Corné Noorlander (1054564), Fabio Wolthuis (1093379), Hannah Saunders (1093894) & Merel van der Leeden (1103194)

PO: DIEDERIK MOORLAG,  Project 3/4

Beveiliginsrapport

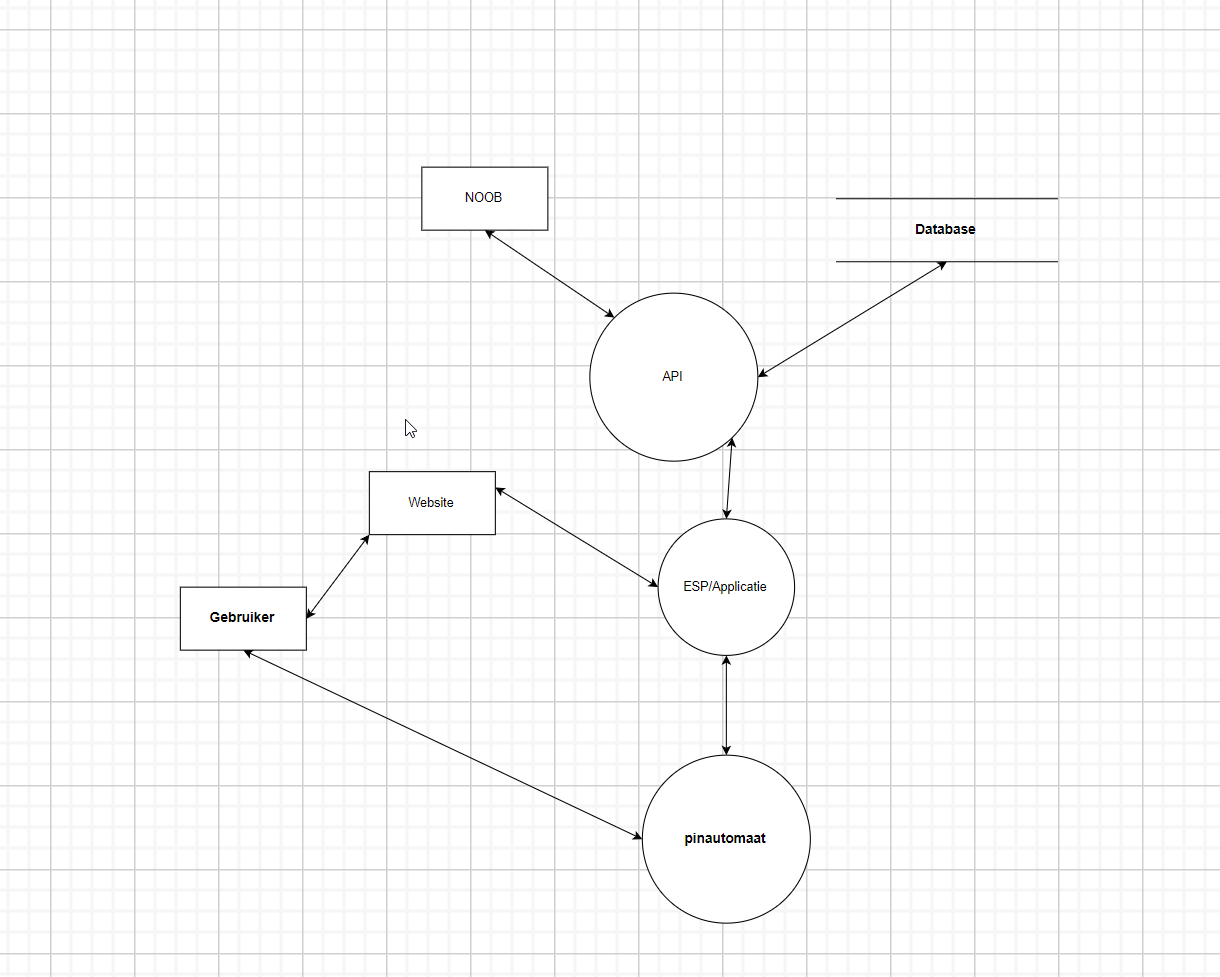
Inhoud

[Inleiding 2](#_Toc198812117)

# Inleiding

Vanuit de bankenvereniging *NOOB* is er gevraagd om een pinautomaat met een banksysteem te ontwerpen en realiseren. Omdat er binnen een banksysteem gevoelige data staat over de klanten, moet zo’n systeem goed beveiligd zijn. In dit verslag wordt er gekeken naar de veiligheidsrisico’s die opkomen bij het maken van een banksysteem met een pinautomaat en hoe deze risico’s verkleint en/of tegengegaan kunnen worden.

