeerde kwetsbaarheden. Hiermee is rekening gehouden bij de inschatting van de kans van optreden van de individuele bevindingen.

Onderstaande risicocorrelatie beschrijft hoe individuele bevindingen, zoals beschreven in paragraaf 3.4, gecombineerd kunnen worden en zo tot een aanval kunnen leiden waarbij de integriteit van de stemdata mogelijk niet meer gewaarborgd kan worden. Hierbij wordt verondersteld dat een kwaadwillende (fysieke) toegang heeft tot de server, waar de applicatie op geïnstalleerd is, of client, welke verbinding maakt met de server.

Een kwaadwillende kan het wachtwoord van een gebruikersaccount raden (bevinding 14) en hier vervolgens op inloggen vanwege het ontbreken van een tweede factor bij het inloggen (bevinding 15). Omdat het mogelijk is dat gelijktijdige sessies met een enkel account bestaan (bevinding 2), kan de aanvaller zich nu voordoen als de rechtmatige eigenaar van het gebruikersaccount. Zelfs als het vermoeden ontstaat dat er misbruik wordt gemaakt van een account en dientengevolge het wachtwoord wordt gewijzigd, dan blijft het mogelijk om een reeds gecreëerde sessie te gebruiken (bevinding 1). Tot slot wordt de ingevoerde stemdata niet gecontroleerd aan de hand van een digitale handtekening (bevinding 3). Een kwaadwillende kan hierdoor aangepaste stemdata invoeren zonder dat dit direct te herleiden valt naar deze persoon.

Verder kan een kwaadwillende het wachtwoord van een gebruikersaccount achterhalen is door het versleutelde databasewachtwoord op te sporen (bevinding 7, 10 en 11), dit te ontsleutelen (bevinding 5 en 6), en mogelijk het wachtwoord van een gebruikersaccount te achterhalen (bevinding 12 en 13). Ook wanneer deze laatste stap niet lukt, heeft de kwaadwillende nog steeds direct toegang tot de database waar alle stemuitslagen en andere gevoelige data in is opgeslagen.

Tot slot heeft een aanvaller de mogelijkheid de te installeren software van tevoren aan te passen, omdat niet alle installatiebestanden ondertekend zijn (bevinding 4). Op deze manier zouden de ingevoerde stemtotalen aangepast kunnen worden omdat een aanvaller het tellingsproces zou kunnen hebben beïnvloed.

Het is echter belangrijk om op te merken dat dit slechts enkele scenario’s zijn, en dat het waarschijnlijk is dat er andere scenario’s uitgevoerd kunnen worden.

3.3 CVSS 3.1 Scores

Onderstaande tabel bevat de risico-inschatting van de individuele bevindingen als beschreven op de volgende pagina, hierbij zijn enkel de *environmental* CVSS scores voor bevindingen met een risico classificatie van 'gemiddeld' bepaald. Bij het bepalen van de scores is uitgegaan van het gebruik van de applicaties in een geïsoleerde omgeving waarbij de kwaadwillende fysieke toegang nodig heeft tot de server waar de webapplicatie op is geïnstalleerd. De scores zijn bepaald zodanig dat bovenstaande eigenschappen in acht genomen zijn.

Bevinding	Beschrijving van bevinding gevolgd door CVSS 3.1 Vector	Environmental Score
1	Sessies blijven geldig na wachtwoordwijziging CVSS:3.1/AV:P/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:H/CR:X/IR:X/AR:X/MAV:P/MAC:L/MPR:L/MUI:N/MS:U/MC:L/MI:L/MA:H	5.4
2	Gelijktijdige sessies met enkel account mogelijk CVSS:3.1/AV:P/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:H/CR:X/IR:X/AR:X/MAV:P/MAC:L/MPR:L/MUI:N/MS:U/MC:L/MI:L/MA:H	5.4
3	Stemdata wordt niet door middel van digitale handtekeningen gecontroleerd CVSS:3.1/AV:P/AC:L/PR:H/UI:N/S:U/C:L/I:H/A:N/CR:L/IR:H/AR:X/MAV:P/MAC:L/MPR:H/MUI:N/MS:U/MC:L/MI:H/MA:X	5.8
4	Digitale ondertekening van meerdere installatiebestanden ontbreekt CVSS:3.1/AV:P/AC:L/PR:H/UI:N/S:U/C:N/I:H/A:H/CR:X/IR:H/AR:H/MAV:P/MAC:L/MPR:H/MUI:N/MS:U/MC:X/MI:H/MA:H	6.2
5	Zwakke cryptografie in gebruik voor encryptie databasewachtwoord CVSS:3.1/AV:P/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H/CR:H/IR:H/AR:H/MAV:P/MAC:H/MPR:H/MUI:N/MS:U/MC:H/MI:H/MA:H	6.1
6	Gebruik van standaard sleutel voor encryptie databasewachtwoord CVSS:3.1/AV:P/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H/CR:H/IR:H/AR:H/MAV:P/MAC:H/MPR:H/MUI:N/MS:U/MC:H/MI:H/MA:H	6.1

3.4 Bevindingen

Inschattingen van het risico van de aangetroffen kwetsbaarheden zijn gebaseerd op de eigen inschatting van de specialisten van Fox-IT. Het is de verantwoordelijkheid van het management van Kiesraad om afhankelijk van de specifieke processen en omstandigheden het risiconiveau te bepalen. Ook doet Fox-IT een concrete technische aanbeveling per kwetsbaarheid, die uitlegt hoe de kwetsbaarheid kan worden verholpen dan wel hoe het risico kan worden gereduceerd. Ook hierbij is het echter de eindverantwoordelijkheid van Kiesraad om de kosten en baten van het overnemen van dit advies tegen elkaar af te wegen. Het rapport van Fox-IT biedt de informatie op basis waarvan u een geïnformeerd besluit kunt nemen.

Bij het inschatten van het risico baseren de specialisten van Fox-IT zich op het volgende:

- Waarschijnlijkheid – de kans dat een aanvaller misbruik zal (kunnen) maken van de beschreven kwetsbaarheid
- Gevolgen – de impact die misbruik van de beschreven kwetsbaarheid zou kunnen hebben voor het netwerk/de applicatie in scope

Volgens de formule “Risico = Waarschijnlijkheid x Impact” leidt dat tot het volgende schema:

Waarschijnlijkheid \ Impact	Laag	Gemiddeld	Hoog
Laag	LAAG	LAAG	GEMIDDELD
Gemiddeld	LAAG	GEMIDDELD	HOOG
Hoog	GEMIDDELD	HOOG	KRITIEK

In deze rapportage is bij het beoordelen van de kans van optreden van de bevindingen de aanname gedaan dat het systeem ingericht wordt zoals voorgeschreven en daarmee als enige functie heeft de OSV-2020-applicatie te draaien, in een geïsoleerd netwerk in een afgesloten ruimte. Dit verlaagt de kans van optreden voor alle aangetroffen kwetsbaarheden.

Bevinding 1: Sessies blijven geldig na wachtwoordwijziging			
Risico Classificatie	Gemiddeld risico	**Waarschijnlijkheid**	Laag
Categorie	Authenticatie	**Gevolgen**	Hoog

Betreft
OSV-2020-U

Omschrijving
Wanneer het wachtwoord van een account gewijzigd wordt, dan worden eventuele bestaande sessies van deze gebruiker niet afgesloten. Deze gebruiker blijft ingelogd in de bestaande sessie en kan alle functionaliteit van het gebruikersaccount blijven gebruiken.

Risico
Indien een wachtwoord uitgelekt is en daarom gewijzigd wordt, dienen alle actieve sessies van het account afgesloten te worden. In de huidige situatie is dat niet het geval. Hierdoor kan het voorkomen dat een aanvaller die het gelekte wachtwoord misbruikt, niet automatisch uitgelogd wordt wanneer het wachtwoord gewijzigd wordt. Daarmee houdt de aanvaller toegang tot de webapplicatie ondanks dat het wachtwoord gewijzigd is.

Aanbeveling
Sluit alle actieve sessies van het betreffende account af wanneer een wachtwoord gewijzigd wordt.

Bevinding 2: Gelijktijdige sessies met enkel account mogelijk			
Risico Classificatie	Gemiddeld risico	**Waarschijnlijkheid**	Laag
Categorie	Configuratie	**Gevolgen**	Hoog

Betreft
OSV-2020-U

Omschrijving
Het is mogelijk om meerdere keren met eenzelfde account in te loggen en daarmee meerdere valide sessies actief te hebben.

Risico
Omdat alle accounts persoonsgebonden zijn, dienen de accounts nooit gedeeld te worden. Het is echter mogelijk om meerdere keren tegelijkertijd met hetzelfde account in te loggen. Hierdoor kunnen accountgegevens gedeeld worden en kan hetzelfde account door meerdere personen tegelijkertijd gebruikt worden. Daarnaast is het dus onvoldoende om het wachtwoord van een gebruikersaccount te wijzigen, om ongeautoriseerd gebruik te voorkomen, wanneer het vermoeden van misbruik bestaat.

Aanbeveling
Zorg ervoor dat het slechts mogelijk is om 1 sessie tegelijkertijd te hebben op de OSV-applicatie. Dit kan bijvoorbeeld door een reeds bestaande sessie te invalideren bij een geldige inlogpoging. Het is wenselijk om een descriptieve melding te geven dat er elders een gebruiker is ingelogd met deze account, vanwege de detectieve waarde van die melding.

Bevinding 3: Stemdata wordt niet door middel van digitale handtekeningen gecontroleerd			
Risico Classificatie	Gemiddeld risico	**Waarschijnlijkheid**	Laag
Categorie	Cryptografie	**Gevolgen**	Hoog

Betreft
OSV-2020-U

Omschrijving
De applicatie maakt gebruik van een controle-hash bij het invoeren van stemdata. Deze hashwaarde is de uitkomst van een cryptografisch sterke hashfunctie, waarbij het bestand met stemdata als invoer is gebruikt. De hashwaarde is bij de gebruiker die de data invoert bekend, en de applicatie berekent deze bij de invoer van een stemdata nogmaals. Nadat de gebruiker de stemdata heeft ingevoerd, vraagt de webapplicatie ter controle tweemaal vier waardes van deze hash. Hierbij ontbreekt echter de mogelijkheid om na te gaan door wie deze hashwaarde is gemaakt (authenticiteit), zowel voor de gebruiker als de applicatie zelf.

De applicatie heeft ook de mogelijkheid om ingezet te worden voor de Tweede Kamer (TK) verkiezingen. Bij nader overleg met de Kiesraad bleek dat de applicatie wel de mogelijkheid heeft tot digitale ondertekening (en verificatie daarvan) in deze opstelling. Om deze reden is hier kort en heel specifiek enige aandacht aan besteed.

De digitale ondertekening in de TK-opstelling werkt echter met losse (privé/publieke) sleutelparen, die alleen locatie- (en niet persoons-) gebonden zijn, zodat de bestanden digitaal ondertekend aangeleverd worden. Het digitaal ondertekende bestand wordt nu naar de ontvangende partij verzonden, via internet of op een USB-stick. Deze opstelling bevat geen technische mogelijkheid om de authenticiteit van de publieke sleutel zelf te verifiëren, wat effectief de voorgenomen authenticiteit van de ondertekende data tenietdoet.

Zie ook:
- `https://nl.wikipedia.org/wiki/Digitale_handtekening`
- `https://www.rijksoverheid.nl/onderwerpen/digitale-overheid/vraag-en-antwoord/wat-is-een-elektronische-handtekening`
- `https://medium.com/@zhaohuabing/digital-signature-and-public-key-as-identities-94f8ecac2a24`

Voor Informatie m.b.t. tot het transport van de het sleutel materiaal in de TK-opstelling, zie:

- `https://osv2020-u.local:20023/was-nl/resources/help/TK/aanmaken-private-key.htm`

Risico
Aangepaste stemdata kan momenteel enkel opgemerkt worden bij de invoer na controle van de hashwaarde.

Uit het eerder genoemde overleg kwam naar voren dat het aanvalsoppervlak bij de gemeenteraadsverkiezingen gering is, doordat er minder tot geen transport van data plaatsvindt in het geval de OSV-2020 software wordt ingezet voor Gemeenteraadsverkiezingen.

De TK-opstelling probeert de integriteit van stemdata te waarborgen door gebruik te maken van een vorm van digitale ondertekening. De opstelling bevat publiek sleutelmateriaal dat echter niet digitaal ondertekend is en daarmee niet op authenticiteit te controleren valt. Dit maakt het mogelijk dat de aanvaller kan pogen deze publieke sleutel zelf aan te leveren. Zo kan een aanvaller zich als een legitieme verzender pogen voor te doen en daarmee de ontvangende partij proberen over te halen deze malafide sleutel te laten invoeren. Hierna kan de aanvaller zijn eigen stemdata ondertekenen en daarna is het scenario gereduceerd tot hetzelfde als voordat de stemdata digitaal ondertekend was. Merk op dat dit beide in de praktijk vrij lastig kan zijn om uit te buiten en vandaar dat de waarschijnlijkheid dat zo'n aanval uitgevoerd wordt als laag ingeschat is.

Aanbeveling
Maak gebruik van digitale ondertekening van alle bestanden waarbij de integriteit en authenticiteit van de data belang is. Zorg er daarbij voor dat bijhorend certificaat van de Certificaat Autoriteit (CA) als enige is geïnstalleerd op zowel alle systemen waar deze applicatie draait, als in de browser (of onderliggende certificatenopslag) bij de gebruikers die de applicatie benaderen.

Daarnaast dient elke gebruiker gebruik te maken van een tot een persoon herleidbare sleutel (bijvoorbeeld door middel van een hardware token) of certificaat. Dit biedt de mogelijk (on)bewust foutief aangeleverde stemdata te traceren tot de persoon die dit ingevoerd heeft, of het systeem waarop dit ingevoerd is.

Een tweetal mogelijke invullingen hiervan zouden als volgt kunnen worden uitgevoerd:

Volledige offline, gebruikerscertificaten worden automatisch door een server van OSV-2020 gebruikt:

- Het installatiebestand is digitaal ondertekend (zie onderstaande bevinding "Installatiebestanden niet digitaal ondertekend").
- De Root CA wordt aangemaakt en beheerd op één systeem dat geen netwerktoegang heeft. De privésleutel van deze Root CA bevindt zich op dit systeem.
- Het Root CA certificaat (en daarmee ook de bijhorende publieke sleutel) is opgeslagen in de nu al gebruikte Java keystore, en wordt door bovengenoemd installatiebestand aangeleverd.
- Er wordt naast het installatiebestand een certificaatbestand aangeleverd met daarin het certificaat en bijhorende privésleutel per benodigde gebruiker, deze is persoonsgebonden en elke locatie zal daarom een ander bestand aangeleverd krijgen.
- Het certificaatbestand wordt door de lokale beheerder ingeladen via de applicatie, die deze in de Java keystore opslaat.
- De applicatie kan vervolgens automatisch stemdata versleutelen aan de hand van de ingelogde gebruiker.

- De ondertekende stemdata wordt samen met de bijhorende certificaten geëxporteerd voor vervoer naar de volgende locatie.
- De applicatie kan automatisch steminvoer controleren, doordat zowel de bijhorende ondertekende steminvoer als certificaten beschikbaar zijn. Daarnaast kan de geldigheid van deze bestanden gevalideerd worden, doordat overal hetzelfde Root CA certificaat beschikbaar is.
- Indien er een stemlocatie bij komt, kan deze simpelweg evenals hierboven genoemd een digitaal ondertekend installatiebestand en een voor deze nieuwe locatie bedoelde certificaten krijgen.

