IPBEV – Beveiligingsplan

Hogeschool Leiden - Informatica

Voor Stichting Wireless Leiden waren wij als student ingezet om een interactieve nodemap te maken om de verschillende nodes van Wireless Leiden te kunnen tonen aan gebruikers en beheerders.

Pieter Naber & David de Boer
17-4-2010

Inhoudsopgave

Pieter Naber (s1054204)
David de Boer (s1013714)

Pieter Naber (s1054204)
David de Boer (s1013714)

Pieter Naber (s1054204)
David de Boer (s1013714)

Pieter Naber (s1054204)
David de Boer (s1013714)

1 Versie beheer

Versie:	Datum:	Actor:	Wijzigingen:
0.1	Vr 12-02-2010	Pieter	Eerste opzet gemaakt voor het beveiligingsplan
0.2	Ma 15-02-2010	Pieter	Eerste beveiligingsrisico's aan beveiligingsplan toegevoegd
0.3	Di 16-02-2010	Pieter	Beveiligingsrisico's toegevoegd, zonder uitwerking
0.4	Zo 21-02-2010	Pieter	Versie beheer toegevoegd en up-to-date gemaakt
0.5	Ma 01-03-2010	Pieter	Flink veel punten uitgewerkt, risico tabel toegevoegd
0.6	Wo 03-03-2010	David	Applicatiebeveiliging uitgewerkt
1.0	Wo 03-03-2010	Beiden	Definitieve indeling beveiligingsplan af en ingeleverd
1.0.1	Ma 08-03-2010	Pieter	Eerste concept netwerk beveiligingspunten opgezet
1.0.2	Di 09-03-2010	Pieter	Alle netwerk beveiligingspunten uitgewerkt
1.0.3	Di 09-03-2010	David	Inleiding toegevoegd
1.1	Di 09-03-2010	Beiden	Netwerk beveiligingspunten af en ingeleverd
1.1.1	Di 09-03-2010	Pieter	Eerste opzet gemaakt voor data beveiligingspunten
1.1.2	Di 09-03-2010	Pieter	Data beveiligingspunten toegevoegd aan de hand van interventie
1.1.3	Ma 15-03-2010	Pieter	Data beveiligingspunten uitgewerkt en toegevoegd.
1.1.4	Di 16-03-2010	Pieter	Data beveiligingspunten verder uitgewerkt, hoofdstukken gemaakt
1.1.5	Wo 17-03-2010	Beiden	Data beveiligingspunten: uitgewerkt en inleiding toegevoegd
1.2	Wo 17-03-2010	Beiden	Data beveiligingsputen af en ingeleverd
1.2.1	Ma 15-03-2010	Pieter	Eerste opzet voor de proces beveiligingspunten gemaakt
1.2.2	Di 16-03-2010	Pieter	Proces beveiligingspunten aangepast aan de hand van input van de opdrachtgever.
1.2.3	Wo 17-03-2010	Pieter	Proces beveiligingspunten aangepast aan de hand van de inverventie.
1.2.4	Wo 17-03-2010	David	Inleiding geschreven
1.2.5	Di 23-03-2010	Pieter	Proces beveiligingspunten 1 en 2 volledig uitgewerkt
1.2.6	Wo 24-03-2010	David	Proces beveiligingspunten 3 en 4 volledig uitgewerkt
1.3	Wo 24-03-2010	Beiden	Proces beveiligingspunten af en ingeleverd
1.3.1	Di 23-03-2010	Pieter	Eerste opzet gemaakt voor applicatie beveiligingspunten
1.3.2	Wo 24-04-2010	Pieter	Alle applicatie beveiligingspunten iets uitgewerkt
1.3.3	Di 30-03-2010	Pieter	Applicatie beveiligingspunten 1 en 2 uitgewerkt
1.3.4	Wo 31-03-2010	David	Inleiding aangepast en applicatie beveiligingspunt 3 verder uitgewerkt
1.3.5	Wo 31-03-2010	Pieter	Laatste applicatie beveiligingspunten aangepast
1.4	Wo 31-03-2010	Beiden	Applicatie beveiligingspunten af en ingeleverd
1.5	Za 17-04-2010	Pieter	Beveiligingsplan samengevoegd uit voorgaande documenten
2.0	Zo 18-03-2010	Beiden	Beveiligingsplan helemaal af!

Pieter Naber (s1054204)
David de Boer (s1013714)

2 Inleiding

Tijdens het project "Beveiliging" (IPBEV) krijgt elke groep een aparte (externe) opdracht. Binnen de groepsopdracht zijn de studenten die IPBEV volgen verantwoordelijk voor de totale beveiliging van het product.

Voor onze groep was dit een externe opdracht bij Stichting Wireless Leiden waarbij we een interactieve nodemap applicatie hebben gebouwd. In dit verslag zullen we het hebben over het beveiligen van de applicatie die door ons geschreven is. We zullen aangeven wat er belangrijk is om te beveiligen en waarom dit zo is. We zullen hierbij kijken naar netwerkbeveiliging, databeveiliging, procesbeveiliging en applicatie beveiliging.

Pieter Naber (s1054204)
David de Boer (s1013714)

3 Functionele beschrijving

Het project Interactive Node Map is ontstaan uit het probleem, dat de wireless nodes van Stichting Wireless Leiden, niet overzichtelijk, en met te weinig informatie op een kaart staan. Het is aan ons de taak een applicatie te maken die ervoor moet zorgen dat alle nodes met bijbehorende informatie overzichtelijk op een dynamische en betrouwbare kaart te zien zijn.

De opdracht bestaat uit het maken van een dynamische kaart die op een overzichtelijk manier alle wireless nodes van Stichting Wireless Leiden op een overzichtelijk manier moet laten zien. Over Leiden en omgeving staan er meerdere nodes, op de kaart moet allerlei informatie te zien zijn over specifieke nodes, zoals en welke andere nodes de node gekoppeld is en de status van de node.

De opdrachtgever heeft al een klein begin van de nieuwe nodemap geprogrammeerd. Er moeten nu wat zaken verbeterd en/of uitgebreid worden.