# Data Flow Diagram



# Risicoanalyse

## Aanvallen op het systeem

**SQL-injectie**

Op het bansysteem staat onder anderen een database waarop (gevoelige)data staat van de gebruikers. Om hier data uit te krijgen maakt de API gebruik van zogenaamde SQL query’s. Het risico hiermee is dat gebruikers kwaadaardige SQL query’s in zouden kunnen voeren en daarmee aan gevoelige data kunnen komen.[1]

**Cross-Site Scripting (XSS)**

Het banksysteem maak gebruik van een webpagina om weer te geven wat er mogelijk is op de pinautomaat. Helaas betekent dit dat er de mogelijkheid is om kwaadaardige scriptcode te injecteren in de webapplicatie, dit heet Cross-Site Scripting ook wel bekend als XSS. XSS zorgt ervoor dat de gebruikers invoer die de webapplicatie ontvangt niet correct wordt verwerkt waardoor de mogelijkheid ontstaat dat er complete controle over de webbrowser te krijgen is. [2]

## Aanvallen op de verbindingen

## Aanvallen op fysieke hardware

**Water**

De pinautomaat zit vol met gevoelige componenten die weinig bescherming hebben tegen water. Dit geeft de mogelijkheid om het systeem te kunnen beschadigen en/of slopen als er water in contact komt met een van de componenten. Dit kan ervoor zorgen dat er kortsluiting komt.

# Oplossingen

Door deze methodes toe te passen aan uw systeem is uw webapplicatie zo effectief mogelijk beschermd tegen externe aanvallen. Maar het is belangrijk om te weten dat geen enkele beveiliging’s methode 100% onkraakbaar is. Om de kans hierop te verminderen is het best om een combinatie van methodes te gebruiken.

## Het systeem

**SQL-injectie**

Om de mogelijkheid op SQL-injectie te voorkomen zijn er meerdere opties om gebruik van te maken. Optie één is het gebruik maken van Prepared Statements. Een prepared statement is een feature die gebruikt wordt om dezelfde (of vergelijkbare) SQL-statements herhaaldelijk uit te voeren met een hoge efficiëntie. Door het te gebruiken van prepared statements wordt de SQL-code gescheiden van de data. Hiermee wordt er voorkomen dat een aanvaller kwaadaardige SQL-code kan injecteren, omdat de data dan niet wordt geïnterpreteerd als code.

Optie twee is het gebruik maken van Stored Procedures. Stored Procedure is een SQL-code die steeds kan opgeslagen en hergebruikt kan worden. In een zekere zin lijkt dit op Prepared Statements, omdat ze beide de SQL-code scheiden van de data. Maar Stored Procedures worden opgeslagen in de database zelf en kunnen aangeroepen worden vanuit de applicatie.

Optie drie is het gebruik maken van Input Validatie. Input Validatie is een proces van verificatie om te controleren of de gebruiker input voldoet aan specifiek gestelde criteria punten. Het gebruiken van Input Validatie voordat het in een SQL-query wordt gezet is essentieel om ervoor te zorgen dat de invoer de verwachte invoer heeft en onverwachte of kwaadaardige waardes zal afwijzen.

Optie vier is het gebruikt maken van Least Privilege. Least Privilege zorgt ervoor dat elke verbinding met de database zo min mogelijk rechten hebben. Least Privilege heeft als voordeel dat als erin geslaagd wordt om SQL-injectie te gebruiker er zo min mogelijk schade gericht kan worden. [1]

**Cross-Site Scripting (XSS)**

Om XSS-aanvallen te voorkomen is er de optie om de gebruikersinvoer te filteren. Dit begint bij het ontvangen van de invoer. Hier moeten sterke filters worden toegepast voor wat er verwacht mag worden qua geldige invoer. Het is hier de bedoeling dat deze invoer nooit ongefilterd in de webapplicatie verwerkt mag worden. De filter zorgt ervoor dat de speciale tekens, zoals “, ‘, <, >, omgezet worden in hun gecodeerde equivalenten om te voorkomen dat kwaadaardige scripts kunnen worden uitgevoerd. Het is ook mogelijk om te filteren op inhoudelijke aspecten zoals dat er alleen maar cijfer ingevoerd mogen worden of dat er een limiet staat op hoeveel karakters er maximaal mogen worden ingevoerd.

Er is ook de optie om als extra beveiligingslaag Content Security Policy (CSP) te implementeren. CSP is een functie in de browser die bepaalt welke bronnen of scripts uitgevoerd mogen worden. [2], [3]

## De componenten

**Water**

Om te voorkomen dat het systeem beschadigd wordt door water is het een optie om het systeem te beschermen door middel van waterdichte componenten, de componenten in een waterdichte omgeving te plaatsen of om een waterdichte laag om de pinautomaat heen te bevestigen.

# Bronvermelding

[1] pentests, ‘SQL Injectie - Hoe uitbuiten, beschermen en SQL injectie voorkomen’, Pentests. Geraadpleegd: 2 juni 2025. [Online]. Beschikbaar op: https://pentests.nl/pentest-blog/sql-injectie/

[2] pentests, ‘Cross-Site Scripting (XSS): wat is het en hoe te voorkomen’, Pentests. Geraadpleegd: 2 juni 2025. [Online]. Beschikbaar op: https://pentests.nl/pentest-blog/cross-site-scripting-wat-is-het-en-hoe-te-voorkomen/

[3] pentests, ‘Content Security Policy (CSP): maak jouw website veiliger met een CSP’, Pentests. Geraadpleegd: 2 juni 2025. [Online]. Beschikbaar op: https://pentests.nl/pentest-blog/csp-content-security-policy/

# Changelog

|  |  |  |
| --- | --- | --- |
| Datum | Versie | wijzigingen |
| 22/05/2025 | 1.0 | Start beveiligingsrapport, toevoeging diagram |
| 27/05/2025 | 2.0 | Voortgang rapport |
| 02/06/2025 | 3.0 | Systeem toegevoegd |