Initieel online, gebruikerscertificaten worden door de gebruiker zelf beheert en veilig opgeslagen op daarvoor bedoeld hardware token:

- Online:
  - De gebruiker krijgt een hardware token aangeleverd welke nog geen gebruikersdata bevat.
  - De Root CA wordt aangemaakt en beheerd op één systeem dat geen netwerktoegang heeft. De privésleutel van deze Root CA bevindt zich op dit systeem.
  - De gebruiker maakt met behulp van het hardware token een “Certifcate Signing Request” (CSR; ook wel een verzoek om certificaat te ondertekenen) aan. De bijbehorende privésleutel wordt op het hardware token opgeslagen.
  - Er wordt per gebruiker initieel gebruik gemaakt van een veilig online gebruikersportaal (alle data is versleuteld en alleen inzichtelijk voor de verzender en ontvanger), waarbij ingelogd wordt door middel van DigiD of eHerkenning.
  - De gebruiker stuurt de CSR-data via het bovengenoemde portaal, welke gebruikt wordt om een door de Root CA ondertekend persoonsgebonden certificaat te creëren, welke via hetzelfde portaal weer naar de gebruiker wordt verstuurd.
  - De gebruiker slaat het certificaat op, op dezelfde token als waar ook de privésleutel van de gebruiker is opgeslagen.

- Offline:
  - De applicatie vraagt vlak voor het onderteken van stemdata om de hardware token van de gebruiker te koppelen, zodat de applicatie lokaal het verzoek kan ondertekenen en vervolgens naar de server kan sturen. De applicatie verzoekt vervolgens om de hardware token te ontkoppelen.
  - De ondertekende stemdata wordt samen met de bijhorende certificaten geëxporteerd voor vervoer naar de volgende locatie (dit zou ook digitaal via het bovengenoemde gebruikersportaal verstuurd kunnen worden).
  - De applicatie kan automatisch steminvoer controleren, doordat zowel de bijhorende ondertekende steminvoer als certificaten beschikbaar zijn. Daarnaast kan de geldigheid van deze bestanden gevalideerd worden, doordat overal hetzelfde Root CA certificaat beschikbaar is.
  - Indien er een stemlocatie bij komt, kan deze simpelweg evenals hierboven genoemd een digitaal ondertekend installatiebestand en een voor deze nieuwe locatie bedoelde certificaten krijgen.

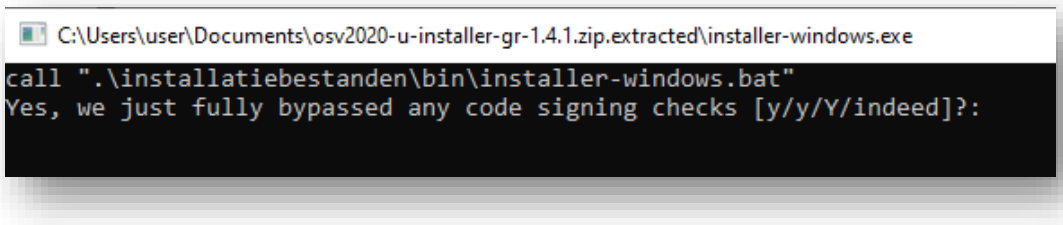
Bevinding 4: Digitale ondertekening van meerdere installatiebestanden ontbreekt			
Risico Classificatie	Gemiddeld risico	**Waarschijnlijkheid**	Laag
Categorie	Cryptografie	**Gevolgen**	Hoog

Betreft
OSV-2020-U

Omschrijving
Meerdere installatiebestanden zijn niet digitaal ondertekend. Om deze reden ontbreekt logischerwijs ook een controle van deze digitale ondertekening. Er wordt wel een controle-hashwaarde meegeleverd met de installatiebestanden, maar bij deze hashwaarde ontbreekt de mogelijkheid om na te gaan door wie deze hashwaarde is gemaakt (authenticiteit). Sommige installatiebestanden zijn wel digitaal ondertekend, maar doordat deze afhankelijk zijn van andere aangeleverde en niet digitaal ondertekende bestanden, is dit niet afdoende. De hoeveelheid en diepgang van ondertekende bestanden verschilt per besturingssysteem:

- Ubuntu Linux: geen enkel gebruikt installatiebestand is digitaal ondertekend;
- Windows 10: het initiële installatiebestand is ondertekend maar dit roept een ".bat"-bestand aan (een commando-script) dat niet digitaal ondertekend is en daarom niet vertrouwd kan worden;
- macOS: er wordt een macOS-applicatie aangeleverd. Deze is digitaal ondertekend met behulp van *Apple code signing*, maar kan niet uitgevoerd worden, omdat deze na het uitvoeren direct wordt afgesloten (geverifieerd op macOS Catalina 10.15.7). Mogelijk is het de bedoeling dat een in de applicatie aanwezige ".command"-bestand wordt uitgevoerd.

Fox-IT heeft als korte test het ".bat"-bestand aangepast en daarna het ondertekende Windows installatiebestand uitgevoerd, dat de commando's uit dit aangepaste bestand uitvoert, zoals te zien is in onderstaande schermafbeelding:



Figuur 1 - Uitvoer aangepast bestand door digitaal ondertekend installatiebestand

Risico
Indien één of meerdere van de installatiebestanden zijn aangepast door een aanvaller en daarbij ook de meegeleverde controle-hashwaarden zijn aangepast door de aanvaller, wordt dit niet opgemerkt. Dit maakt het bijvoorbeeld mogelijk om de applicatie malafide code uit te laten voeren, die bijvoorbeeld de daadwerkelijke stemdata in de achtergrond manipuleert. Indien dit vroegtijdig in het distributieproces gebeurt, zal overal waar deze aangepaste installatiebestanden gebruikt worden, de malafide applicatie geïnstalleerd worden, waardoor gefraudeerd zou kunnen worden.

Aanbeveling
Twee oplossingen kunnen worden gebruikt om te controleren of de software in het distributiekanaal niet is aangepast: een controle van een digitale ondertekening van het installatiebestand en een controle van een hashwaarde van het installatiebestand.

Controle van een digitale ondertekening van het installatiebestand
Om een zo consistent mogelijke uitrol van de aangeleverde bestanden te garanderen, is het aanbevolen de aangeleverde JAR-bestanden digitaal te ondertekenen met behulp van een CA. Een voorwaarde is dat de CA door alle ondersteunde besturingssystemen reeds vertrouwd wordt. Dit omdat er altijd ergens een begin moet zijn van een vertrouwd proces en het in dit geval al beschikbaar moet zijn voordat de installatiebestanden gebruikt worden. Dit mitigeert de behoefte om op elk besturingssysteem afhankelijk te zijn van een eigen manier van ondertekenen van bijvoorbeeld uitvoerbare bestanden of scripts.

De aangeleverde JAR-bestanden bevatten alle code die nodig is voor de installatie van de applicatie OSV-2020. Deze kan ondertekend worden met jarsigner, een Java-applicatie. Het is daarbij belangrijk dat alle JAR-bestanden welke aangeleverd worden door de leverancier, al door de leverancier zelf ondertekend zijn. Dit zorgt ervoor dat een JAR-bestand en alle daarin opgeslagen content ondertekend zijn met een publiek vertrouwd certificaat, door bijvoorbeeld een code signing certificaat bij Microsoft aan te vragen Bovendien geeft dit de mogelijkheid om de geldigheid van de JAR-bestanden te laten verlopen door de certificaten een beperkte tijd geldig te laten zijn. Dit maakt het mogelijk de bestanden van korte duur geldig te laten zijn, bijvoorbeeld alleen gedurende de maand van de verkiezingen.

Het ondertekenen van dergelijke bestanden kan gerealiseerd worden met onderstaande opdracht (let op: dit vereist de bijhorende Java JDK binaries):

```
jarsigner –keystore /pad/naar/osv_keystore –storepass “customstorepassword“
-keypass “customkeypassword” OSV2020_installatiebestand.jar OSV2020
```

Hier is OSV-2020 de alias in de lokale keystore, die opgeslagen is op locatie `/pad/naar/osv_keystore`. De gebruikte wachtwoorden dienen uiteraard vervangen te worden met een uniek sterk wachtwoord dat gebruikt is bij het versleutelen van bijhorende privésleutel. Deze manier van ondertekenen ondersteunt zowel het gebruik van DSA als RSA (SHA-256). Wanneer dit bestand is ondertekend kan het worden gebruikt op verschillende besturingssystemen, wat de uniformiteit van de wijze van ondertekening garandeert.

Vervolgens kan het aangeleverde bestand door de installerende partij worden gecontroleerd op het bevatten van een valide certificaat door middel van het uitvoeren van de volgende opdracht:

```
jarsigner -verify -verbose:summary OSV2020_installatiebestand.jar
```

Wanneer in de getoonde output het juiste certificaat te zien is waarmee het JAR-bestand is ondertekend, en er geen niet-ondertekende bestanden in dit bestand zitten, kan de installatie voortgezet worden. Alvorens het distribueren zal aan de installerende partijen kenbaar gemaakt moeten worden met welk certificaat het bestand ondertekend is, en zal de meest recente versie van Java geïnstalleerd moeten worden op de besturingssystemen waar de applicatie op geïnstalleerd wordt.

Voor de drie ondersteunde besturingssystemen is op het moment van schrijven de vereiste versie van Java te downloaden via `https://docs.oracle.com/en/java/javase/17/install/overview-jdk-installation.html`.

Controle van een digitale ondertekening van het installatiebestand
Voor het verifiëren van de integriteit en authenticiteit van de installatieprogramma's kan een hashwaarde worden gepubliceerd op een website van de overheid die gebruik maakt van TLS voor verbindingsbeveiliging. Door de hashwaarde van het ontvangen bestand te berekenen en te vergelijken met de hashwaarde die is gepubliceerd, kan worden gecontroleerd of het bestand in het distributiekanaal niet is aangepast.

Zie ook:
- `https://nl.wikipedia.org/wiki/Code_signing`
- `https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms537361(v=vs.85)`
- `https://docs.microsoft.com/en-us/windows-hardware/drivers/dashboard/get-a-code-signing-certificate`

Bevinding 5: Zwakke cryptografie in gebruik voor encryptie databasewachtwoord			
Risico Classificatie	Gemiddeld risico	**Waarschijnlijkheid**	Laag
Categorie	Configuratie	**Gevolgen**	Hoog

Betreft
OSV-2020-U

Omschrijving
Voor het versleutelen van het databasewachtwoord wordt gebruik gemaakt van encryptie door middel van Blowfish. Blowfish maakt gebruik van een block-size van 64 bits. De NIST raadt het gebruik van sterke encryptie aan door middel van standaard algoritmen, Blowfish behoort daar niet toe. Het is daarom aan te raden gebruik te maken van algoritmen zoals Twofish of AES. Beide kunnen gebruik maken van een block-size van 128 bits, waardoor ze beschouwd worden als een sterk algoritme.

Risico
Het gebruik van Blowfish maakt de applicatie kwetsbaar voor onder andere *Birthday attacks*. Deze aanval berust op een grotere waarschijnlijkheid van het voordoen van hash collisions wanneer meerdere gerandomiseerde aanvallen worden uitgevoerd. Merk op dat dit alleen een risico is wanneer een kwaadwillende gebruiker toegang heeft tot de server waar de webapplicatie op uitgevoerd wordt en deze het versleutelde databasewachtwoord kan achterhalen. Een voorbeeld hiervan is beschreven in bevinding 10.

Aanbeveling
Gebruik een sterk versleutelingsalgoritme zoals AES in XTS mode, zoals aanbevolen door de NIST. Zie: `https://csrc.nist.gov/publications/detail/sp/800-38e/final`

Bevinding 6: Gebruik van standaard sleutel voor encryptie databasewachtwoord			
Risico Classificatie	Gemiddeld risico	**Waarschijnlijkheid**	Laag
Categorie	Cryptografie	**Gevolgen**	Hoog

Betreft
OSV-2020-U

Omschrijving
In de onderstaande afbeeldingen is te zien dat het voor het versleutelen van het databasewachtwoord gebruikte Blowfish, gebruikt maakt van een standaard sleutel *"jaas is the way"*. Deze sleutel wordt sinds 2004 gebruikt als standaard sleutel in de JBoss module. Deze standaard string is ook online te vinden, zie `https://github.com/picketbox/picketbox/blob/master/security-jboss-sx/jbosssx/src/main/java/org/picketbox/datasource/security/SecureIdentityLoginModule.java`

```
private static String encode(String secret)
   throws NoSuchPaddingException, NoSuchAlgorithmException,
   InvalidKeyException, BadPaddingException, IllegalBlockSizeException
{
   byte[] kbytes ="jaas is the way".getBytes();
   SecretKeySpec key = new SecretKeySpec(kbytes,Blowfish);

   Cipher cipher = Cipher.getInstance("Blowfish");
   cipher.init(Cipher.ENCRYPT_MODE, key);
   byte[] encoding = cipher.doFinal(secret.getBytes());
   BigInteger n = new BigInteger(encoding);
   return n.toString(16);
}
```

Figuur 2 - Blowfish wordt gebruikt in combinatie met een bekende sleutel om wachtwoorden versleuteld op te slaan.

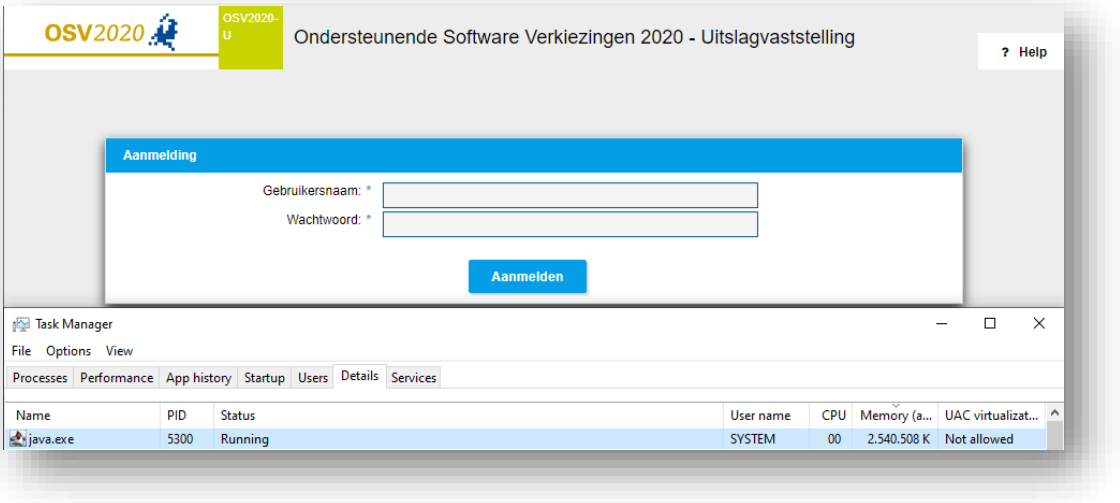
Risico
Vanwege het gebruik van makkelijk te raden cryptografische sleutels, vergroot dit de kans dat een aanvaller de originele wachtwoorden te achterhalen. Wanneer een aanvaller het databasewachtwoord kan achterhalen, geeft dit toegang tot de database waardoor de integriteit van de data in de database niet meer gewaarborgd is. Zoals in de risico-analyse van de voorgaande bevinding beschreven, vormt dit alleen een risico wanneer het versleutelde databasewachtwoord te achterhalen is door een kwaadwillende gebruiker.

Aanbeveling
Maak geen gebruik van een standaard sleutel, maar gebruik een sleutel die per installatie verschilt.

Bevinding 7: Least privilege principe niet geïmplementeerd bij installatie			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Configuratie	**Gevolgen**	Laag

Betreft
OSV-2020-U

Omschrijving
De applicatie draait op Linux niet onder een eigen account, maar met het account waarmee de computergebruiker inlogt en de applicatie installeert. Op Windows draait de software met SYSTEM-rechten, zoals te zien in onderstaande afbeelding.



Figuur 3 - De applicatie draait met SYSTEM-rechten op een Windows host.

Risico
Als de software kwetsbaarheden bevat die via het netwerk kunnen worden uitgebuit, kan mogelijk toegang worden verkregen tot het onderliggende systeem. Vervolgens kunnen systeemcommando's worden uitgevoerd die dan worden uitgevoerd met de rechten van het account waaronder de software draait. In het geval van de SYSTEM-gebruiker op Windows zouden dit volledige rechten zijn, omdat dit account de hoogst mogelijke rechten heeft op een Windows-systeem.
Wanneer geen apart account wordt gebruikt voor het uitvoeren van de software, is het ook niet goed mogelijk om de installatiebestanden of de bestanden die worden gebruikt door de software af te schermen van andere gebruikersaccounts. Mocht de applicatie in tegenstelling tot de gebruikelijke opstelling toch gevirtualiseerd zijn, vergroot het verkrijgen van SYSTEM rechten daarnaast ook de kans op uitbreken van deze of andere gevirtualiseerde systemen naar de onderliggende hypervisor, gevolgd door toegang tot alle gevirtualiseerde systemen op deze hypervisor.