- Het bereik van de nodes maakt de kaart momenteel onleesbaar. Leiden bestaat nu uit één grote groene vlek en straatnamen zijn niet te lezen. Onze taak is een manier vinden om het bereik weer te geven en bovendien de kaart leesbaar te maken.
- De verbindingen tussen de nodes zorgen net als het bereik voor een onleesbare kaart. Ook zijn de verbindingen statisch, en zou het wenselijk zijn om bijvoorbeeld de verbindingen groter of kleiner af te beelden afhankelijk van de bandwidth die gebruikt wordt. Hier moeten we een oplossing voor bedenken.
- Elke node laat nu alleen de locatie zien. Meer informatie is gewenst voor zowel de doorsnee gebruiker als de beheerder.
- De applicatie moet geprogrammeerd worden. Een methode om dit te doen is Python.

Pieter Naber (s1054204)
David de Boer (s1013714)

4 Beveiligingsrisico's

Tijdens IPBEV zullen er 4 groepen risico's worden besproken en behandeld. Elke week zal er een onderdeel verder besproken worden tijdens de lessen van ICS2 en door ons zelf verder worden behandeld voor IPBEV. De groepen zijn:

- Netwerkbeveiliging
- Databeveiliging
- Procesbeveiliging
- Applicatiebeveiliging

Risico kan berekent worden door de "formule": "Risico = Kans x Impact". Daarom zullen we per risico de volgende punten behandelen:

1. Benoemen van risico
2. Kans op risico
3. Impact van risico
4. Wat kan je er aan doen?

Pieter Naber (s1054204)
David de Boer (s1013714)

5 Netwerkbeveiliging

Als er zich problemen voordoen met het netwerk, dan kan dit grote problemen veroorzaken voor de applicatie. In de netwerkbeveiliging gaat het over het beschrijven van de risico's die te maken hebben met het netwerk van de node map applicatie. Dit omvat niet alleen het fysieke netwerk, maar ook de data die hierover verstuurd en ontvangen wordt. Omdat de applicatie ook afhankelijk is van de bereikbaarheid van de kaarten, is dit ook meegenomen in het netwerk beveiligingsplan.

5.1 (Distributed) Denial-Of-Service

5.1.1 Omschrijving risico

De server waar onze applicatie op draait, de router en de switch die de server gebruikt kunnen het slachtoffer worden van een (D)DOS aanval. Hierbij voert een computer of een groep computers een groot aantal requests uit op een andere computer, server, router of een ander apparaat. Hierdoor raakt het aangevallen apparaat over belast en raakt onbereikbaar voor normale requests.

5.1.2 Kans op risico

De kans hierop is middel. Het gaat om een kleine applicatie van een stichting en zal hierdoor minder in trek zijn bij hackers dan bijvoorbeeld applicaties van grote bedrijven. Wel wordt de server gehost bij Hogeschool Leiden en de back-up server bij Universiteit Leiden. Hierdoor zouden ook routers en switches slachtoffer kunnen worden van een (D)DOS, zonder dat onze applicatie of Stichting Wireless Leiden als doel wordt gezien.

5.1.3 Impact van risico

De impact van het risico is klein. Als onze applicatie niet meer draait, betekent niet dat het netwerk van de stichting niet meer werkt. Alleen kunnen gebruikers en beheerders de kaart niet meer gebruiken. Verder heeft Stichting Wireless Leiden ook al een back-up server draaien op een andere locatie, waardoor het risico nog verder wordt geminimaliseerd.

5.1.4 Minimaliseren van het risico

Bescherming tegen (D)DOS aanvallen is vaak duur en daardoor geen optie voor de stichting. Waarschijnlijk is het beste om hier niets extra's aan te doen, meegenomen dat de stichting al een back-up server tot zijn beschikking heeft.

5.2 Uitval netwerk

5.2.1 Omschrijving risico

Het netwerk zelf kan uitvallen door bijvoorbeeld een stroomstoring. Ook zou door onwetendheid de server uitgezet kunnen worden door een beheerder of zou een router of switch in het netwerk uitvallen. Als het netwerk uitvalt, dan zou dit kunnen leiden tot niet opgeslagen data en de onbereikbaarheid van de applicatie.

5.2.2 Kans op risico

De kans dat dit optreed is klein. Dit komt doordat de stichting een back-up server heeft bij Universiteit Leiden en doordat er bij beiden een goed opgebouwd netwerk is waar het niet zomaar mogelijk is om de server uit te zetten.

Pieter Naber (s1054204)
David de Boer (s1013714)

5.2.3 Impact van risico
De impact van dit risico is klein, van kritieke gegevens wordt elk uur een back-up gemaakt, van minder kritieke gegevens eens per dag. Verder valt bij storing alleen onze applicatie uit, en niet het hele netwerk van Wireless Leiden.

5.2.4 Minimaliseren van het risico
Omdat de stichting al een back-up server heeft, kunnen we weinig extra's doen om het risico nog verder te minimaliseren wat binnen de begroting van de stichting zou vallen.

5.3 Afhankelijkheid landkaarten software

5.3.1 Omschrijving risico
Naast het eigen netwerk is het systeem afhankelijk van landkaarten software, zoals bijvoorbeeld Google Maps. Dit landkaarten systeem kan uitvallen of ophouden met bestaan, waardoor ook onze software niet meer zal werken.

5.3.2 Kans op risico
De kans dat dit gebeurd is verschillend en ligt aan de keuze welke landkaarten software we gaan gebruiken. We zetten deze op middel.

5.3.3 Impact van risico
De impact van dit risico is erg groot. Zonder de landkaarten software werkt de hele applicatie niet meer.

5.3.4 Minimaliseren van het risico
Om dit risico aan te pakken zouden we kunnen kijken naar meerdere varianten van landkaarten software en zorgen dat de software op meerdere varianten werkt. De software zou automatisch moeten kunnen schakelen tussen verschillende varianten, zodat er geen hinder ontstaat als een variant uit valt. Welke de beste opties zouden zijn wordt momenteel onderzocht door de IPACC groep als deel van hun "Analyse rapport".

5.4 Afluisteren van netwerk

5.4.1 Omschrijving risico
Iemand luistert af welke berichten worden verstuurd / ontvangen op de server.

5.4.2 Kans op risico
De kans dat dit gebeurd is klein, gezien het feit dat het gaan om een stichting erg klein.

5.4.3 Impact van risico
De impact van dit risico is klein, we verzenden alleen data over de belasting van nodes, geen kritieke informatie.

5.4.4 Minimaliseren van het risico
Een optie zou zijn om met een bepaalde regelmaat sowieso een bericht te sturen, soms met nutteloze informatie, soms met nuttige informatie.

Pieter Naber (s1054204)
David de Boer (s1013714)

5.5 Onderschepping van data

5.5.1 Omschrijving risico

Iemand onderschept verstuurde berichten van of naar de server zodat deze niet aankomen.

5.5.2 Kans op risico

De kans dat dit gebeurd is klein, gezien het feit dat het gaan om een stichting erg klein.

5.5.3 Impact van risico

De impact van dit risico is middel, omdat een vrijwilliger het niet ontvangen van data zou kunnen interpreteren als een defecte node en dan onnodig op pad zou kunnen gaan.

5.5.4 Minimaliseren van het risico

Zorgen dat er geen data onderschept kan worden kan door het netwerk zelf goed te beveiligen waardoor niet iedereen er zomaar op kan.

5.6 Snooping

5.6.1 Omschrijving risico

Zoeken in andermans gegevens zonder dat diegene daar rechten toe heeft. Denk aan een normale bezoeker van de website die naast de normale node map ook gegevens ziet over belasting van verbindingen en nodes zelf.

5.6.2 Kans op risico

De kans dat dit gebeurd is klein gezien de omvang van het project erg klein.

5.6.3 Impact van risico

De impact van dit risico is klein omdat er geen kritieke informatie wordt verzonden.

5.6.4 Minimaliseren van het risico

Dit punt valt te grotendeels te voorkomen door het goed testen van de beveiliging van de applicatie.

5.7 Analyse en traffic

5.7.1 Omschrijving risico

Kijken hoevaak berichten worden verstuurd tussen nodes en de server(s). Hier is de inhoud niet belangrijk, maar het aantal berichten en wanneer deze worden verstuurd wel.

5.7.2 Kans op risico

De kans dat dit gebeurd is klein gezien het feit dat het gaan om een stichting erg klein.

5.7.3 Impact van risico

De impact van dit risico is klein, we verzenden alleen data over de belasting van nodes, geen kritieke informatie.

5.7.4 Minimaliseren van het risico

Een optie zou zijn om met een bepaalde regelmaat sowieso een bericht te sturen, soms met nutteloze informatie, soms met nuttige informatie.

Pieter Naber (s1054204)
David de Boer (s1013714)

5.8 Uiteindelijke toepassingen in de software

Stichting Wireless Leiden heeft al veel dingen goed op orde. Een backup server op een andere locatie is hier een voorbeeld van. Verder maakt Wireless Leiden gebruik van Open Source software en zijn dingen als het afluisteren van het netwerk minder belangrijk omdat de data toch openbaar bereikbaar is. Mede door bovenstaande zaken, maar ook door tijdgebrek hebben we met bovenstaande beveiligingspunten niets kunnen doen.

Ook het switchen tussen verschillende leveranciers van landkaarten software (Google Maps, Yahoo Maps, Open Street Map) is uiteindelijk niet geïmplementeerd. Vooral door tijdgebrek om rekening te houden met de verschillende interpretaties van KML bestanden door de verschillende leveranciers. Hoewel KML namelijk een open standaard is, is Google, die de open standaard indirect onderhoud, de enige die volledig up-to-date is.

Pieter Naber (s1054204)
David de Boer (s1013714)

6 Databeveiliging

Bij databeveiliging worden de risico's behandeld die betrekking hebben tot de data van de nodemap applicatie. Al het onderzoek dat wordt gedaan met verkeerde gegevens, kan weggegooid worden. Daarom is het belangrijkste onderdeel, van het data beveiligingsplan, de correctheid van de data. De Stichting Wireless Leiden bestaat al enige tijd en daarom is de fysieke laag van het systeem al aanwezig. Toch worden er nog onderwerpen hierover behandeld, omdat dit ook een grote impact kan hebben op onze nodemap applicatie.

6.1 Generieke data

6.1.1 Omschrijving risico

Het is belangrijk dat de data die geleverd word voor het onderzoek goed is. De kans dat corrupte data optreed is altijd aanwezig, al zal het niet regelmatig voorkomen. Toch is corrupte data niet acceptabel. De risico’s zijn dat de ingevoerde gegevens niet klopt en daardoor onbruikbaar is. Dit zijn dan verloren werkuren.

6.1.2 Kans op risico

De kans dat dit optreed is redelijk. Niet alleen kan er informatie foutief worden verzonden, het kan ook door iemand anders zijn verzonden dan de bedoeling is of er kan tijdens het verzenden data verloren gaan.

6.1.3 Impact van risico

De impact van het risico is groot omdat de kaart dan gedeeltelijk of zelfs volledig niet meer zou overeenkomen met de werkelijk. Vrijwilligers zouden aan de hand van de foutieve informatie op pad kunnen gaan om nodes te repareren terwijl dit niet nodig is.

6.1.4 Minimaliseren van risico

Om te zorgen dat de generiekheid van data is gewaarborgd, zijn er enkele dingen die we aan de beveiliging kunnen doen. Ten eerste is er een optie om alle data te normaliseren, we kunnen data encrypten voordat we het versturen en als laatste optie kunnen we een hash van het verstuurde bericht meesturen naar de server.

1. Door data te normaliseren kan je de applicatie zo programmeren dat hij bepaalde waarden verwacht en bepaalde data juist meteen aanvinkt als incorrect of corrupt. Denk hierbij bijvoorbeeld naar de snelheid waarmee een node draadloos opereert. Als deze snelheid volgens de log hoger ligt dan dat technisch haalbaar is, dan kan de data als corrupt worden aangevinkt en niet worden verwerkt. Voor alle data die verstuurd wordt is normaliseren mogelijk en kunnen we op deze manier het risico minimaliseren.