Aanbeveling
Aanbevolen wordt tijdens de installatie een account aan te maken waarmee niet kan worden ingelogd op het systeem, maar waarmee alleen de applicatie kan worden opgestart (bijvoorbeeld met behulp van een service). Dit account moet zo min mogelijk rechten hebben.

Bevinding 8: Logregelinjectie mogelijk zonder authenticatie			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Gemiddeld
Categorie	Datavalidatie	**Gevolgen**	Laag

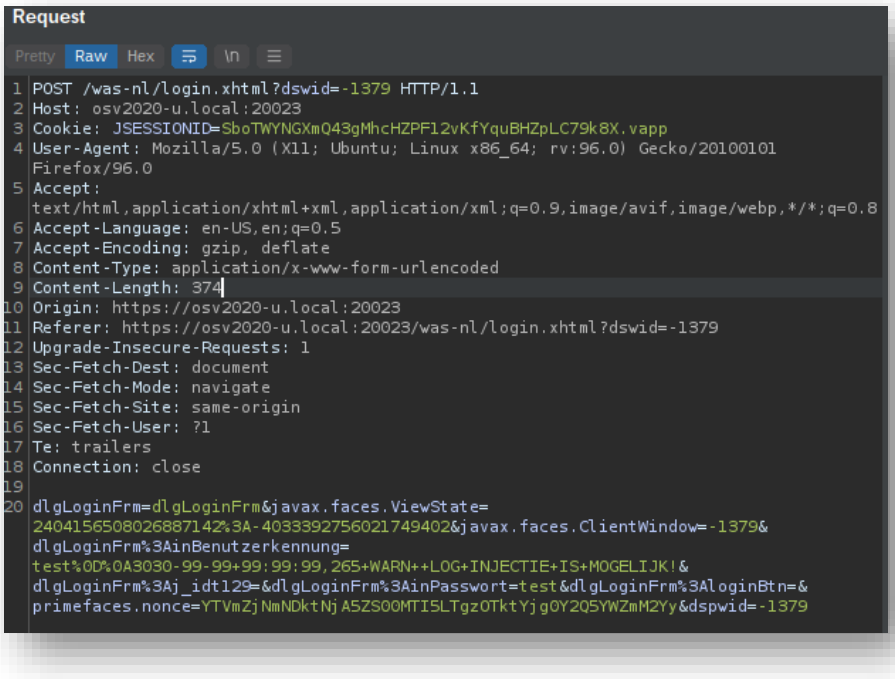
Betreft
OSV-2020-U

Omschrijving
De applicatie maakt gebruik van een logbestand dat als audit-trail en als debug log gebruikt kan worden. Dit logbestand bevat echter in sommige gevallen ook ongefilterde gebruikersinvoer. Zo kan een gebruiker, zonder enige vorm van authenticatie, via het inlogscherm de gebruikersnaam zo manipuleren dat een losse logregel gerapporteerd wordt.

Dit kan door de volgende karakters te gebruiken, en deze op te volgen met de te injecteren logregel:

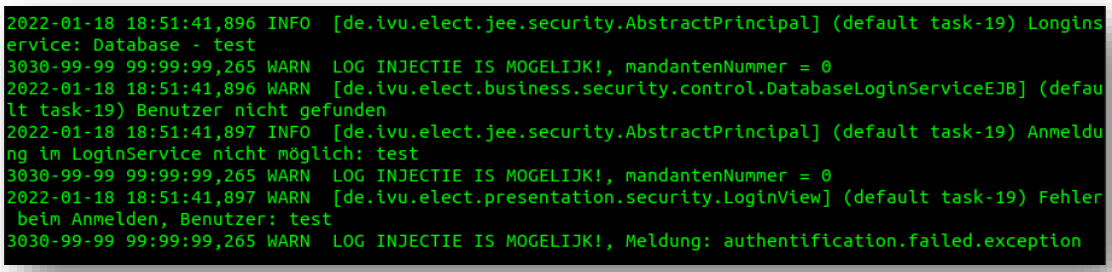
`'%0D%0A'`

Indien dit ingevuld wordt in de POST parameter genaamd `dlgLoginFrm%3AinBenutzerkennung` (zie ook onderstaande schermafbeelding), zal deze de invoer door de applicatie geaccepteerd worden en de karakters zoals eerder genoemd als *newline* geïnterpreteerd worden, waardoor de tekst die volgt als nieuwe logregel wordt opgeslagen.



Figuur 4 - Log injectie in POST request

In de volgende schermafbeelding is de geïnjecteerde logregel te zien, zoals deze in de het logbestand terecht komt:



Figuur 5 - Log bestand met geïnjecteerde logregel

Naast de control karakters voor een nieuwe regel, worden ook andere control karakters zoals “backspace” (URL gecodeerd `%08`) geïnterpreteerd, waardoor ook de statische tekst op de log regel zelf aangepast kan worden.

Merk op dat dit een ongeauthenticeerde aanval betreft.

Risico
Een aanvaller kan logregels aanpassen en extra logregels toevoegen en daarmee valse informatie plaatsten. Dit kan voor allerlei doeleinden gebruikt worden, zoals het incrimineren van een door de aanvaller gekozen slachtoffer of een poging een aanval te maskeren met allerlei afleidende log meldingen. Dit ondermijnt de integriteit van loggegevens en daarmee aan de controleerbaarheid van het stemproces en raakt dus aan een te beschermen belang.

Aanbeveling
Filter en valideer alle gebruikersinvoer, en maak hierbij waar mogelijk gebruik van een allowlist. Zorg er daarbij voor dat gebruikersinvoer altijd als statische tekst wordt geïnterpreteerd, om onbedoelde uitvoer van door een gebruiker opgegeven code te voorkomen. Encodeer vervolgens alle gebruikersinvoer, alvorens dit gebruikt wordt voor applicatie uitvoer. Ga na of de code van de applicatie op andere plaatsen identieke kwetsbaarheden bevat, en los deze op een generieke manier op.

Bevinding 9: Niet-ondersteunde software in gebruik			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Patching	**Gevolgen**	Gemiddeld

Betreft
OSV-2020-U

Omschrijving
De applicatie maakt gebruik van niet langer ondersteunde software waarin publiek bekende kwetsbaarheden aanwezig zijn. Deze bevinding heeft betrekking op ten minste de volgende software:

- Oracle Java SE version 1.11.0_1
- Thorntail 2.6.0.Final (WildFly Core 10.0.3.Final)

Het is binnen de beschikbaar gestelde tijd niet mogelijk geweest een uitputtend overzicht te creëren van de gebruikte softwareonderdelen en de bijbehorende kwetsbaarheden.

Risico
Nieuwe versies van software lossen bekende beveiligingsproblemen op. Wanneer niet de meest recente versie van de betrokken software wordt gebruikt, kunnen derhalve beveiligingsproblemen aanwezig zijn waar reeds een patch voor is uitgebracht.

In de in deze bevinding genoemde softwarepakketten zijn publiek bekende kwetsbaarheden aanwezig, die misbruikt kunnen worden om een Denial of Service (DoS) aanval te veroorzaken of vertrouwelijke informatie in te zien.

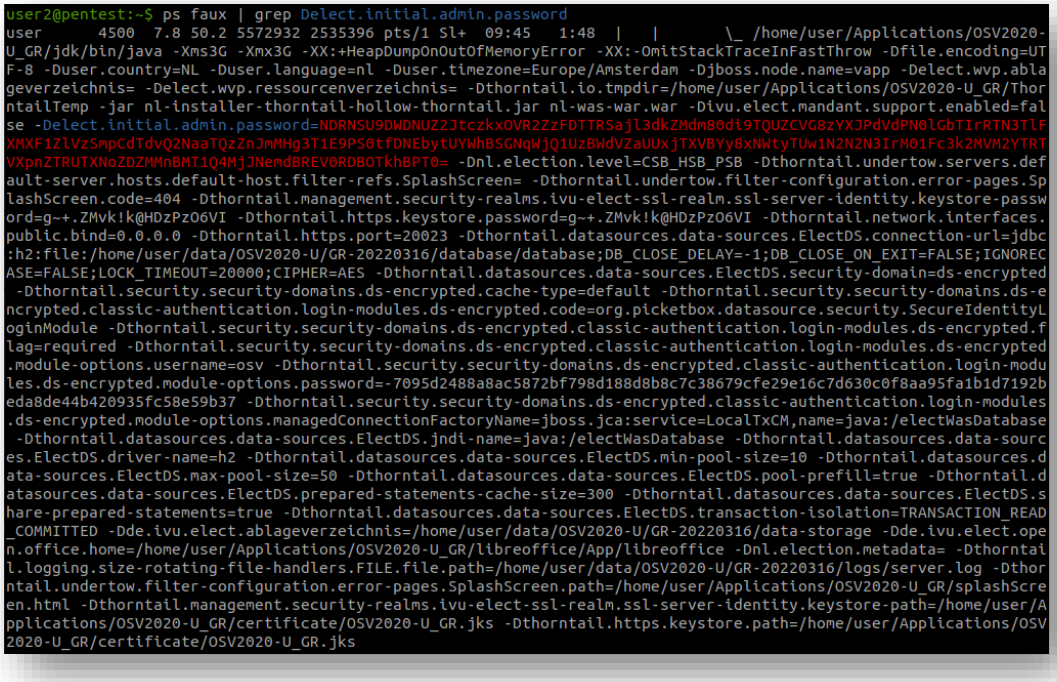
Voor kwetsbaarheden in niet-ondersteunde producten worden geen updates meer uitgebracht door de leveranciers. De gebruikte software blijft hierdoor kwetsbaar voor mogelijke aanvallers.

Aanbeveling
Spreek met de leverancier van de software af dat, en controleer of, de meest recente versie van software wordt meegeleverd. In het geval van Thorntail wordt aangeraden te onderzoeken of gebruik gemaakt kan worden van opvolgers daarvan: Wildfly en Quarkus.

Bevinding 10: Databasewachtwoord kan door elke gebruiker op de server worden achterhaald			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Configuratie	**Gevolgen**	Laag

Betreft
OSV-2020-U

Omschrijving
Omdat het versleutelde databasewachtwoord wordt meegeven als command line argument tijdens het opstarten van de applicatie, is het zichtbaar voor elke lokaal ingelogde gebruiker. Indien er een andere gebruiker op het systeem aanwezig is dan de gebruiker benodigd voor het uitvoeren van de OSV-applicatie, dan kan deze gebruiker het versleutelde databasewachtwoord inzien. Deze gebruiker hoeft zonder beheerrrechten enkel een lijst met draaiende processen op te vragen zonder om het versleutelde wachtwoord uit te kunnen lezen, zoals te zien is in onderstaande afbeelding.



Figuur 6 – Het is mogelijk het versleutelde databasewachtwoord in te zien door details over het proces van de applicatie op te vragen.

Risico
Het versleutelde databasewachtwoord kan mogelijk offline worden gekraakt. Ook kan met behulp van de logica in de Java-bestanden die op het systeem of elders te vinden zijn, het wachtwoord worden ontsleuteld. Vervolgens kan met het wachtwoord toegang verkregen worden tot de database en kan de integriteit van de gegevens in de database niet gegarandeerd blijven.

Aanbeveling
Geef het databasewachtwoord niet mee als command line argumentEen mogeljike oplossing is het opslaan van het wachtwoord in een key store, zoals beschreven in de aanbeveling van de volgende bevinding.

Bevinding 11: Niet alle wachtwoorden opgeslagen in een versleutelde key store			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Configuratie	**Gevolgen**	Laag

Betreft
OSV-2020-U

Omschrijving
Enkele gevoelige bestanden, waaronder de private key van het certificaat van de applicatie, zijn opgeslagen in de key store. Sommige wachtwoorden zijn echter opgeslagen in configuratiebestanden die door eenieder met toegang tot deze bestanden uit te lezen zijn. Deze bestanden zijn versleuteld, maar kunnen met voldoende tijd potentieel worden herleid naar het oorspronkelijke wachtwoord. Enkele voorbeelden van dergelijke bestanden zijn het databasewachtwoord dat versleuteld opgeslagen is onder Linux in het bestand `database-configuration.sh` en onder Windows onder `database-configuration.bat`.

```
SET DATABASE_TYPE=h2
SET "DATABASE_URL=jdbc:h2:file:C:\ProgramData\OSV2020-U\GR-20220316\database\database;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;IGNORECASE=FALSE;LOCK_TIMEOUT=20000;CIPHER=AES"
SET DATABASE_USER=osv
SET DATABASE_PASSWORD=-7095d2488a8ac5872bf798d188d8b8c74127577b2f742b21ab1f0995cd15934bd6d1325fb273d8f064265f4892564719
SET DATABASE_ENCRYPTION_KEY=
```

Figuur 7 - Het versleutelde wachtwoord voor de database in database-configuration.bat.

Daarnaast kan het versleutelde wachtwoord voor de key store in Linux en Windows worden uitgelezen in het bestand *keystore-configuration.sh* respectievelijk *keystore-configuration.bat*.

```
SET KEYSTORE_PATH=C:\Program Files\OSV2020-U_GR\certificate\OSV2020-U_GR.jks
SET KEYSTORE_PASSWORD=55539b431085b5c128f800b8af4e750706d629cb3f6d3dd2f9886421c5c0d731f0fe001c43192df3137d955f79c957936d029bc11aa8a64cadfee275b2e96d9d
```

Figuur 8 - Het versleutelde wachtwoord de voor key store in keystore-configuration.bat.

Risico
Met kennis van de betreffende versleutelde wachtwoorden, kan een aanvaller een poging doen het originele wachtwoord te achterhalen en zo de database of de key store te ontgrendelen. Daarnaast gaat het opslaan van wachtwoorden in losse tekstbestanden tegen de richtlijnen van OWASP in, zie: `https://cheatsheetseries.owasp.org/cheatsheets/Key_Management_Cheat_Sheet.html`

Doordat er meerdere manieren van encodering en versleuteling gebruikt worden om data veilig op te slaan, is de kans groter dat één van deze methoden een kwetsbaarheid bevat.

Aanbeveling
Reduceer het aanvalsoppervlak door gebruik te maken van één uniforme veilige manier van opslag van gevoelige data zoals wachtwoorden. Gebruik hiervoor bij voorkeur een standaardmethode zoals beschreven in de bron hierboven: sla wachtwoorden en andere gevoelige gegevens op in de key store en scherm het bijbehorende wachtwoord af. Een manier om het key store wachtwoord op te slaan is in een zogenaamd *stashfile*. Dit is een bestand dat de wachtwoorden bevat om toegang te krijgen tot de key store. Dergelijke stashfiles zijn alleen uit te lezen door een service account dat enkel bedoeld is voor de applicatie waar de gevoelige bestanden van in de key store staan.