2. Data encrypten met een geheime sleutel is ook een optie. Hierdoor wordt vooral tegen gegaan dat een hacker foutieve data naar de server kan versturen, omdat deze de encryptie en/of de geheime sleutel zou moeten kraken voordat hij een bericht kan produceren dat de server zou accepteren.

3. Als laatste kunnen we een hash maken van het verstuurde bericht. Hiermee verzekeren we vooral dat het bericht tijdens de verzending niet is aangetast. Hierbij is wel van belang dat er gekozen wordt voor een veilige hash, denk hierbij aan de Whirlpool hash die standaard in PHP zit. Deze produceert een key van 128 characters lang.

Pieter Naber (s1054204)
David de Boer (s1013714)

In de huidige situatie zijn het invoeren van het normaliseren van data (optie 1) en het meesturen van een hash (optie 3) de beste opties. Het normaliseren is goed toe te passen gezien de snelheden die bijvoorbeeld een draadloos netwerk maximaal zou kunnen halen. De hash meesturen is een goede garantie dat het bericht niet beschadigd is en is makkelijk toe te passen.

Het encrypten van data zou extra performance vragen van onze simpele applicatie, wat niet wenselijk is gezien het aantal bestanden dat onze applicatie moet gaan verwerken per uur. Ook zou de veiligheid van de encryptie niet gegarandeerd kunnen worden, omdat het gaat om een Open Source project waarbij ook de geheime sleutel bij veel gebruikers in de software zou moeten worden gezet en daardoor makkelijk bekend zou kunnen worden.

6.2 Fysieke beveiliging

6.2.1 Omschrijving risico

het systeem kan op verschillende manieren goed beveiligd worden tegen aanvallen, maar als iemand fysiek toegang kan krijgen tot het systeem, worden al deze beveiligingen omzeild. Daarom is het ook belangrijk om rekening te houden met de risico's hiervan.

6.3 Kans op risico

De kans op dit risico is erg klein. Omdat de stichting wireless Leiden een non-profit organisatie is, zal het ook niet veel kwaadwillenden aantrekken. Mensen zullen ook niet snel fysiek toegang proberen te krijgen tot de servers. Ook heeft de stichting echter geen eigen pand, staan de servers op de Hogeschool en Universiteit Leiden. De ruimtes waar deze servers staan zijn uiteraard niet publiekelijk toegankelijk, maar ze zijn ook niet exclusief toegankelijk voor de stichting wireless Leiden. Ook kunnen er aanvallen gericht worden tot de hogeschool, wat dan een impact kan hebben op het systeem. Deze omstandigheden verhogen de kans op dit risico.

6.3.1 Impact van risico

De impact van een fysieke aanval kan groot zijn, omdat de hele applicatie gekraakt kan worden en alle informatie verwijderd kan worden.

6.3.2 Minimaliseren van risico

Er zijn nu al veel voorzorgsmaatregelen genomen om de servers fysiek te beveiligen en zo de kansen op dit risico zoveel mogelijk te minimaliseren. Zo is er een backup server op een andere locatie als de hoofd server en zijn deze ruimtes afgesloten voor het publiek. Om de kansen op dit risico nog verder te minimaliseren, zouden de servers in ruimtes gezet moeten worden die enkel toegankelijk zijn voor de medewekers van de stichting.

6.4 Beveiligen tegen technische mankementen

6.4.1 Omschrijving risico

Naast alle vormen van hacking zou het ook kunnen dat er door een stroomstoring de server uit valt. Ook zou er, door bijvoorbeeld ouderedom of technische mankementen, hardware kapot kunnen gaan.

6.4.2 Kans op risico

Omdat de servers in betrouwbare serverruimtes staan, zullen er niet snel stroomstoringen optreden. De kans hierop is daarom ook klein. De kans dat er hardware stuk gaat is echter meer aanwezig.

6.4.3 Impact van risico

De impact dat dit gebeurd is redelijk, omdat de applicatie niet meer werkt zonder stroom of werkend systeem, hierdoor zullen er geen gegevens beschikbaar zijn. Als er hardware kapot gaat, zal de server ook niet meer beschikbaar zijn tot dit probleem verholpen word. Dat eventueel voor een lange downtime zou kunnen zorgen.

6.4.4 Minimaliseren van risico

Een backup server, geplaatst in een ander data centrum op een andere locatie zou de oplossing bieden voor eventuele serveruitvallen. Als de ene server niet meer bereikbaar is, dan neemt de andere het over. Hierdoor zal het systeem ten aller tijde bereikbaar zijn. Om ervoor te zorgen dat het uitgevallen systeem, na een hardware fout, spoedig weer in gebruik genomen kan worden, zal er een reserve systeem aanwezig moeten zijn. Hier kunnen dan onderdelen uit worden gehaald, die de hardware kan vervangen op het uitgevallen systeem. Vervolgens kan dit onderdeel opnieuw besteld worden en in het reservesysteem worden geplaats. Zo zijn er altijd reserveonderdelen beschikbaar en zal de downtime altijd zo laag mogelijk zijn.

6.5 Databeveiliging bij de gegevens

6.5.1 Omschrijving risico

De data folder zelf zou gehackt kunnen worden, direct op de server. Hierdoor kan de hacker toegang krijgen tot de data en deze eventueel aanpassen of bestanden toevoegen.

6.5.2 Kans op risico

De kans dat dit gebeurd is klein, gezien het feit dat het om een stichting gaat en hierdoor niet meteen het doelpunt zal zijn van hackers.

6.5.3 Impact van risico

De impact is groot, onze hele applicatie is afhankelijk van de data die wordt opgeslagen in een folder. Door deze aan te passen zou het kunnen voorkomen dat een vrijwilliger eropuit gaat om een node te maken, terwijl er helemaal niets aan de hand is.

6.5.4 Minimaliseren van risico

Om het risico te minimaliseren zijn een aantal dingen belangrijk om rekening mee te houden. Het besturingssysteem van de server is belangrijk en de rechten die aan de map worden gegeven zijn belangrijk.

1. Het besturingssysteem van de server is erg belangrijk. Niet alleen de keuze voor het systeem in het begin is belangrijk, maar ook het up-to-date houden van het besturingssysteem en het eventueel zelf installeren van beveiligingspatches is cruciaal om de server veilig te houden.

2. De rechten die aan de map worden gegeven zijn ook belangrijk. Onze applicatie moet data kunnen schrijven naar de data folder, maar andere gebruikers en/of gewone bezoekers moeten deze rechten natuurlijk niet hebben.

Punt 1 (besturingssysteem up-to-date houden) wordt nu al meegenomen bij Stichting Wireless Leiden, dit gebeurd namelijk door de beheerders. Punt 2 (rechten van de map) is erg belangrijk om mee te nemen bij ons programma. Ook zullen we dit goed moeten testen.

6.6 Uiteindelijke toepassingen in de software

Stichting Wireless Leiden heeft dingen als fysieke beveiliging al goed op orde. Dit bleek ook uit verschillende gesprekken met de opdrachtgever. Mede daarom hebben we niets kunnen doen met fysieke beveiliging, beveiligen tegen technische mankementen en databeveiliging bij de gegevens.

Maar we hebben wel veel aandacht besteed aan de generiekheid van data. Het status bestand wordt per regel gecontroleerd op de gegevens die erin staan en of dit klopt met de gegevens die erin zouden moeten staan. Het locatie bestand wordt niet per regel gecontroleerd omdat er veel verschillende data in verschillende vormen in kan staan. Wel wordt alles wat we uitlezen gecontroleerd.

Pieter Naber (s1054204)
David de Boer (s1013714)

7 Procesbeveiliging

Het is belangrijk om bij de verschillende processen te kijken wat de beveiligingsrisico's zijn. Als de risico's hiervan niet goed onderzocht zijn, zouden er grote beveiligingsproblemen kunnen ontstaan. Ook kunnen goed doorgedachte processen de algemene kwaliteit van het product verhogen.

7.1 Binnenhalen gegevens

7.1.1 Omschrijving risico

Het proces voor het binnenhalen van de informatie moet goed geprogrammeerd zijn. Het risico is dat er, door bijvoorbeeld een verbindingsfout met een node, er wel data verzonden kan worden, maar niet meer goed ontvangen kan worden, waardoor er een loop van requests ontstaat.

7.1.2 Kans van risico

De kans dat dit gebeurd is middel, gezien het feit dat het een netwerk is dat steeds verder wordt ontwikkeld. Hierdoor zouden verbindingsfouten kunnen ontstaan waardoor de verzonden data corrupt wordt.

7.1.3 Impact van risico

De impact van dit punt is middel, omdat het niet ontvangen van data zou kunnen betekenen dat een vrijwilliger de node gaat repareren terwijl de fout ergens anders in het netwerk zit. Dit is natuurlijk iets dat de stichting niet zou willen, een onnodige extra belasting voor de vrijwilligers.

7.1.4 Risico minimaliseren

Om het risico te minimaliseren zijn er enkele opties die we kunnen gebruiken. Sommige van deze opties zijn al besproken in voorgaande verslagen. Deze opties zullen we niet opnieuw volledig gaan bespreken, maar we zullen verwijzen naar voorgaande verslagen.

1. Als eerste optie noemen we nogmaals het normaliseren van data en het meesturen van een hash van het verstuurde bericht. Meer informatie hierover is te vinden in het "Data Beveiligingsplan". Dit zorgt dat we in ieder geval kunnen herkennen of de data die verstuurd wordt corrupt is of nog gewoon te gebruiken voor onze applicatie.

2. Als de server een request om informatie verstuurd naar de nodes, dan kunnen we het volgende proces beschrijven:

- De server verstuurt om de 60 minuten een request naar een node:
    - Als er een corrupt bericht wordt ontvangen:
        - De server verstuurt direct een nieuwe request gestuurd.
        - Als na 3 requests nog steeds een corrupt bericht van de node wordt ontvangen, wordt de node aangemerkt als offline met een melding dat er corrupte berichten zijn ontvangen.
        - De server gaat verder met het normale patroon van requests versturen om de 60 minuten om het netwerk niet onnodig te belasten.
    - Als er geen bericht wordt ontvangen:
        - De server verstuurt na 5 minuten een nieuwe request.
        - Als na 3 requests nog steeds geen bericht van de node is ontvangen, wordt de node aangemerkt als offline met een melding dat er geen berichten zijn ontvangen.

        - De server gaat verder met het normale patroon van requests versturen om de 60 minuten om het netwerk niet onnodig te belasten.
    - Als er een bericht wordt ontvangen:
        - Het bericht wordt verwerkt.
        - De server gaat verder met het normale patroon van requests versturen om de 60 minuten.

3. Als de server een request om informatie verstuurd naar de nodes, maar we het netwerk niet extra willen belasten met extra requests voor als een node niet (goed) reageert, kan optie 2 ook worden aangepast met de wijziging dat er meteen na het niet ontvangen van een bericht of het ontvangen van een corrupt bericht de node wordt aangemerkt als offline.

4. Als de node een bericht stuurt naar de server, dan kunnen we het volgende proces beschrijven:

- Zodra de node online komt stuurt hij een bericht naar de server met zijn naam en locatie.
- De server stuurt een welkomstbericht terug en geeft de node toestemming om data naar de server te sturen.
- De node verstuurt om de 60 minuten een bericht naar een server:
    - Als de server een corrupt bericht ontvangt:
        - De server verstuurt direct een nieuwe request naar de node.
        - Als na 3 requests nog steeds een corrupt bericht van de node wordt ontvangen, wordt de node aangemerkt als offline met een melding dat er corrupte berichten zijn ontvangen.
    - Als er geen bericht wordt ontvangen:
        - De server verstuurt na 5 minuten een nieuwe request.
        - Als na 3 requests nog steeds geen bericht van de node is ontvangen, wordt de node aangemerkt als offline met een melding dat er geen berichten zijn ontvangen.
    - Als er een bericht wordt ontvangen:
        - Het bericht wordt verwerkt.