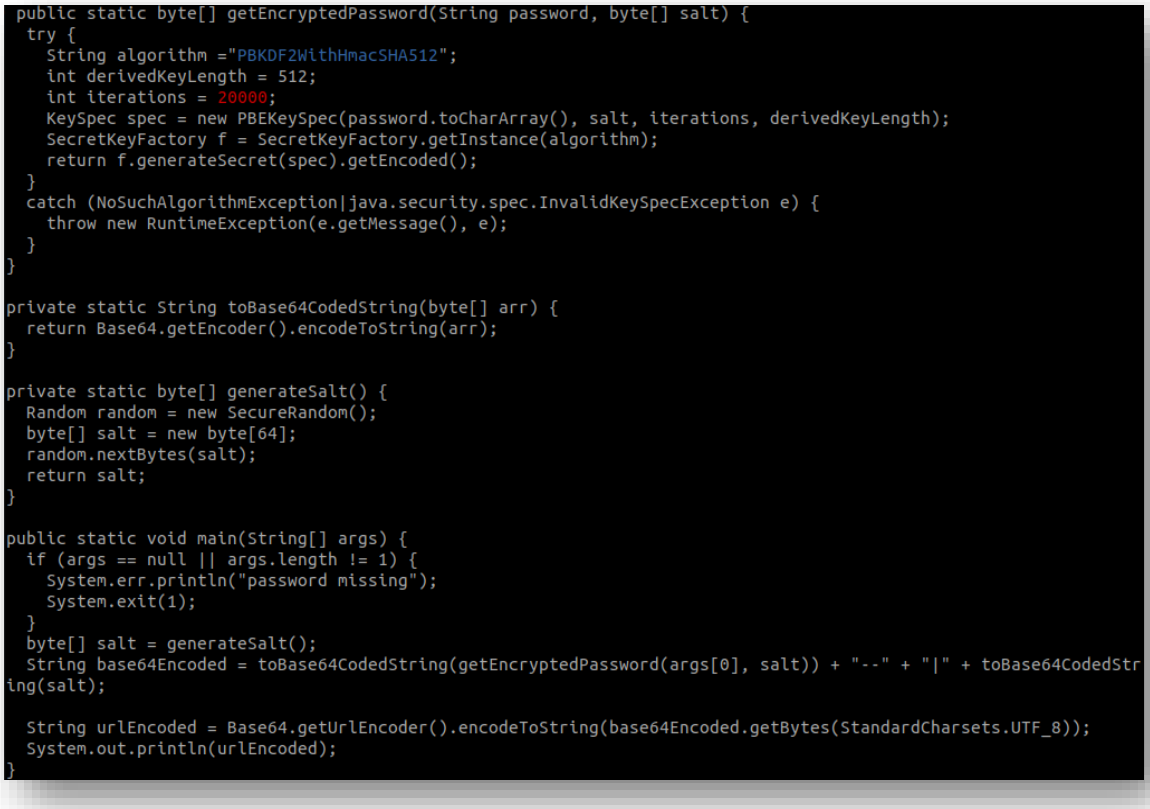
Bevinding 12: Niet genoeg PBKDF2-iteraties			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Cryptografie	**Gevolgen**	Laag

Betreft
OSV-2020-U

Omschrijving
Ondanks dat een broncodereview niet tot de scope van het onderzoek behoorde, is het mogelijk gebleken door middel van reverse engineering te achterhalen welke cryptografische implementatie gebruikt is voor het maken van sommige wachtwoordhashes. Voor het opslaan van gebruikerswachtwoorden wordt gebruik gemaakt van PBKDF2 voor het creëren van deze wachtwoordhashes. Deze gebruikerswachtwoorden worden gebruikt om in te loggen op het portaal van de applicatie, met admin als standaard gebruikersnaam. Tijdens het hashen van dergelijke wachtwoorden worden te weinig iteraties uitgevoerd om te voldoen aan de geldende norm van OWASP. Het gebruikte aantal iteraties is 20.000, maar aangeraden wordt minimaal 120.000 iteraties te gebruiken voor PBKDF2-HMAC-SHA512. Zie ook `https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#pbkdf2`

Aangezien het kraken lineair schaalt met het aantal afgedwongen iteraties, zal het per hashwaarde zes keer zo lang duren om deze te kraken wanneer de geldende norm van OWASP is doorgevoerd.

In de navolgende afbeelding is te zien welke code gebruik wordt voor het aanmaken van de hashes.



```
public static byte[] getEncryptedPassword(String password, byte[] salt) {
  try {
    String algorithm ="PBKDF2WithHmacSHA512";
    int derivedKeyLength = 512;
    int iterations = 20000;
    KeySpec spec = new PBEKeySpec(password.toCharArray(), salt, iterations, derivedKeyLength);
    SecretKeyFactory f = SecretKeyFactory.getInstance(algorithm);
    return f.generateSecret(spec).getEncoded();
  }
  catch (NoSuchAlgorithmException|java.security.spec.InvalidKeySpecException e) {
    throw new RuntimeException(e.getMessage(), e);
  }
}

private static String toBase64CodedString(byte[] arr) {
  return Base64.getEncoder().encodeToString(arr);
}

private static byte[] generateSalt() {
  Random random = new SecureRandom();
  byte[] salt = new byte[64];
  random.nextBytes(salt);
  return salt;
}

public static void main(String[] args) {
  if (args == null || args.length != 1) {
    System.err.println("password missing");
    System.exit(1);
  }
  byte[] salt = generateSalt();
  String base64Encoded = toBase64CodedString(getEncryptedPassword(args[0], salt)) + "--" + "|" + toBase64CodedStr
ing(salt);

  String urlEncoded = Base64.getUrlEncoder().encodeToString(base64Encoded.getBytes(StandardCharsets.UTF_8));
  System.out.println(urlEncoded);
}
```

Figuur 9 – Voor het berekenen van de hashes worden 20.000 iteraties gebruikt waar 120.000 aanbevolen is.

Risico

Een onvoldoende sterke implementatie van het eerdergenoemde algoritme vergroot de kans op het succesvol raden van het gebruikte wachtwoord. Dit wachtwoord in combinatie met de gekoppelde gebruikersnaam geeft toegang tot de interface van de applicatie waar vervolgens verkiezingsdata ingevoerd en uitgelezen kan worden.

Aanbeveling

Gebruik minimaal 120.000 PBKDF2-iteraties. Aangeraden wordt een recentere Key Derivation Function (KDF) als bijvoorbeeld Argon2 te gebruiken. Deze KDF is beschikbaar in de module Password4j van Java, en is ten opzichte van vergelijkbare KDF's sterk bestand tegen GPU aanvallen en side-channel attacks wanneer gekozen wordt voor Argon2id. Belangrijk is om de variabelen zorgvuldig in te stellen bij de implementatie van de KDF. Dit valt echter buiten de scope van de aanbeveling omdat er op algemene gronden te weinig kan worden gezegd over de juiste implementatie.

Bevinding 13: Het huidige wachtwoordbeleid is niet veilig genoeg			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Authenticatie	**Gevolgen**	Gemiddeld

Betreft
OSV-2020-U

Omschrijving
Het huidige wachtwoordbeleid onveilig om de volgende redenen:
- Er wordt een minimumlengte van slechts 9 karakters geforceerd;
- Er wordt geforceerd om speciale karakters in het wachtwoord te gebruiken.

Dit maakt het onder andere mogelijk om zwakke wachtwoorden zoals “Welkom01!” en “Lente2022!” te gebruiken.

Risico
Aanvallers kunnen proberen om wachtwoorden te raden. Indien er voorspelbare wachtwoorden gebruikt worden dan is de kans groot dat een aanvaller een wachtwoord weet te raden en daarmee toegang krijgt tot het desbetreffende account. Hierna kan de aanvaller de account gebruiken zoals de normale gebruiker dat ook kan.

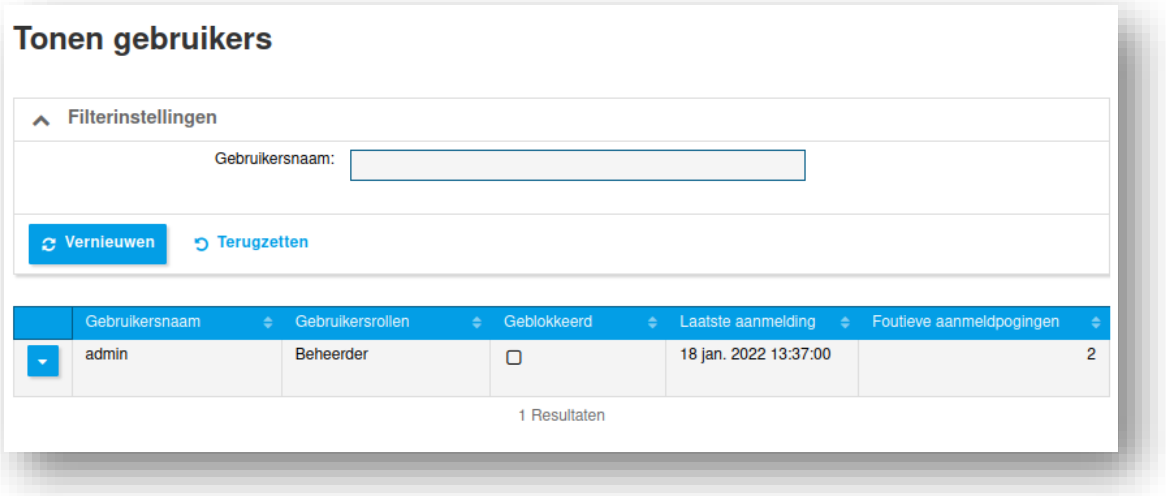
Aanbeveling
Gebaseerd op de richtlijnen van de NIST (https://pages.nist.gov/800-63-3/) heeft Fox-IT een lijst van aanbevelingen opgesteld om het wachtwoordbeleid te verbeteren en het gebruik van sterke wachtwoorden te stimuleren:
- Reduceer de lockout threshold tot 5 pogingen;
- Maak gebruik van wachtwoorden met een minimumlengte van 12 karakters;
- Vergroot de lockout observation window tot een onbeperkte tijd;
- Vereis menselijke goedkeuring voordat accounts worden vrijgegeven;
- Vereis geen speciale karakters in wachtwoorden;
- Vermijd de eis dat gebruikers het wachtwoord periodiek moeten veranderen;
- Stimuleer het gebruik van wachtwoordzinnen in plaats van wachtwoorden;
- Vergelijk wachtwoorden automatisch met een lijst met bekende slechte wachtwoorden.

We raden het af om wachtwoorden op een vast interval aan te passen, omdat onderzoek uitwijst dat gebruikers die gedwongen worden om hun wachtwoord periodiek aan te passen, uiteindelijk veelal gemakkelijk te raden wachtwoorden zullen gebruiken.

Bevinding 14: Melding van het aantal foutieve inlogpogingen ontbreekt			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Auditing en Logging	**Gevolgen**	Laag

Betreft
OSV-2020-U

Omschrijving
Binnen de applicatie houdt een teller bij hoeveel foutieve inlogpogingen verricht zijn voor elk afzonderlijk account. Wanneer de teller op zes staat wordt het bijbehorende account geblokkeerd. Een (ander) admin-account kan het account deblokkeren. De teller wordt echter weer op nul gezet nadat iemand succesvol is ingelogd met dit account. Wanneer één of meerdere foutieve pogingen zijn verricht om in te loggen met een gebruikersaccount op de applicatie, wordt de gebruiker daar niet van op de hoogte gesteld wanneer deze weer succesvol ingelogd is.



Figuur 10 - Voor het gebruikersaccount 'admin' zijn reeds twee foutieve inlogpogingen uitgevoerd.

Risico
Omdat de gebruiker niet op de hoogte wordt gesteld van mislukte inlogpogingen op het gebruikersaccount, kan een aanvaller meerdere foutieve inlogpogingen uitvoeren zonder dat de rechtmatige eigenaar dit door heeft. Dit vergroot de kans dat een aanvaller ongedetecteerd blijft tijdens het uitvoeren van bruteforce aanvallen op een of meerdere gebruikersaccounts. Het enige vereiste om onopgemerkt te blijven is dat de aanvaller voorkomt dat de teller van de foutieve inlogpogingen op zes (of hoger) komt te staan. Wanneer de rechtmatige eigenaar van het account opnieuw inlogt, zal de teller weer op nul worden gezet.

Aanbeveling
Laat een melding zien die de gebruiker attendeert op het feit dat één of meerdere foutieve inlogpogingen gedaan zijn, als dit het geval is.

Bevinding 15: Geen tweede factor benodigd bij inloggen			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Authenticatie	**Gevolgen**	Gemiddeld

Betreft
OSV-2020-U

Omschrijving
Om in te loggen op de web interface van de applicatie zijn een gebruikersnaam en wachtwoord benodigd. Indien zowel de gebruikersnaam als het bijhorende wachtwoord bekend zijn dan kan het betreffende account gebruikt worden door eenieder die bekend is met de combinatie en die tevens toegang hebben tot de OSV-omgeving.

Risico
Het ontbreken van de tweede factor betekent dat een achterhaalde combinatie van een gebruikersnaam en bijbehorend wachtwoord, een aanvaller in staat is om toegang te krijgen tot de applicatie.

De kans wordt vergroot als het gebruik van deze gegevens op een minder veilige manier gebeurt, zoals het gebruik van post-its met het wachtwoord, het gebruik van eenvoudig te raden wachtwoorden zoals “Welkom01!” of het delen van wachtwoorden onderling. Hierbij moet opgemerkt worden dat deze manier van wachtwoordgebruik buiten de scope van dit onderzoek valt, en dit enkel dient ter illustratie.

Aanbeveling
Om meer zekerheid te geven dat een gebruiker die inlogt ook daadwerkelijk de medewerker met OSV-autorisatie betreft, is het raadzaam om een tweede factor toe te voegen aan de authenticatie. Voor deze applicatie lijkt het toepasselijk om een tweede factor toe te voegen, zeker gezien het in het stemproces belangrijk is dat eventuele inbreuken op de integriteit van de telling van de stemmen traceerbaar, controleerbaar, verifieerbaar en corrigeerbaar zijn. Hierbij is het van belang dat deze tweede factor gebonden is aan de persoon en niet gedeeld wordt.

Doordat de tweede factor gebonden is aan de persoon is het significant moeilijker om alle benodigde informatie te verkrijgen die nodig is om in te loggen, wat indirect meer garantie geeft dat de uitgevoerde (gelogde) acties ook daadwerkelijk door de ingelogde persoon uitgevoerd worden.

Bevinding 16: Clientside air-gap controle ontbreekt			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Configuratie	**Gevolgen**	Laag

Betreft
OSV-2020-U

Omschrijving
De applicatie dient volgens de gebruiksvoorschriften op een air-gapped omgeving te draaien. Dit betekent dat de applicatie op een omgeving dient te draaien waar geen connectie is naar een extern netwerk, zoals het internet.

De applicatie test of er een air-gap aanwezig is. Deze controle wordt echter enkel op de server van de OSV-applicatie uitgevoerd. Indien de gebruiker op een ander systeem de applicatie benadert met behulp van de daarop geïnstalleerde webbrowser, is daar geen enkele air-gap controle aanwezig.

De Kiesraad heeft desgevraagd aangegeven dat al nagedacht is over een air-gap controle aan de gebruikerszijde, maar deze is nog niet in de code doorgevoerd.

Risico
Indien er op het systeem van de gebruiker geen air-gap controle aanwezig is, is het mogelijk dat dit systeem (per ongeluk) nog wel verbonden is met internet, wat het aanvalsoppervlak blootstelt aan nieuwe factoren. Indien een aanvaller toegang kan verkrijgen verkregen tot dit systeem, kan deze toegang misbruikt worden om het aantal van stemmen te beïnvloeden. Daarnaast kan de aanvaller deze internetkoppeling gebruiken om op afstand eventuele aanvallen op aanliggende systemen, zoals het systeem waar de OSV-applicatie op draait, voort te zetten.

Aanbeveling
Forceer ook het gebruik van een air-gap controle aan de gebruikerskant. Waarschuw de gebruiker indien dit niet het geval is en beperk de toegang tot de applicatie vanaf die client.

Bevinding 17: SSL/TLS-configuratie onvoldoende veilig			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Cryptografie	**Gevolgen**	Laag

Betreft
OSV-2020-U

Omschrijving
Geconstateerd is dat de SSL/TLS-configuratie kwetsbaarheden bevat die het voor aanvallers mogelijk maken om toegang te krijgen tot het versleutelde netwerkverkeer. Het betreft de volgende configuratieproblemen die niet voldoen aan de meest recente richtlijnen zoals opgesteld door het NCSC (2021):

- De server maakt gebruik van zwakke Diffie-Hellman-parameters;
- De server ondersteunt datacompressie voor het HTTPS-protocol;
- De server ondersteunt Client-Initiated Renegotiation.

De richtlijnen van het NCSC (2021) kunnen ingezien worden via de volgende URL:

`https://www.ncsc.nl/binaries/ncsc/documenten/publicaties/2021/januari/19/ict-beveiligingsrichtlijnen-voor-transport-layer-security-2.1/ICT-beveiligingsrichtlijnen+voor+Transport+Layer+Security+v2.1.pdf`

Risico
Als gevolg van de hierboven genoemde kwetsbaarheden is het onder specifieke omstandigheden voor een aanvaller mogelijk om (delen van) het verkeer tussen de client en de server in te zien en mogelijk te manipuleren.

Hoewel deze aanvallen geen hoge kans van slagen hebben en er eenvoudigere (maar beter detecteerbare) manieren bestaan om toegang tot versleuteld verkeer te krijgen (bijvoorbeeld door het uitvoeren van een Man-in-the-Middle-aanval), kan het ondersteunen van kwetsbare SSL/TLS-implementaties leiden tot reputatieschade. Dit is alleen een risico wanneer de server en de client niet op hetzelfde systeem draaien. Als de stemmen ingevoerd worden op hetzelfde systeem als waar de applicatieserver draait vormt dit een nihil risico.

Aanbeveling
Fox-IT raadt aan meer aandacht te besteden aan het veilig configureren van SSL/TLS.
Wanneer Diffie-Hellman wordt ondersteund voor sleuteluitwisseling raden wij aan unieke en sterke (groter dan 2048 bits) DH-parameters te genereren en te gebruiken.

Verder raden we aan het gebruik van TLS-compressie en ondersteuning voor Client-Initiated Renegotiation uit te schakelen in de configuratie van de server.

Tot slot is het belangrijk om serversoftware en de betrokken encryptiesoftware te updaten naar de meest recente versie en alle beschikbare beveiligingsupdates te installeren.

Aanvullende informatie met betrekking tot het veilig configureren van SSL/TLS is beschikbaar via `https://www.ssllabs.com/projects/best-practices/`

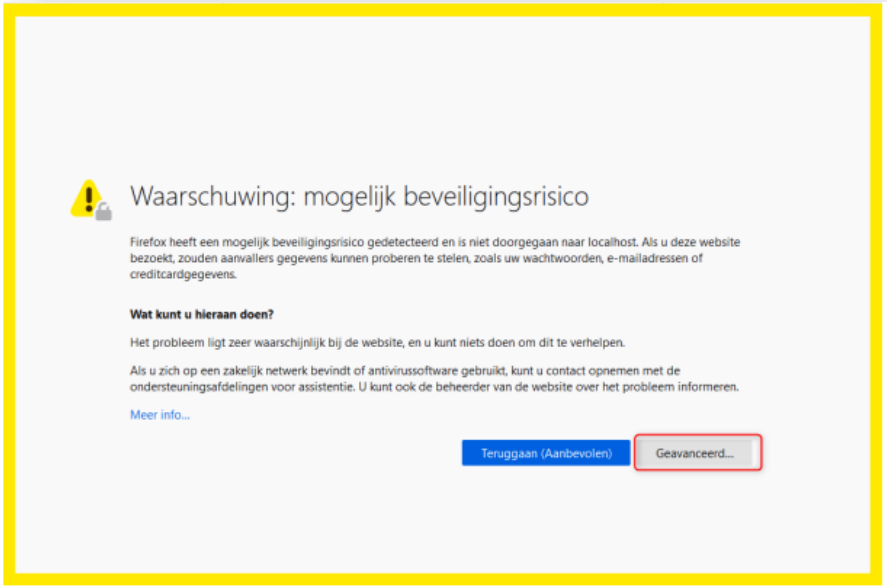
Bevinding 18: Gebruik van correct servercertificaat wordt niet afgedwongen			
Risico Classificatie	Laag risico	**Waarschijnlijkheid**	Laag
Categorie	Datavalidatie	**Gevolgen**	Gemiddeld

Betreft
OSV-2020-U

Omschrijving
Het gebruik van servercertificaten die ondertekend zijn door een vertrouwde CA (Certificaat Autoriteit) zorgt ervoor dat een gebruiker van een server zeker kan zijn dat deze met de eigenlijke server communiceert en daarmee de authenticiteit van de server kan bevestigen.

De server bevestigt momenteel al de authenticiteit van de gebruiker door een geldige gebruikersnaam en bijhorend wachtwoord te vereisen, alvorens interactie met de daadwerkelijke OSV-applicatie mogelijk is.

De gebruiker daarentegen kan momenteel niet het servercertificaat verifiëren, indien de OSV-applicatie op ander systeem is geïnstalleerd dan die van de gebruiker van deze applicatie, doordat deze niet is ondertekend door een vertrouwde CA. De gebruiker zal dan ook een waarschuwing krijgen van de browser, zoals ook is vermeld in de bijhorende handleiding.



Figuur 2.1 Waarschuwing Firefox browser bij gebruik 'self-signed certificate' voor de versleuteling van communicatie tussen browser en applicatie server.

Figuur 11 - Negeren van certificaat vanuit huidige gebruikershandleiding

Momenteel wordt de gebruiker verzocht om deze waarschuwing te negeren en daarmee ook het gebrek aan de mogelijkheid om de authenticiteit van de server te controleren te negeren (zie bovenstaande schermafbeelding). Dit gebruik is in strijd met OWASP ASVS richtlijnen en leert gebruikers om beveiligingswaarschuwingen te negeren. Volgens deze richtlijnen zou de host enkel specifieke interne CA's en self-signed certificaten moeten vertrouwen en alle andere moeten weigeren. Zie ook:

- handleiding_OSV2020-U-GR_v1.4.1.pdf
- https://owasp.org/www-project-application-security-verification-standard/

Risico

Indien een aanvaller zich in een netwerkpositie tussen de OSV-applicatieserver en gebruiker weet te manoeuvreren, kan deze met een malafide OSV-applicatieserver een Man-in-the-Middle (MitM) aanval uitvoeren. De waarschijnlijkheid dat dit in de standaard opstelling lukt is klein, maar bij afwijkende opstellingen is het wel degelijk een risico dat voorkomen behoort te worden.

Aanbeveling

Zorg er procedureel voor dat een vertrouwde beheerder het correcte servercertificaat, of bijhorende OSV Root CA certificaat (waarmee het servercertificaat getekend is), installeert. Dit betreft de installatie van dit certificaat in de internetbrowser van de gebruikers van OSV-applicatie. In veel internetbrowsers vereist dit enkel een handeling van een beheerder in deze internetbrowser.

Maak daarnaast gebruik van een zogenaamd HTTP Strict Transport Security (HSTS) *preload domain*. Dit wil zeggen een domein dat van tevoren geregistreerd is als HSTS-vereisend-domein en als zodanig meegeleverd wordt in internetbrowsers. Dit zorgt ervoor dat de internetbrowser afdwingt dat voor alle HTTP-verzoeken naar de betreffende server een HTTPS-verbinding benodigd is, zelfs zonder dat een gebruiker ooit met de betreffende server is verbonden.

Voor meer informatie met betrekking tot HSTS *preload domains*, zie:

- https://hstspreload.org/
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security#preloading_strict_transport_security

4 Richtlijnen

De Kiesraad heeft voor de uit te voeren opdracht bepaald dat de meest recente versies van de volgende richtlijnen gehanteerd dienen te worden:

- NCSC ICT-beveiligingsrichtlijnen voor Transport Layer Security (TLS);
- NCSC ICT-beveiligingsrichtlijnen voor webapplicaties;
- OWASP ASVS;
- OWASP Top 10.

De applicatie is getest volgens deze richtlijnen, hieruit volgen de bevindingen zoals weergegeven in de risicomatrix in paragraaf 3.1. Een toelichting op sommige afwijkingen binnen deze checklists zijn opgenomen in de volgende paragrafen.

4.1 TLS van NCSC

De richtlijnen van het Nationaal Cyber Security Centrum (NCSC) voor het Transport Layer Security-protocol (TLS) zijn bedoeld bij het beoordelen van configuraties van applicaties. Tijdens de test is gebruikt gemaakt van de meest recente publieke lijst. Deze lijst en aanvullende details zijn te vinden op `https://www.ncsc.nl/documenten/publicaties/2021/januari/19/ict-beveiligingsrichtlijnen-voor-transport-layer-security-2.1`.

Volgens deze richtlijnen voldoet de applicatie niet aan alle gestelde eisen. Zoals te zien is in Tabel 3 - Checklist van het NCSC ten behoeve van de ICT-beveiligingsrichtlijnen voor Transport Layer Security (TLS)., wijkt de applicatie af bij de implementatie van de algoritmeselecties (B2-5). De applicatie dwingt geen algoritmeselectie-ordening af terwijl niet alle algoritmeselecties als “Goed” kunnen worden bestempeld.

De vertrouwde elliptische curve Secp384r1 wordt aangeboden, maar ook ongeschikte curves als prime256v1, X25519 en X448, waardoor de applicatie strikt gezien niet voldoet aan de richtlijnen (B5-1). Desalniettemin is het risico verwaarloosbaar gegeven de context van de opstelling, zoals te lezen is in de eerder benoemde richtlijnen.

In de gebruikte versie van TLS is het mogelijk om een nieuwe handshake te initiëren, door gebruik te maken van deze mogelijkheid tot renegotiation is een Denial-of-Service mogelijk (B8-1).
Een aanvaller kan de mogelijkheid tot renegotiation veelvuldig inzetten zodat de applicatie uiteindelijk niet of moeilijker bereikbaar is voor legitieme gebruikers.

Nr	Maatregel	Status
B1-1	Alle ondersteunde TLS-versies zijn Goed, voldoende of Uit te faseren.	Voldoet
B2-1	Alle ondersteunde algoritmeselecties bevatten een Goed, Voldoende of Uit te faseren algoritme voor certificaatverificatie.	Voldoet
B2-2	Alle ondersteunde algoritmeselecties bevatten een Goed, Voldoende of Uit te faseren algoritme voor sleutelwisseling.	Voldoet
B2-3	Alle ondersteunde algoritmeselecties bevatten een Goed, Voldoende of Uit te faseren algoritme voor bulkversleuteling.	Voldoet
B2-4	Alle ondersteunde algoritmeselecties bevatten een Goed, Voldoende of Uit te faseren algoritme voor hashing.	Voldoet
B2-5	Alle ondersteunde algoritmeselecties zijn Goed, of de algoritmeselecties worden op basis van de voorgeschreven ordening door de servers gekozen.	Voldoet niet
B3-1	De server biedt een certificaat aan ter authenticatie.	Voldoet
B3-2	De ondertekende vingerafdruk is gecreëerd middels een Goed, Voldoende of Uit te faseren algoritme voor hashing.	Voldoet
B3-3	Als de server een certificaat aanbiedt met een RSA-sleutel, is de lengte van deze sleutel Goed of Voldoende.	Voldoet
B3-4	Als het aangeboden certificaat niet direct door de root CA is ondertekend, biedt de server tussenliggende CA's aan die het pad authenticeren tussen de root CA en het aangeboden certificaat.	Voldoet
B4-1	Wanneer er voor de sleuteluitwisseling gebruik wordt gemaakt van DHE, dan is de geheime parameter tijdelijk van aard, willekeurig gekozen uit een uniforme verdeling en van een adequate omvang voor het gekozen finite field.	Niet van toepassing
B4-2	Wanneer er voor de sleuteluitwisseling gebruik wordt gemaakt van ECDHE, dan is de geheime parameter tijdelijk van aard, willekeurig gekozen uit een uniforme verdeling en van een adequate omvang voor de gekozen elliptic curve.	Niet van toepassing
B5-1	Alle gebruikte elliptic curves zijn Goed, Voldoende of Uit te faseren.	Voldoet niet
B6-1	Alle gebruikte finite fields zijn Goed, Voldoende of Uit te faseren.	Uit te faseren
B7-1	De instellingen voor compressie zijn Goed, Voldoende of Uit te faseren.	Voldoet
B8-1	De instellingen voor renegotiation zijn Goed, Voldoende of Uit te faseren.	Voldoet niet
B9-1	De instellingen voor 0-RTT zijn Goed, Voldoende of Uit te faseren.	Niet van toepassing
B10-1	Voor alle ondersteunde configuraties die Uit te faseren zijn, worden voorwaarden vastgelegd voor de geplande beëindiging van het gebruik ervan.	Onbekend
B10-2	Alle Uit te faseren configuraties worden niet meer gebruikt nadat de termijn voor de geplande beëindiging is verstreken.	Onbekend

Tabel 3 - Checklist van het NCSC ten behoeve van de ICT-beveiligingsrichtlijnen voor Transport Layer Security (TLS).

4.2 Webapplicaties van NCSC

De richtlijnen van het Nationaal Cyber Security Centrum (NCSC) voor webapplicaties zijn bedoeld als leidraad voor het veiliger ontwikkelen van webapplicaties en bijbehorende infrastructuur. Tijdens de test is gebruikt gemaakt van de meest recente publieke lijst op het moment van schrijven. Deze lijst en aanvullende details zijn te vinden op `https://www.ncsc.nl/documenten/publicaties/2019/mei/01/ict-beveiligingsrichtlijnen-voor-webapplicaties`.

In onderstaande tabel zijn de uitslagen van deze test weergegeven. Hierbij zijn de volgende resultaatdefinities in gebruik:

- +: Geen afwijkingen gevonden binnen de beschikbaar gestelde tijd;
- x: De applicatie voldoet strikt genomen niet aan de gestelde eis, maar Fox-IT merkt dit niet aan als kwetsbaarheid.
- -: De eis is niet van toepassing, kon niet worden vastgesteld, of valt buiten de scope van het onderzoek.
- Y: De applicatie voldoet niet aan de gestelde eis, meer detail is opgenomen in de benoemde bevinding.

Belangrijk is om op te merken dat wanneer een maatregel een ‘+’ kreeg dit betekent dat er geen verwijzing is naar een bevinding.

In de tabel op de volgende pagina hebben enkele maatregelen de markering ‘x’ gekregen. Het is hierbij goed om uit te leggen wat de nuance is waarom deze tekortkomingen niet aangemerkt zijn als bevinding, terwijl het strikt gezien wel afwijkt van de gestelde normen. Zo zijn er regels met opmerkingen gevonden in de JavaScript code, waardoor de applicatie niet voldoet aan de gestelde norm (U/WA.06-01). Fox-IT schat de gevolgen van de kwetsbaarheid echter als verwaarloosbaar in, als gevolg waarvan dit niet als bevinding is opgenomen.

Daarnaast is een help-pagina beschikbaar zonder authenticatie (U/PW.02-02) op `https://osv2020-u.local:20023/was-nl/resources/help/GR/wachtwoord-policy.htm?zoom_highlightsub=policy`.

Dit onthult het wachtwoordbeleid, maar wanneer dit voldoet aan de richtlijnen zoals benoemd in bevinding 13, zou dit geen gevaar moeten vormen. Het openbaren van het wachtwoordbeleid zou geen kwetsbaarheid moeten introduceren wanneer wachtwoorden voldoende sterk gekozen zijn. Een dergelijke maatregel zou eerder kunnen leiden tot schijnveiligheid. Daarbij komt dat de OSV-2020 software door eenieder te installeren is, waardoor het wachtwoordbeleid ook zonder help-pagina kenbaar is.

Het is mogelijk verschillende HTTP-request methoden te gebruiken die niet noodzakelijk zijn voor het gebruik van de applicatie (U/PW.02-03). Zo wordt een OPTIONS HTTP-request afhanklijk van de URL, omgezet naar een GET of POST request, in plaats van dat deze resulteert in een bad request (error 400). Dit geldt ook voor PUT en DELETE HTTP-requests, PATCH en DEBUG leiden wel tot een bad request. In verschillende responses van de webapplicatie worden de HTTP-headers Server en X-Powered-By meegeven, zonder dat deze informatie bevatten (U/PW.02-04 en U/PW.02-05). Ten slotte worden er cookies geplaatst met de naam dsrwid-XXXX, waar XXXX een combinatie is van vier cijfers, die de secure en httpOnly missen (U/PW.03-03).

Echter lijken deze cookies niet van significant belang te zijn voor het functioneren van webapplicatie en is de impact van het missen van deze flags derhalve minimaal.