5. Als de node een bericht stuurt naar de server, maar we het netwerk en het programma niet extra willen belasten met requests van de server, kunnen we optie 4 ook aanpassen door het versturen van requests te verwijderen en een node direct aan te merken als offline.

6. Als laatste zouden we ook gebruik kunnen maken van routing binnen het netwerk. De nodes zijn altijd verbonden door middel van meerdere links. Mocht een route niet werken, dan zou of de server (zoals bij voorgaande opties) of de node (zoals bij voorgaande opties) kunnen kiezen voor een alternatieve route voor het bericht en/of de request.

Als we deze 6 opties meenemen willen we vooral de schaalbaarheid van de applicatie in acht houden. Daarom is het niet wenselijk om elke 5 minuten berichten te moeten sturen naar de server en/of nodes die niet (goed) reageren. Optie 5 lijkt daarom ideaal, omdat hier het minst aantal berichten wordt verstuurd.

Daarnaast zien we in dit project af van optie 6, de routing, omdat we hier niet mee te maken krijgen en niet haalbaar is in dit stadium. Wel zou dit meegenomen kunnen worden in latere versies.

Wel gaan we gebruik maken van optie 1, om de integriteit van de data te kunnen waarborgen, daarnaast is dit punt ook noodzakelijk om bij punt 5 corrupte berichten te herkennen.

7.2 Up-to-date gegevens

7.2.1 Omschrijving risico

Gebruikers moeten de laatste versies van de KML bestanden uitlezen via de applicatie, anders kijken ze naar oude informatie die niet meer up-to-date is, met de kans dat ze denken dat er ergens netwerk zou moeten zijn terwijl de node offline is of andersom.

7.2.2 Kans van risico

De kans dat dit gebeurd is groot. Alle populaire browser types werken met caching van gegevens, wat dit risico vergroot. Vaak is er wel een controle vanuit de browser of de data niet is aangepast sinds een laatste bezoek, maar dit werkt soms slecht met via HTML ingevoegde bestanden.

7.2.3 Impact van risico

De impact is middel, omdat het kan betekenen dat er meerdere vrijwilligers reageren op een storing, maar dit ook gewoon kan worden opgelost door goede communicatie tussen vrijwilligers en het goed vernieuwen van de pagina.

7.2.4 Risico minimaliseren

Om dit risico te minimaliseren kunnen we enkele opties bedenken:

1. Het simpelste is natuurlijk om niets te doen en te verwachten dat gebruikers die up-to-date informatie willen de pagina wel goed vernieuwen en eventueel hun cache legen.

2. We kunnen gebruik maken van unieke namen voor de KML bestanden die we gaan gebruiken. Via een simpel PHP script verwijzen we de gebruiker altijd door naar het laatste KML bestand dat is aangemaakt. In het PHP script zorgen we ook dat het script niet gecached wordt door de browsers. PHP script stuurt de client dus door naar laatste KML bestand.

3. Als derde optie kunnen we ook gebruik maken van een KML bestand, maar zorgen dat wordt aangeroepen in combinatie met een timestamp, waardoor de browser deze altijd opnieuw laadt. Te denken valt aan bestand.kml?date=23:44:12,23-03-2010.

4. Datum in KML bestand, die op kaart te zien is.

Optie 3 heeft de voorkeur omdat dit zorgt dat het KML bestand dat up-to-date is altijd een vaste naam heeft en direct kan worden aangeroepen. Daarnaast is het wel wenselijk dat de data altijd up-to-date is.

7.3 Offline raken kaarten software

7.3.1 Omschrijving risico

Als de kaarten software niet bereikbaar is, zal ook de kaart van de node map niet werken. Het zal ook lijken alsof de node map zelf offline is, wat niet goed is voor de naam wireless leiden.

Pieter Naber (s1054204)
David de Boer (s1013714)

7.3.2 Kans van risico
De verschillende kaarten sites hebben een goede bereikbaarheid. Op de lange termijn is er echter nog de mogelijkheid dat de kaart site stopt met het aanbieden van zijn diensten. Dit is niet te voorspellen en daarom is de kans op dit risico middel.

7.3.3 Impact van risico
De impact van dit risico is groot, omdat de applicatie niet meer zou werken. De code zou opnieuw geschreven moeten worden, wat voor extra kosten en downtime zal zorgen.

7.3.4 Risico minimaliseren
Om dit risico te minimaliseren zijn er een aantal oplossingen. Uit voorzorg kan er gekozen worden voor de grootste aanbieder. Deze heeft de financiën om zijn product online te houden. Deze waarborgt niet alleen een hoge up time voor de korte termijn, maar ook op de lange termijn. Daarom gaat de voorkeur uit naar Google Maps.

Daarnaast kan er, met behulp van KML bestanden, makkelijk van kaart veranderd worden. Daarom zal er ook een backup map aanwezig zijn. Als de kaart van Google het niet meer doet, dan zal de applicatie een andere kaart gebruiken. Dit zal de gebruiker, los van het andere ontwerp van de kaart, niet merken.

Mocht de backup map ook offline gaan, dan zorgen de KML bestanden ook voor een gemakkelijke overgang naar een andere kaart. Wat de eventuele kosten, voor het aanpassen van de code, zal drukken.

Een andere oplossing zou zijn om een eigen kaart systeem te maken. Er is namelijk alleen maar een plattegrond van Leiden en omstreken nodig. De kaart zal dan altijd bereikbaar zijn, als de site ook online is. Het nadeel is echter dat de ontwikkelingskosten van de applicatie dan een stuk hoger zouden liggen. Ook zal deze applicatie van een minder hoge kwaliteit zijn, omdat de grotere bedrijven meer tijd hieraan kunnen besteden. Het word daarom ook niet aangeraden om dit te doen.