Nr	Maatregel	Status
B.01	alle	-
B.02	alle	-
B.03	alle	-
B.04	alle	-
B.05	alle	-
B.06	alle	-
U/TV.01	alle	-
U/WA.01	alle	-
U/WA.02	alle	-
U/WA.03	01 Valideer de invoer op de server	Bevinding 8
	02 Verbied of beperk het gebruik van dynamische file includes	+
	03 Converteer alle invoer naar een veilig formaat, waarbij risicovolle tekens uit de invoer 'onschadelijk' worden gemaakt	Bevinding 8
	04 Weiger foute, ongeldige of verboden invoer	+
U/WA.04	01 Converteer alle uitvoer naar een veilig formaat	Bevinding 8
U/WA.05	01 Pas, wanneer de webapplicatie persoonsgegevens verwerkt, de privacy-by-designprincipes toe	-
	02 Maak waar mogelijk gebruik van privacybevorderende technieken	-
	03 Versleutel of hash gevoelige gegevens in databases en bestanden	Bevinding 10
	04 Gebruik cryptografisch sterke sessie-identificerende cookies	-
	05 Versleutel communicatie	+
	06 Onderteken transacties met een digitale handtekening	Bevinding 3
U/WA.06	01 Verwijder commentaarregels uit de scripts (code)	x
	02 Verwijder of pseudonimiseer verwijzingen naar interne bestands- of systeemnamen	+
U/WA.07	01 Bouw commando- en queryteksten op met uitsluitend in de code reeds aanwezige vaste tekstfragmenten	-
	02 Geef gebruikersinvoer die gebruikt moet worden in commando's en queries op een zodanige manier door, dat de beoogde werking niet wordt gewijzigd	-
	03 Houd van elke webapplicatie bij welke functionaliteit van backendsystemen nodig is	-
	04 Verbied directe data-toegang tot backendsystemen, tenzij andere opties niet voorhanden zijn	+
U/WA.08	01 Maak bij het aanmelden een nieuwe sessie aan en verbreek een eventueel al bestaande sessie van die gebruiker. Maak de oude sessie-identifier ongeldig	Bevinding 2
	02 Beëindig de sessie na een vooraf vastgestelde en geconfigureerde tijdspanne van inactiviteit van de gebruiker (idle-time).	+
	03 Beëindig de sessie na een vooraf vastgestelde en geconfigureerde sessietijd (session-time).	Bevinding 2
	04 Bied de gebruiker de mogelijkheid de sessie op eigen initiatief te beëindigen (uitloggen).	+
	05 De sessie is na beëindiging niet langer geautoriseerd binnen de webapplicatie	+
U/WA.09	alle	-
U/PW.01	alle	-
U/PW.02	01 Behandel alleen http-requests waarvan de gegevens een correct type, lengte, formaat, tekens en patronen hebben.	+
	02 Behandel alleen http-requests van initiators met een correcte authenticatie en autorisatie.	x

	03 Sta alleen de voor de ondersteunde webapplicaties benodigde http-requestmethoden (GET, POST, etc.) toe en blokkeer de overige niet noodzakelijke http-requestmethoden.	x
	04 Verstuur alleen http-headers die voor het functioneren van http van belang zijn.	x
	05 Toon in http-headers alleen de hoogst noodzakelijke informatie die voor het functioneren van belang is.	x
	06 Bij het optreden van een fout wordt de informatie in een http-response tot een minimum beperkt. Een eventuele foutmelding zegt wel dat er iets is fout gegaan, maar niet hoe	+
U/PW.03	01 Beschrijf de parametrisering van de webserver in een configuratiedocument.	-
	02 Verbied het opvragen van de inhoud van het filesysteem van de server. Ondersteun geen directory-listings.	+
	03 Stel voor alle cookies de flags ‘secure’ en ‘HttpOnly’ in.	x
	04 Verstuur bij alle http-responses de http-headers ‘Content-Security-Policy: frame-ancestors’ en (tijdelijk) ‘X-Frame-Options’.	+
U/PW.04	alle	-
U/PW.05	01 Gebruik uitsluitend beveiligde (communicatie)protocollen voor de toegang tot beheermechanismen.	+
	02 Gebruik sterke authenticatie voor de toegang tot de beheermechanismen.	Bevinding 15
U/PW.06	alle	-
U/PW.07	alle	-
U/PW.08	alle	-
U/NW.01	alle	-
U/NW.02	alle	-
U/NW.03	alle	-
U/NW.04	alle	-
U/NW.05	alle	-
U/NW.06	alle	-
U/NW.07	alle	-
U/NW.08	alle	-
C.01	alle	-
C.02	alle	-
C.03	alle	-
C.04	alle	-
C.05	alle	-
C.06	alle	-
C.07	alle	-
C.08	alle	-
C.09	alle	-
C.10	alle	-
C.11	alle	-

Tabel 4 - Checklist van het NCSC ten behoeve van de ICT-beveiligingsrichtlijnen voor webapplicaties.

4.3 ASVS van OWASP

De Application Security Verification Standard (ASVS), gedefinieerd door Open Web Application Security Project (OWASP), biedt een basis voor het testen van de technische beveiligingscontroles van webapplicaties.

Ook in onderstaande tabel wordt gebruik gemaakt van de definities met betrekking tot het resultaat zoals genoemd in paragraaf 4.2.

Het is mogelijk gebleken wachtwoorden in te stellen voor gebruikersaccounts met meer dan 128 karakters (V2.1.2). Daarnaast leek geen wachtwoord sterktemeter aanwezig te zijn (V2.1.8) en leek het niet mogelijk te zijn het getypte wachtwoord te bekijken (V2.1.12). Dit gaat tegen de richtlijnen van ASVS in, maar lijkt geen significante kwetsbaarheid te introduceren binnen de webapplicatie.

Dat geldt ook voor het ontbreken van meldingen wanneer een e-mailadres gewijzigd is dat aan een account gekoppeld is of dat er ingelogd wordt vanaf een onvertrouwde locatie (V2.2.3). Echter wordt dit ondervangen door de omgeving waarbinnen de OSV-2020-applicatie wordt gebruikt.

Cookies maken geen gebruik van SameSite attribute (V3.4.3) noch van de "__HOST-" (V3.4.4) prefix, daarnaast mist de X-Content-Type-Options: nosniff header (V14.4.4), maar het lijkt slechts zeer beperkt mogelijk te zijn om hier als aanvaller misbruik van te maken

Tevens is het mogelijk extra data mee te geven in POST requests (V5.1.3) wat in te zien is in de reponse. Binnen de tijd die beschikbaar was voor dit onderzoek is echter geen XSS, XXE of ander gerelateerde kwetsbaarheid geïdentificeerd.

Er is een open-redirect gevonden (V5.1.5). Dit is uit te voeren door de host header aan te passen naar de redirect-host in een HTTP-verzoek naar `https://osv2020-u.local:20023/was-nl/login.xhtml?dswid=-6412`. Echter is het als aanvaller onmogelijk de HOST-header op afstand aan te passen zonder aanvullende kwetsbaarheden te gebruiken. Dit maakt het scenario onwaarschijnlijk en daarom is dit niet opgenomen als bevinding.

Het is gebleken dat er gebruikt wordt gemaakt van een 'eval' statement (V5.2.4) in de response van sommige XML-data, maar binnen de beschikbare tijd is er geen gerelateerde kwetsbaarheid gevonden.

Het is voor eindgebruikers niet mogelijk om de eigen data te verwijderen of te exporteren (V8.3.2). Het gaat hier echter om algemene data en niet om PII, wat niet leidt tot een significante kwetsbaarheid.

Het is mogelijk gebleken gevoelige informatie van de applicatie uit het geheugen te lezen nadat de applicatie gesloten was (V8.3.6). Hiervoor is een memory dump gemaakt, vervolgens is deze handmatig geanalyseerd en bleken sporen aanwezig te zijn van de database en het bijbehorende wachtwoord. Wanneer een aanvaller echter de mogelijkheid heeft om een memory dump te maken, zijn er legio andere aanvalsvectoren te vinden.

Nr	Omschrijving	Status
V1.1.1	Verify the use of a secure software development lifecycle that addresses security in all stages of development.	-
V1.1.2	Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing.	-
V1.1.3	Verify that all user stories and features contain functional security constraints, such as "As a user, I should be able to view and edit my profile. I should not be able to view or edit anyone else's profile"	-
V1.1.4	Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.	-
V1.1.5	Verify definition and security analysis of the application's high-level architecture and all connected remote services.	-
V1.1.6	Verify implementation of centralized, simple (economy of design), vetted, secure, and reusable security controls to avoid duplicate, missing, ineffective, or insecure controls.	-
V1.1.7	Verify availability of a secure coding checklist, security requirements, guideline, or policy to all developers and testers.	-
V1.2.1	Verify the use of unique or special low-privilege operating system accounts for all application components, services, and servers.	Bevinding 7
V1.2.2	Verify that communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed.	+
V1.2.3	Verify that the application uses a single vetted authentication mechanism that is known to be secure, can be extended to include strong authentication, and has sufficient logging and monitoring to detect account abuse or breaches.	Bevinding 15, 2, 8
V1.2.4	Verify that all authentication pathways and identity management APIs implement consistent authentication security control strength, such that there are no weaker alternatives per the risk of the application.	+
V1.4.1	Verify that trusted enforcement points, such as access control gateways, servers, and serverless functions, enforce access controls. Never enforce access controls on the client.	+
V1.4.2	[DELETED, NOT ACTIONABLE]	-
V1.4.3	[DELETED, DUPLICATE OF 4.1.3]	-
V1.4.4	Verify the application uses a single and well-vetted access control mechanism for accessing protected data and resources. All requests must pass through this single mechanism to avoid copy and paste or insecure alternative paths.	-
V1.4.5	Verify that attribute or feature-based access control is used whereby the code checks the user's authorization for a feature/data item rather than just their role. Permissions should still be allocated using roles.	-
V1.5.1	Verify that input and output requirements clearly define how to handle and process data based on type, content, and applicable laws, regulations, and other policy compliance.	-
V1.5.2	Verify that serialization is not used when communicating with untrusted clients. If this is not possible, ensure that adequate integrity controls (and possibly encryption if sensitive data is sent) are enforced to prevent deserialization attacks including object injection.	+
V1.5.3	Verify that input validation is enforced on a trusted service layer.	-
V1.5.4	Verify that output encoding occurs close to or by the interpreter for which it is intended.	-
V1.6.1	Verify that there is an explicit policy for management of cryptographic keys and that a cryptographic key lifecycle follows a key management standard such as NIST SP 800-57.	-
V1.6.2	Verify that consumers of cryptographic services protect key material and other secrets by using key vaults or API based alternatives.	-

V1.6.3	Verify that all keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data.	-
V1.6.4	Verify that the architecture treats client-side secrets--such as symmetric keys, passwords, or API tokens--as insecure and never uses them to protect or access sensitive data.	-
V1.7.1	Verify that a common logging format and approach is used across the system.	x
V1.7.2	Verify that logs are securely transmitted to a preferably remote system for analysis, detection, alerting, and escalation.	x
V1.8.1	Verify that all sensitive data is identified and classified into protection levels.	-
V1.8.2	Verify that all protection levels have an associated set of protection requirements, such as encryption requirements, integrity requirements, retention, privacy and other confidentiality requirements, and that these are applied in the architecture.	-
V1.9.1	Verify the application encrypts communications between components, particularly when these components are in different containers, systems, sites, or cloud providers.	-
V1.9.2	Verify that application components verify the authenticity of each side in a communication link to prevent person-in-the-middle attacks. For example, application components should validate TLS certificates and chains.	Java app and (h2) backend database do not use certificate based mutual authentication
V1.10.1	Verify that a source code control system is in use, with procedures to ensure that check-ins are accompanied by issues or change tickets. The source code control system should have access control and identifiable users to allow traceability of any changes.	-
V1.11.1	Verify the definition and documentation of all application components in terms of the business or security functions they provide.	-
V1.11.2	Verify that all high-value business logic flows, including authentication, session management and access control, do not share unsynchronized state.	-
V1.11.3	Verify that all high-value business logic flows, including authentication, session management and access control are thread safe and resistant to time-of-check and time-of-use race conditions.	-
V1.12.1	[DELETED, DUPLICATE OF 12.4.1]	-
V1.12.2	Verify that user-uploaded files - if required to be displayed or downloaded from the application - are served by either octet stream downloads, or from an unrelated domain, such as a cloud file storage bucket. Implement a suitable Content Security Policy (CSP) to reduce the risk from XSS vectors or other attacks from the uploaded file.	+
V1.14.1	Verify the segregation of components of differing trust levels through well-defined security controls, firewall rules, API gateways, reverse proxies, cloud-based security groups, or similar mechanisms.	-
V1.14.2	Verify that binary signatures, trusted connections, and verified endpoints are used to deploy binaries to remote devices.	Bevinding 4
V1.14.3	Verify that the build pipeline warns of out-of-date or insecure components and takes appropriate actions.	Bevinding 9
V1.14.4	Verify that the build pipeline contains a build step to automatically build and verify the secure deployment of the application, particularly if the application infrastructure is software defined, such as cloud environment build scripts.	-
V1.14.5	Verify that application deployments adequately sandbox, containerize and/or isolate at the network level to delay and deter attackers from attacking other applications, especially when they are performing sensitive or dangerous actions such as deserialization.	-

V1.14.6	Verify the application does not use unsupported, insecure, or deprecated client-side technologies such as NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL, or client-side Java applets.	+
V2.1.1	Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined).	Bevinding 13
V2.1.2	Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied.	x
V2.1.3	Verify that password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space.	+
V2.1.4	Verify that any printable Unicode character, including language neutral characters such as spaces and Emojis are permitted in passwords.	+
V2.1.5	Verify users can change their password.	+
V2.1.6	Verify that password change functionality requires the user's current and new password.	+
V2.1.7	Verify that passwords submitted during account registration, login, and password change are checked against a set of breached passwords either locally (such as the top 1,000 or 10,000 most common passwords which match the system's password policy) or using an external API. If using an API a zero knowledge proof or other mechanism should be used to ensure that the plain text password is not sent or used in verifying the breach status of the password. If the password is breached, the application must require the user to set a new non-breached password.	Bevinding 13
V2.1.8	Verify that a password strength meter is provided to help users set a stronger password.	x
V2.1.9	Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters.	Bevinding 13
V2.1.10	Verify that there are no periodic credential rotation or password history requirements.	+
V2.1.11	Verify that "paste" functionality, browser password helpers, and external password managers are permitted.	+
V2.1.12	Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as built-in functionality.	x
V2.2.1	Verify that anti-automation controls are effective at mitigating breached credential testing, brute force, and account lockout attacks. Such controls include blocking the most common breached passwords, soft lockouts, rate limiting, CAPTCHA, ever increasing delays between attempts, IP address restrictions, or risk-based restrictions such as location, first login on a device, recent attempts to unlock the account, or similar. Verify that no more than 100 failed attempts per hour is possible on a single account.	+
V2.2.2	Verify that the use of weak authenticators (such as SMS and email) is limited to secondary verification and transaction approval and not as a replacement for more secure authentication methods. Verify that stronger methods are offered before weak methods, users are aware of the risks, or that proper measures are in place to limit the risks of account compromise.	-
V2.2.3	Verify that secure notifications are sent to users after updates to authentication details, such as credential resets, email or address changes, logging in from unknown or risky locations. The use of push notifications - rather than SMS or email - is preferred, but in the absence of push notifications, SMS or email is acceptable as long as no sensitive information is disclosed in the notification.	x
V2.2.4	Verify impersonation resistance against phishing, such as the use of multi-factor authentication, cryptographic devices with intent (such as connected keys with a push to authenticate), or at higher AAL levels, client-side certificates.	Bevinding 15
V2.2.5	Verify that where a Credential Service Provider (CSP) and the application verifying authentication are separated, mutually authenticated TLS is in place between the two endpoints.	-

V2.2.6	Verify replay resistance through the mandated use of One-time Passwords (OTP) devices, cryptographic authenticators, or lookup codes.	-
V2.2.7	Verify intent to authenticate by requiring the entry of an OTP token or user-initiated action such as a button press on a FIDO hardware key.	-
V2.3.1	Verify system generated initial passwords or activation codes SHOULD be securely randomly generated, SHOULD be at least 6 characters long, and MAY contain letters and numbers, and expire after a short period of time. These initial secrets must not be permitted to become the long term password.	+
V2.3.2	Verify that enrollment and use of user-provided authentication devices are supported, such as a U2F or FIDO tokens.	Bevinding 15
V2.3.3	Verify that renewal instructions are sent with sufficient time to renew time bound authenticators.	-
V2.4.1	Verify that passwords are stored in a form that is resistant to offline attacks. Passwords SHALL be salted and hashed using an approved one-way key derivation or password hashing function. Key derivation and password hashing functions take a password, a salt, and a cost factor as inputs when generating a password hash.	+
V2.4.2	Verify that the salt is at least 32 bits in length and be chosen arbitrarily to minimize salt value collisions among stored hashes. For each credential, a unique salt value and the resulting hash SHALL be stored.	+
V2.4.3	Verify that if PBKDF2 is used, the iteration count SHOULD be as large as verification server performance will allow, typically at least 100,000 iterations.	Bevinding 12
V2.4.4	Verify that if bcrypt is used, the work factor SHOULD be as large as verification server performance will allow, with a minimum of 10.	-
V2.4.5	Verify that an additional iteration of a key derivation function is performed, using a salt value that is secret and known only to the verifier. Generate the salt value using an approved random bit generator [SP 800-90Ar1] and provide at least the minimum security strength specified in the latest revision of SP 800-131A. The secret salt value SHALL be stored separately from the hashed passwords (e.g., in a specialized device like a hardware security module).	Bevinding 5
V2.5.1	Verify that a system generated initial activation or recovery secret is not sent in clear text to the user.	-
V2.5.2	Verify password hints or knowledge-based authentication (so-called "secret questions") are not present.	+
V2.5.3	Verify password credential recovery does not reveal the current password in any way.	-
V2.5.4	Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").	-
V2.5.5	Verify that if an authentication factor is changed or replaced, that the user is notified of this event.	-
V2.5.6	Verify forgotten password, and other recovery paths use a secure recovery mechanism, such as time-based OTP (TOTP) or other soft token, mobile push, or another offline recovery mechanism.	-
V2.5.7	Verify that if OTP or multi-factor authentication factors are lost, that evidence of identity proofing is performed at the same level as during enrollment.	-
V2.6.1	Verify that lookup secrets can be used only once.	-
V2.6.2	Verify that lookup secrets have sufficient randomness (112 bits of entropy), or if less than 112 bits of entropy, salted with a unique and random 32-bit salt and hashed with an approved one-way hash.	-
V2.6.3	Verify that lookup secrets are resistant to offline attacks, such as predictable values.	-
V2.7.1	Verify that clear text out of band (NIST "restricted") authenticators, such as SMS or PSTN, are not offered by default, and stronger alternatives such as push notifications are offered first.	-
V2.7.2	Verify that the out of band verifier expires out of band authentication requests, codes, or tokens after 10 minutes.	-

V2.7.3	Verify that the out of band verifier authentication requests, codes, or tokens are only usable once, and only for the original authentication request.	-
V2.7.4	Verify that the out of band authenticator and verifier communicates over a secure independent channel.	-
V2.7.5	Verify that the out of band verifier retains only a hashed version of the authentication code.	-
V2.7.6	Verify that the initial authentication code is generated by a secure random number generator, containing at least 20 bits of entropy (typically a six digital random number is sufficient).	-
V2.8.1	Verify that time-based OTPs have a defined lifetime before expiring.	-
V2.8.2	Verify that symmetric keys used to verify submitted OTPs are highly protected, such as by using a hardware security module or secure operating system based key storage.	-
V2.8.3	Verify that approved cryptographic algorithms are used in the generation, seeding, and verification of OTPs.	-
V2.8.4	Verify that time-based OTP can be used only once within the validity period.	-
V2.8.5	Verify that if a time-based multi-factor OTP token is re-used during the validity period, it is logged and rejected with secure notifications being sent to the holder of the device.	-
V2.8.6	Verify physical single-factor OTP generator can be revoked in case of theft or other loss. Ensure that revocation is immediately effective across logged in sessions, regardless of location.	-
V2.8.7	Verify that biometric authenticators are limited to use only as secondary factors in conjunction with either something you have and something you know.	-
V2.9.1	Verify that cryptographic keys used in verification are stored securely and protected against disclosure, such as using a Trusted Platform Module (TPM) or Hardware Security Module (HSM), or an OS service that can use this secure storage.	-
V2.9.2	Verify that the challenge nonce is at least 64 bits in length, and statistically unique or unique over the lifetime of the cryptographic device.	-
V2.9.3	Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.	-
V2.10.1	Verify that intra-service secrets do not rely on unchanging credentials such as passwords, API keys or shared accounts with privileged access.	-
V2.10.2	Verify that if passwords are required for service authentication, the service account used is not a default credential. (e.g. root/root or admin/admin are default in some services during installation).	+
V2.10.3	Verify that passwords are stored with sufficient protection to prevent offline recovery attacks, including local system access.	Bevinding 6
V2.10.4	Verify passwords, integrations with databases and third-party systems, seeds and internal secrets, and API keys are managed securely and not included in the source code or stored within source code repositories. Such storage SHOULD resist offline attacks. The use of a secure software key store (L1), hardware TPM, or an HSM (L3) is recommended for password storage.	-
V3.1.1	Verify the application never reveals session tokens in URL parameters.	+
V3.2.1	Verify the application generates a new session token on user authentication.	+
V3.2.2	Verify that session tokens possess at least 64 bits of entropy.	+
V3.2.3	Verify the application only stores session tokens in the browser using secure methods such as appropriately secured cookies (see section 3.4) or HTML 5 session storage.	+
V3.2.4	Verify that session tokens are generated using approved cryptographic algorithms.	-
V3.3.1	Verify that logout and expiration invalidate the session token, such that the back button or a downstream relying party does not resume an authenticated session, including across relying parties.	+
V3.3.2	If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period.	-

V3.3.3	Verify that the application gives the option to terminate all other active sessions after a successful password change (including change via password reset/recovery), and that this is effective across the application, federated login (if present), and any relying parties.	Bevinding 1
V3.3.4	Verify that users are able to view and (having re-entered login credentials) log out of any or all currently active sessions and devices.	Bevinding 2
V3.4.1	Verify that cookie-based session tokens have the 'Secure' attribute set.	+
V3.4.2	Verify that cookie-based session tokens have the 'HttpOnly' attribute set.	+
V3.4.3	Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks.	x
V3.4.4	Verify that cookie-based session tokens use the "__Host-" prefix so cookies are only sent to the host that initially set the cookie.	x
V3.4.5	Verify that if the application is published under a domain name with other applications that set or use session cookies that might disclose the session cookies, set the path attribute in cookie-based session tokens using the most precise path possible.	-
V3.5.1	Verify the application allows users to revoke OAuth tokens that form trust relationships with linked applications.	-
V3.5.2	Verify the application uses session tokens rather than static API secrets and keys, except with legacy implementations.	-
V3.5.3	Verify that stateless session tokens use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks.	+
V3.6.1	Verify that Relying Parties (RPs) specify the maximum authentication time to Credential Service Providers (CSPs) and that CSPs re-authenticate the user if they haven't used a session within that period.	-
V3.6.2	Verify that Credential Service Providers (CSPs) inform Relying Parties (RPs) of the last authentication event, to allow RPs to determine if they need to re-authenticate the user.	-
V3.7.1	Verify the application ensures a full, valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications.	+
V4.1.1	Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.	-
V4.1.2	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	+
V4.1.3	Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege.	+
V4.1.4	[DELETED, DUPLICATE OF 4.1.3]	-
V4.1.5	Verify that access controls fail securely including when an exception occurs.	+
V4.2.1	Verify that sensitive data and APIs are protected against Insecure Direct Object Reference (IDOR) attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records.	+
V4.2.2	Verify that the application or framework enforces a strong anti-CSRF mechanism to protect authenticated functionality, and effective anti-automation or anti-CSRF protects unauthenticated functionality.	+
V4.3.1	Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.	Bevinding 15
V4.3.2	Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.	+

V4.3.3	Verify the application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.	-
V5.1.1	Verify that the application has defenses against HTTP parameter pollution attacks, particularly if the application framework makes no distinction about the source of request parameters (GET, POST, cookies, headers, or environment variables).	+
V5.1.2	Verify that frameworks protect against mass parameter assignment attacks, or that the application has countermeasures to protect against unsafe parameter assignment, such as marking fields private or similar.	+
V5.1.3	Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (allow lists).	x
V5.1.4	Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers, e-mail addresses, telephone numbers, or validating that two related fields are reasonable, such as checking that suburb and zip/postcode match).	+
V5.1.5	Verify that URL redirects and forwards only allow destinations which appear on an allow list, or show a warning when redirecting to potentially untrusted content.	x
V5.2.1	Verify that all untrusted HTML input from WYSIWYG editors or similar is properly sanitized with an HTML sanitizer library or framework feature.	-
V5.2.2	Verify that unstructured data is sanitized to enforce safety measures such as allowed characters and length.	+
V5.2.3	Verify that the application sanitizes user input before passing to mail systems to protect against SMTP or IMAP injection.	-
V5.2.4	Verify that the application avoids the use of eval() or other dynamic code execution features. Where there is no alternative, any user input being included must be sanitized or sandboxed before being executed.	x
V5.2.5	Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.	-
V5.2.6	Verify that the application protects against SSRF attacks, by validating or sanitizing untrusted data or HTTP file metadata, such as filenames and URL input fields, and uses allow lists of protocols, domains, paths and ports.	+
V5.2.7	Verify that the application sanitizes, disables, or sandboxes user-supplied Scalable Vector Graphics (SVG) scriptable content, especially as they relate to XSS resulting from inline scripts, and foreignObject.	+
V5.2.8	Verify that the application sanitizes, disables, or sandboxes user-supplied scriptable or expression template language content, such as Markdown, CSS or XSL stylesheets, BBCode, or similar.	+
V5.3.1	Verify that output encoding is relevant for the interpreter and context required. For example, use encoders specifically for HTML values, HTML attributes, JavaScript, URL parameters, HTTP headers, SMTP, and others as the context requires, especially from untrusted inputs (e.g. names with Unicode or apostrophes, such as ねこ or O'Hara).	+
V5.3.2	Verify that output encoding preserves the user's chosen character set and locale, such that any Unicode character point is valid and safely handled.	+
V5.3.3	Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS.	+
V5.3.4	Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks.	-

V5.3.5	Verify that where parameterized or safer mechanisms are not present, context-specific output encoding is used to protect against injection attacks, such as the use of SQL escaping to protect against SQL injection.	-
V5.3.6	Verify that the application protects against JSON injection attacks, JSON eval attacks, and JavaScript expression evaluation.	+
V5.3.7	Verify that the application protects against LDAP injection vulnerabilities, or that specific security controls to prevent LDAP injection have been implemented.	-
V5.3.8	Verify that the application protects against OS command injection and that operating system calls use parameterized OS queries or use contextual command line output encoding.	+
V5.3.9	Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.	+
V5.3.10	Verify that the application protects against XPath injection or XML injection attacks.	+
V5.4.1	Verify that the application uses memory-safe string, safer memory copy and pointer arithmetic to detect or prevent stack, buffer, or heap overflows.	-
V5.4.2	Verify that format strings do not take potentially hostile input, and are constant.	-
V5.4.3	Verify that sign, range, and input validation techniques are used to prevent integer overflows.	-
V5.5.1	Verify that serialized objects use integrity checks or are encrypted to prevent hostile object creation or data tampering.	+
V5.5.2	Verify that the application correctly restricts XML parsers to only use the most restrictive configuration possible and to ensure that unsafe features such as resolving external entities are disabled to prevent XML eXternal Entity (XXE) attacks.	+
V5.5.3	Verify that deserialization of untrusted data is avoided or is protected in both custom code and third-party libraries (such as JSON, XML and YAML parsers).	+
V5.5.4	Verify that when parsing JSON in browsers or JavaScript-based backends, JSON.parse is used to parse the JSON document. Do not use eval() to parse JSON.	-
V6.1.1	Verify that regulated private data is stored encrypted while at rest, such as Personally Identifiable Information (PII), sensitive personal information, or data assessed likely to be subject to EU's GDPR.	-
V6.1.2	Verify that regulated health data is stored encrypted while at rest, such as medical records, medical device details, or de-anonymized research records.	-
V6.1.3	Verify that regulated financial data is stored encrypted while at rest, such as financial accounts, defaults or credit history, tax records, pay history, beneficiaries, or de-anonymized market or research records.	-
V6.2.1	Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.	-
V6.2.2	Verify that industry proven or government approved cryptographic algorithms, modes, and libraries are used, instead of custom coded cryptography.	-
V6.2.3	Verify that encryption initialization vector, cipher configuration, and block modes are configured securely using the latest advice.	-
V6.2.4	Verify that random number, encryption or hashing algorithms, key lengths, rounds, ciphers or modes, can be reconfigured, upgraded, or swapped at any time, to protect against cryptographic breaks.	-
V6.2.5	Verify that known insecure block modes (i.e. ECB, etc.), padding modes (i.e. PKCS#1 v1.5, etc.), ciphers with small block sizes (i.e. Triple-DES, Blowfish, etc.), and weak hashing algorithms (i.e. MD5, SHA1, etc.) are not used unless required for backwards compatibility.	-
V6.2.6	Verify that nonces, initialization vectors, and other single use numbers must not be used more than once with a given encryption key. The method of generation must be appropriate for the algorithm being used.	-

V6.2.7	Verify that encrypted data is authenticated via signatures, authenticated cipher modes, or HMAC to ensure that ciphertext is not altered by an unauthorized party.	-
V6.2.8	Verify that all cryptographic operations are constant-time, with no 'short-circuit' operations in comparisons, calculations, or returns, to avoid leaking information.	-
V6.3.1	Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved cryptographically secure random number generator when these random values are intended to be not guessable by an attacker.	-
V6.3.2	Verify that random GUIDs are created using the GUID v4 algorithm, and a Cryptographically-secure Pseudo-random Number Generator (CSPRNG). GUIDs created using other pseudo-random number generators may be predictable.	-
V6.3.3	Verify that random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances.	-
V6.4.1	Verify that a secrets management solution such as a key vault is used to securely create, store, control access to and destroy secrets.	Bevinding 11
V6.4.2	Verify that key material is not exposed to the application but instead uses an isolated security module like a vault for cryptographic operations.	Bevinding 10
V7.1.1	Verify that the application does not log credentials or payment details. Session tokens should only be stored in logs in an irreversible, hashed form.	x
V7.1.2	Verify that the application does not log other sensitive data as defined under local privacy laws or relevant security policy.	x
V7.1.3	Verify that the application logs security relevant events including successful and failed authentication events, access control failures, deserialization failures and input validation failures.	+
V7.1.4	Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens.	x
V7.2.1	Verify that all authentication decisions are logged, without storing sensitive session tokens or passwords. This should include requests with relevant metadata needed for security investigations.	+
V7.2.2	Verify that all access control decisions can be logged and all failed decisions are logged. This should include requests with relevant metadata needed for security investigations.	+
V7.3.1	Verify that all logging components appropriately encode data to prevent log injection.	Bevinding 8
V7.3.2	[DELETED, DUPLICATE OF 7.3.1]	-
V7.3.3	Verify that security logs are protected from unauthorized access and modification.	+
V7.3.4	Verify that time sources are synchronized to the correct time and time zone. Strongly consider logging only in UTC if systems are global to assist with post-incident forensic analysis.	x
V7.4.1	Verify that a generic message is shown when an unexpected or security sensitive error occurs, potentially with a unique ID which support personnel can use to investigate.	x
V7.4.2	Verify that exception handling (or a functional equivalent) is used across the codebase to account for expected and unexpected error conditions.	-
V7.4.3	Verify that a "last resort" error handler is defined which will catch all unhandled exceptions.	-
V8.1.1	Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.	-
V8.1.2	Verify that all cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.	-
V8.1.3	Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.	+