7.4 Error handling

7.4.1 Omschrijving risico
Als er een fout optreed in de applicatie, dan krijgt de gebruiker een foutmelding in zijn browser. Dit ziet er niet professioneel uit, maar daarnaast heeft de gebruiker ook niets aan de foutmelding. Ook ben je als beheerder niet op de hoogte van deze waarschuwing, waardoor je eventuele fouten niet kunt verhelpen.

7.4.2 Kans van risico
De kans op dit risico is middel. Het is namelijk altijd mogelijk dat er ergens iets niet goed bereikbaar is. Hierdoor zou je verschillende foutmeldingen kunnen krijgen.

7.4.3 Impact van risico
Gebruiksvriendelijkheid is uiteraard belangrijk, maar het beïnvloed niet de werking van de applicatie. Zonder correcte error behandeling zouden eventuele fouten echter niet goed kunnen worden opgespoord. Daarom is de impact van dit risico middel.

Pieter Naber (s1054204)
David de Boer (s1013714)

7.4.4 Risico minimaliseren

Om de risico hierop te minimaliseren, zullen er zelfgemaakte foutmeldingpagina's gemaakt moeten worden. Hierin komt alleen een bericht te staan dat er iets is fout gegaan en dat de beheerders hiervan op de hoogte zijn gesteld. Op de achtergrond word vervolgens een bericht gestuurd, naar de beheerder, met de foutmelding in kwestie.

De gebruiker krijgt zo geen vreemde foutmeldingen te zien, waardoor het ontwerp van de site gewaarborgd blijft. Ook zijn de beheerders altijd op de hoogte van de foutmeldingen, die de de gebruikers krijgen. Om ervoor te zorgen dat er geen overbodige foutmeldingen komen, zullen er geen dubbele foutmeldingen verstuurd worden van dezelfde gebruiker.

Voor het behandelen van de foutmeldingen zijn er verschillende mogelijkheden.

Error e-mailen naar beheerders. Het voordeel hiervan is dat de beheerders direct op de hoogte zijn van een eventuele foutmelding. Het nadeel is dat de beheerders bij een fout misschien een groot aantal mails toegestuurd krijgt. Ook is de beheerder niet op hoogte van de genomen handelingen van de gebruiker. Wat bij sommige foutmeldingen van pas kan komen.

Error opslaan in log bestand. Her voordeel hiervan is dat de beheerders niet overspoeld worden met foutmeldingen van gebruikers. Het nadeel is echter dat de beheerders niet direct op de hoogte zijn van een foutmelding. Dit zou verholpen kunnen worden als er, bij een wijziging aan het logbestand, een mail word verstuurd en dat er vervolgens elk uur een nieuwe update word gemaild. Ook hier zijn de beheerders niet op de hoogte van de handelingen van de gebruikers.

Alle handelingen opslaan in log bestand, en daarbij ook error. Het voordeel hiervan is dat de beheerders beter fouten kunnen verhelpen als ze de handelingen van de gebruikers weten. Het nadeel is echter dat het log bestand teveel informatie bevat, waardoor het systeem dicht slipt. Ook kan het logbestand dan onoverzichtelijk worden. Dit kan verholpen worden door alleen de noodzakelijke handelingen op te slaan.

Niets doen met errors. De meeste foutmeldingen zullen te maken hebben met het niet bereikbaar zijn een node. Deze foutmeldingen zijn daarom waarschijnlijk van korte duur. Daarom zou er voor gekozen kunnen worden om niks met de foutmeldingen te doen. De beheerders zijn dan niet op de hoogte van de opgetreden fout, waardoor ze de fout minder snel verhelpen.

Errors direct weergeven in kaart. Op deze manier zal het ontwerp van de site intact gehouden worden en word de gebruiker toch op de hoogte gesteld van de eventuele situatie. Deze zal alleen wel moeten begrijpen wat de betreffende error inhoud. Ook hier worden de beheerders niet op de hoogte gebracht van een error.

Errors in kaart weergeven door midden van kleuren. Zoals eerder al vermeld is, zullen de meeste foutmeldingen te maken hebben met het niet kunnen bereiken van een node. Om toch feedback te geven aan de gebruiker over de fout, kan er met behulp van een kleur weergegeven worden of de node correct werkt. Als deze groen is, dan werkt alles goed. Als de node rood gekleurd is, dan is er fout opgetreden en werkt de node dus niet goed. Op deze manier krijgt de gebruiker toch enige feedback. De beheerders worden niet op de hoogte gesteld.

Pieter Naber (s1054204)
David de Boer (s1013714)

Voor deze applicatie lijkt het ons het best om de gebruikers op de hoogte te stellen door middel van het kleuren van de nodes. De beheerders kunnen deze foutmeldingen lezen in een log bestand, waar ook de handelingen van de gebruiker worden opgeslagen bij het optreden van de fout. Bij het eerste optreden van een fout word er een mail gestuurd naar de beheerders. Elk uur krijgen ze een update, waarin staat hoeveel foutmeldingen er zijn geweest en bij welke node deze foutmeldingen zich voordeden.

Om te voorkomen dat het systeem vol komt te staan met logbestanden, zullen meldingen die ouder zijn dan 1 maand worden overgezet naar een archief. Het archief word elk jaar verwijderd, hierover krijgen de beheerders een mail. Hierin moeten ze deze actie goedkeuren.

7.5 Uiteindelijke toepassingen in de software

Het binnenhalen van de gegevens gaat voor onze applicatie uiteindelijk niet per node, maar als geheel. We doen dit zodra onze applicatie wordt aangeroepen, maar cachen wel onze KML bestanden.

Offline raken van landkaarten software hebben we niet geïmplementeerd, zie hiervoor het vorige hoofdstuk. Error handling hebben we uitgebreid geïmplementeerd, hierover meer in het volgende hoofdstuk.

Pieter Naber (s1054204)
David de Boer (s1013714)

8 Applicatiebeveiliging

Bij applicatie beveiliging worden de verschillende beveiligingsrisico's besproken, die van toepassing zijn op de applicatie. Zo zal de code goed getest moeten worden, om gaten in de applicatie te voorkomen. Ook zullen de risico's besproken worden, die het uitgeven via open source met zich mee brengt. Als alle voorzorgsmaatregelen genomen zijn, dan is de kans altijd aanwezig dat er alsnog iets fout gaat. Daarom worden ook de logbestanden besproken in dit verslag.