V8.1.4	Verify the application can detect and alert on abnormal numbers of requests, such as by IP, user, total per hour or day, or whatever makes sense for the application.	+
V8.1.5	Verify that regular backups of important data are performed and that test restoration of data is performed.	-
V8.1.6	Verify that backups are stored securely to prevent data from being stolen or corrupted.	-
V8.2.1	Verify the application sets sufficient anti-caching headers so that sensitive data is not cached in modern browsers.	-
V8.2.2	Verify that data stored in browser storage (such as localStorage, sessionStorage, IndexedDB, or cookies) does not contain sensitive data.	+
V8.2.3	Verify that authenticated data is cleared from client storage, such as the browser DOM, after the client or session is terminated.	+
V8.3.1	Verify that sensitive data is sent to the server in the HTTP message body or headers, and that query string parameters from any HTTP verb do not contain sensitive data.	+
V8.3.2	Verify that users have a method to remove or export their data on demand.	X
V8.3.3	Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way.	-
V8.3.4	Verify that all sensitive data created and processed by the application has been identified, and ensure that a policy is in place on how to deal with sensitive data.	-
V8.3.5	Verify accessing sensitive data is audited (without logging the sensitive data itself), if the data is collected under relevant data protection directives or where logging of access is required.	-
V8.3.6	Verify that sensitive information contained in memory is overwritten as soon as it is no longer required to mitigate memory dumping attacks, using zeroes or random data.	X
V8.3.7	Verify that sensitive or private information that is required to be encrypted, is encrypted using approved algorithms that provide both confidentiality and integrity.	Bevinding 5
V8.3.8	Verify that sensitive personal information is subject to data retention classification, such that old or out of date data is deleted automatically, on a schedule, or as the situation requires.	-
V9.1.1	Verify that TLS is used for all client connectivity, and does not fall back to insecure or unencrypted communications.	+
V9.1.2	Verify using up to date TLS testing tools that only strong cipher suites are enabled, with the strongest cipher suites set as preferred.	Bevinding 17
V9.1.3	Verify that only the latest recommended versions of the TLS protocol are enabled, such as TLS 1.2 and TLS 1.3. The latest version of the TLS protocol should be the preferred option.	Bevinding 17
V9.2.1	Verify that connections to and from the server use trusted TLS certificates. Where internally generated or self-signed certificates are used, the server must be configured to only trust specific internal CAs and specific self-signed certificates. All others should be rejected.	Bevinding 18
V9.2.2	Verify that encrypted communications such as TLS is used for all inbound and outbound connections, including for management ports, monitoring, authentication, API, or web service calls, database, cloud, serverless, mainframe, external, and partner connections. The server must not fall back to insecure or unencrypted protocols.	x
V9.2.3	Verify that all encrypted connections to external systems that involve sensitive information or functions are authenticated.	+
V9.2.4	Verify that proper certification revocation, such as Online Certificate Status Protocol (OCSP) Stapling, is enabled and configured.	-
V9.2.5	Verify that backend TLS connection failures are logged.	+
V10.1.1	Verify that a code analysis tool is in use that can detect potentially malicious code, such as time functions, unsafe file operations and network connections.	-

V10.2.1	Verify that the application source code and third party libraries do not contain unauthorized phone home or data collection capabilities. Where such functionality exists, obtain the user's permission for it to operate before collecting any data.	-
V10.2.2	Verify that the application does not ask for unnecessary or excessive permissions to privacy related features or sensors, such as contacts, cameras, microphones, or location.	+
V10.2.3	Verify that the application source code and third party libraries do not contain back doors, such as hard-coded or additional undocumented accounts or keys, code obfuscation, undocumented binary blobs, rootkits, or anti-debugging, insecure debugging features, or otherwise out of date, insecure, or hidden functionality that could be used maliciously if discovered.	-
V10.2.4	Verify that the application source code and third party libraries do not contain time bombs by searching for date and time related functions.	-
V10.2.5	Verify that the application source code and third party libraries do not contain malicious code, such as salami attacks, logic bypasses, or logic bombs.	-
V10.2.6	Verify that the application source code and third party libraries do not contain Easter eggs or any other potentially unwanted functionality.	-
V10.3.1	Verify that if the application has a client or server auto-update feature, updates should be obtained over secure channels and digitally signed. The update code must validate the digital signature of the update before installing or executing the update.	-
V10.3.2	Verify that the application employs integrity protections, such as code signing or subresource integrity. The application must not load or execute code from untrusted sources, such as loading includes, modules, plugins, code, or libraries from untrusted sources or the Internet.	-
V10.3.3	Verify that the application has protection from subdomain takeovers if the application relies upon DNS entries or DNS subdomains, such as expired domain names, out of date DNS pointers or CNAMEs, expired projects at public source code repos, or transient cloud APIs, serverless functions, or storage buckets (*autogen-bucket-id*.cloud.example.com) or similar. Protections can include ensuring that DNS names used by applications are regularly checked for expiry or change.	-
V11.1.1	Verify that the application will only process business logic flows for the same user in sequential step order and without skipping steps.	-
V11.1.2	Verify that the application will only process business logic flows with all steps being processed in realistic human time, i.e. transactions are not submitted too quickly.	-
V11.1.3	Verify the application has appropriate limits for specific business actions or transactions which are correctly enforced on a per user basis.	+
V11.1.4	Verify that the application has anti-automation controls to protect against excessive calls such as mass data exfiltration, business logic requests, file uploads or denial of service attacks.	+
V11.1.5	Verify the application has business logic limits or validation to protect against likely business risks or threats, identified using threat modeling or similar methodologies.	-
V11.1.6	Verify that the application does not suffer from "Time Of Check to Time Of Use" (TOCTOU) issues or other race conditions for sensitive operations.	+
V11.1.7	Verify that the application monitors for unusual events or activity from a business logic perspective. For example, attempts to perform actions out of order or actions which a normal user would never attempt.	-
V11.1.8	Verify that the application has configurable alerting when automated attacks or unusual activity is detected.	Bevinding 8
V12.1.1	Verify that the application will not accept large files that could fill up storage or cause a denial of service.	+
V12.1.2	Verify that the application checks compressed files (e.g. zip, gz, docx, odt) against maximum allowed uncompressed size and against maximum number of files before uncompressing the file.	+

V12.1.3	Verify that a file size quota and maximum number of files per user is enforced to ensure that a single user cannot fill up the storage with too many files, or excessively large files.	+
V12.2.1	Verify that files obtained from untrusted sources are validated to be of expected type based on the file's content.	+
V12.3.1	Verify that user-submitted filename metadata is not used directly by system or framework filesystems and that a URL API is used to protect against path traversal.	-
V12.3.2	Verify that user-submitted filename metadata is validated or ignored to prevent the disclosure, creation, updating or removal of local files (LFI).	-
V12.3.3	Verify that user-submitted filename metadata is validated or ignored to prevent the disclosure or execution of remote files via Remote File Inclusion (RFI) or Server-side Request Forgery (SSRF) attacks.	-
V12.3.4	Verify that the application protects against Reflective File Download (RFD) by validating or ignoring user-submitted filenames in a JSON, JSONP, or URL parameter, the response Content-Type header should be set to text/plain, and the Content-Disposition header should have a fixed filename.	+
V12.3.5	Verify that untrusted file metadata is not used directly with system API or libraries, to protect against OS command injection.	+
V12.3.6	Verify that the application does not include and execute functionality from untrusted sources, such as unverified content distribution networks, JavaScript libraries, node npm libraries, or server-side DLLs.	+
V12.4.1	Verify that files obtained from untrusted sources are stored outside the web root, with limited permissions.	+
V12.4.2	Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload and serving of known malicious content.	-
V12.5.1	Verify that the web tier is configured to serve only files with specific file extensions to prevent unintentional information and source code leakage. For example, backup files (e.g. .bak), temporary working files (e.g. .swp), compressed files (.zip, .tar.gz, etc) and other extensions commonly used by editors should be blocked unless required.	+
V12.5.2	Verify that direct requests to uploaded files will never be executed as HTML/JavaScript content.	+
V12.6.1	Verify that the web or application server is configured with an allow list of resources or systems to which the server can send requests or load data/files from.	+
V13.1.1	Verify that all application components use the same encodings and parsers to avoid parsing attacks that exploit different URI or file parsing behavior that could be used in SSRF and RFI attacks.	+
V13.1.2	[DELETED, DUPLICATE OF 4.3.1]	-
V13.1.3	Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.	-
V13.1.4	Verify that authorization decisions are made at both the URI, enforced by programmatic or declarative security at the controller or router, and at the resource level, enforced by model-based permissions.	+
V13.1.5	Verify that requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type).	+
V13.2.1	Verify that enabled RESTful HTTP methods are a valid choice for the user or action, such as preventing normal users using DELETE or PUT on protected API or resources.	-
V13.2.2	Verify that JSON schema validation is in place and verified before accepting input.	-
V13.2.3	Verify that RESTful web services that utilize cookies are protected from Cross-Site Request Forgery via the use of at least one or more of the following: double submit cookie pattern, CSRF nonces, or Origin request header checks.	-
V13.2.4	[DELETED, DUPLICATE OF 11.1.4]	-

V13.2.5	Verify that REST services explicitly check the incoming Content-Type to be the expected one, such as application/xml or application/json.	-
V13.2.6	Verify that the message headers and payload are trustworthy and not modified in transit. Requiring strong encryption for transport (TLS only) may be sufficient in many cases as it provides both confidentiality and integrity protection. Per-message digital signatures can provide additional assurance on top of the transport protections for high-security applications but bring with them additional complexity and risks to weigh against the benefits.	+
V13.3.1	Verify that XSD schema validation takes place to ensure a properly formed XML document, followed by validation of each input field before any processing of that data takes place.	+
V13.3.2	Verify that the message payload is signed using WS-Security to ensure reliable transport between client and service.	+
V13.4.1	Verify that a query allow list or a combination of depth limiting and amount limiting is used to prevent GraphQL or data layer expression Denial of Service (DoS) as a result of expensive, nested queries. For more advanced scenarios, query cost analysis should be used.	-
V13.4.2	Verify that GraphQL or other data layer authorization logic should be implemented at the business logic layer instead of the GraphQL layer.	-
V14.1.1	Verify that the application build and deployment processes are performed in a secure and repeatable way, such as CI / CD automation, automated configuration management, and automated deployment scripts.	-
V14.1.2	Verify that compiler flags are configured to enable all available buffer overflow protections and warnings, including stack randomization, data execution prevention, and to break the build if an unsafe pointer, memory, format string, integer, or string operations are found.	-
V14.1.3	Verify that server configuration is hardened as per the recommendations of the application server and frameworks in use.	-
V14.1.4	Verify that the application, configuration, and all dependencies can be re-deployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.	+
V14.1.5	Verify that authorized administrators can verify the integrity of all security-relevant configurations to detect tampering.	+
V14.2.1	Verify that all components are up to date, preferably using a dependency checker during build or compile time.	Bevinding 9
V14.2.2	Verify that all unneeded features, documentation, sample applications and configurations are removed.	+
V14.2.3	Verify that if application assets, such as JavaScript libraries, CSS or web fonts, are hosted externally on a Content Delivery Network (CDN) or external provider, Subresource Integrity (SRI) is used to validate the integrity of the asset.	+
V14.2.4	Verify that third party components come from pre-defined, trusted and continually maintained repositories.	+
V14.2.5	Verify that a Software Bill of Materials (SBOM) is maintained of all third party libraries in use.	-
V14.2.6	Verify that the attack surface is reduced by sandboxing or encapsulating third party libraries to expose only the required behaviour into the application.	-
V14.3.1	[DELETED, DUPLICATE OF 7.4.1]	-
V14.3.2	Verify that web or application server and application framework debug modes are disabled in production to eliminate debug features, developer consoles, and unintended security disclosures.	+
V14.3.3	Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.	+

V14.4.1	Verify that every HTTP response contains a Content-Type header. Also specify a safe character set (e.g., UTF-8, ISO-8859-1) if the content types are text/*, /+xml and application/xml. Content must match with the provided Content-Type header.	+
V14.4.2	Verify that all API responses contain a Content-Disposition: attachment; filename="api.json" header (or other appropriate filename for the content type).	-
V14.4.3	Verify that a Content Security Policy (CSP) response header is in place that helps mitigate impact for XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.	+
V14.4.4	Verify that all responses contain a X-Content-Type-Options: nosniff header.	x
V14.4.5	Verify that a Strict-Transport-Security header is included on all responses and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains.	+
V14.4.6	Verify that a suitable Referrer-Policy header is included to avoid exposing sensitive information in the URL through the Referer header to untrusted parties.	+
V14.4.7	Verify that the content of a web application cannot be embedded in a third-party site by default and that embedding of the exact resources is only allowed where necessary by using suitable Content-Security-Policy: frame-ancestors and X-Frame-Options response headers.	+
V14.5.1	Verify that the application server only accepts the HTTP methods in use by the application/API, including pre-flight OPTIONS, and logs/alerts on any requests that are not valid for the application context.	+
V14.5.2	Verify that the supplied Origin header is not used for authentication or access control decisions, as the Origin header can easily be changed by an attacker.	+
V14.5.3	Verify that the Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin header uses a strict allow list of trusted domains and subdomains to match against and does not support the "null" origin.	+
V14.5.4	Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.	+

Tabel 5 - Checklist van OWASP ten behoeve van de Application Security Verification Standard.

Documentbeheer

Versiebeheer	
Projectnaam	Beveiligingsonderzoek OSV-2020
Referentie	PR-210537
Opdrachtgever	Kiesraad
Onderwerp	Pentest Rapport
Datum	9 maart 2022
Versie	1.1
Status	Definitief

Deze versie vervangt alle voorgaande versies van dit document. Vernietig alle voorgaande exemplaren!

Distributielijst

Versie	Datum	Verspreidingsvorm	Naam/functie/opmerking
1.0	17 februari 2022	PDF via Client Portal	Kiesraad; aangeleverd door Sam Berkhout
1.1	9 maart 2022	PDF via Client Portal	Kiesraad; aangeleverd door Sam Berkhout

Reviews

Versie	Datum	Naam	Functie
0.2	21 januari 2022	Paul Pols	Principal Security Expert
0.4	11 februari 2022	Erik de Jong	Strategy Advisor
0.6	16 februari 2022	Paul Pols	Principal Security Expert
0.9	17 februari 2022	Dennis de Hoog	PS Regional Head

Wijzigingen

Versie	Datum	Opmerkingen
0.1	17 januari 2022	Eerste versie
0.3	24 januari 2022	Review verwerkt
0.5	14 februari 2022	Review verwerkt
1.0	17 februari 2022	Review voorafgaand aan formele vrijgave van de resultaten aan De Kiesraad verwerkt
1.1	9 maart 2022	Uiteindelijke versie

Gerelateerde documenten

Versie	Datum	Omschrijving	Opmerkingen
0.1	28 december 2021	Standaard Pentest Vervanging OSV 0.1	Projectinitiatiedocument

Fox-IT

Fox-IT voorkomt, onderzoekt en beperkt de meest serieuze dreigingen door cyberaanvallen, datalekken of fraude met innovatieve oplossingen voor overheid, defensie, politie, vitale infrastructuur, banken en grote bedrijven wereldwijd. In zijn aanpak combineert het bedrijf slimme ideeën met technologie om hiermee innovatieve oplossingen te bieden die zorgen voor een veilige maatschappij. Fox-IT ontwikkelt producten en maatwerkoplossingen om de beveiliging van gevoelige overheidssystemen te garanderen, industriële netwerken te beschermen, online bankiersystemen te verdedigen en strikt vertrouwelijke data te beveiligen.

Bezoek onze website voor meer informatie over Fox-IT en onze partners.

FOX IT
part of nccgroup

fox-it.com

Fox-IT
Olof Palmestraat 6, Delft
Postbus 638, 2600 AP Delft
Nederland

T +31 (0)15 284 7999
F +31 (0)15 284 7990
fox@fox-it.com