8.1 Testen van code

8.1.1 Benoemen van risico

De code moet goed worden opgeleverd, dit betekent dat we het maken van fouten in de code willen minimaliseren en dat we de code zorgvuldig en uitvoerig willen testen. Door fouten in de code kan de applicatie zijn werk niet goed doen of door nog slechtere code kan zelfs de server in het gevaar komen.

8.1.2 Kans op risico

De kans dat dit gebeurd is redelijk. Programmeurs maken nou eenmaal fouten als ze code aan het schrijven zijn. Van kleine typefouten die snel aangepast kunnen worden, maar ook grotere die de applicatie in gevaar brengen.

8.1.3 Impact van risico

De impact is groot, omdat slechte code (beperkte) toegang kan geven op de server waar naast onze applicatie nog meer applicaties draaien.

8.1.4 Risico minimaliseren

Binnen de groep zijn enkele dingen besproken om het risico te minimaliseren en te garanderen dat we de code goed testen:

1. Opzetten van een test plan. Door gebruik te maken van de methode van TestGoal willen we voor onze applicatie een logisch en een fysiek testplan opzetten waarmee we de code later kunnen testen.

2. Tijdens het programmeren gebruik maken van "Eclipse for PHP Developers". Door deze applicatie te gebruiken wordt de code tijdens het schrijven van de code al gecontroleerd op syntax fouten. Hierdoor kunnen we het aantal fouten dat we maken minimaliseren.

3. Code testen op een lokale computer in plaats van de server zelf. De code zal tijdens de ontwikkelingsfase alleen lokaal gedraaid worden en dus niet op de server zelf. Hierdoor minimaliseren we de risico's totdat de code goed getest is.

4. Elkaars code controleren. Omdat we samenwerken in een groep van 4 studenten zullen we elkaars code kunnen controleren op fouten. Hierdoor worden fouten in de code sneller gevonden. 4 paar ogen zien meer dan 1 paar...

Moet bug tracking centralisatie hier ook niet in staan? -Rick van der Zwet 3/31/10 10:01 PM

Voor ons project zullen we gebruik maken van alle 4 de opties die we opperen omdat ze goed zijn te gebruiken bij dit huidige project.

8.2 Log bestanden

8.2.1 Benoemen van risico

Er kan altijd wat gebeuren wat niet de bedoeling is binnen de applicatie. Door corrupte input die door de applicatie wel als goed wordt herkend of door een kwaadwillend persoon (hacker). Het is belangrijk om tijdens of na een fout te weten wat er is gebeurd binnen de applicatie waardoor het fout ging.

8.2.2 Kans op risico

Na alle tests en beveiligingen die we hebben uitgevoerd is de kans dat er toch iets fout gaat klein.

8.2.3 Impact van risico

Als het systeem niet weet wat er fout ging, kan dezelfde fout vaker voorkomen. Ook als een kwaadwillend persoon iets heeft gedaan wat niet de bedoeling is en als dit niet wordt opgemerkt kan deze persoon de actie blijven herhalen met alle gevolgen van dien.

8.2.4 Risico minimaliseren

We gaan om goed bij te houden wat er gebeurd een log bestand bijhouden. We maken gebruik van de volgende levels van logging:

- LOG_EMERG - Systeem is niet bruikbaar meer
- LOG_ALERT - Actie is meteen nodig
- LOG_CRIT - Een kritieke toestand
- LOG_ERR - Een error heeft zich voorgedaan
- LOG_WARNING - Een waarschuwing heeft zich voorgedaan
- LOG_NOTICE - Een normale situatie, maar wel opmerkelijk
- LOG_INFO - Normale informatie
- LOG_DEBUG - Extra informatie voor debugging van de code

Deze levels komen overeen met SYSLOG, een veelgebruikt systeem voor het loggen bij applicaties.

De applicatie kan ingesteld worden op een bepaald niveau, logs van dat niveau en erboven worden gelogd, de rest wordt genegeerd. Tijdens het programmeren kan bijvoorbeeld de logging op LOG_DEBUG gezet worden, waardoor alle logdata wordt opgeslagen. Als de applicatie daadwerkelijk draait kan gekozen worden om het niveau te verhogen naar LOG_NOTICE om alleen de belangrijke log data op te slaan.

De log data wordt opgeslagen in een log bestand op de server die openbaar is voor alle gebruikers, maar vooral door beheerders gebruikt zal worden.

8.3 Open Source

8.3.1 Benoemen van risico

De code zal via een opensource licentie openbaar beschikbaar worden gesteld. De code kan dan worden bestudeerd op eventuele gaten en er kan zo toegang worden verkregen tot het systeem.

8.3.2 Kans op risico
De kans dat dit gebeurd is klein. Omdat het een open source project is, is de code openbaar toegankelijk. Dit verhoogt het risico, in vergelijking met closed source. Wel vind je potentieel meer fouten door gebruikers en geïnteresseerden die de code bekijken.

8.3.3 Impact van risico
Omdat iemand toegang kan krijgen tot het systeem, is de impact van dit risico groot.

8.3.4 Risico minimaliseren
Om te voorkomen dat er gaten in de code zitten, zal de code goed gecontroleerd moeten worden. Ook zullen er geen wachtwoorden en dergelijke in de code te vinden zijn. Zo zal de code goed beschermd zijn tegen hedendaagse beveiligingslekken. Om ervoor te zorgen dat de code ook goed blijft, zal het in de toekomst eventueel nog bijgewerkt moeten worden.

Ook zou er voor gekozen kunnen worden om het project niet openbaar uit te brengen. Dit zou de kans op het risico aanzienlijk verlagen. Omdat de opdrachtgever heeft aangegeven de applicatie Open Source te willen uitbrengen, zal dit echter niet gebeuren.

8.4 Uiteindelijke toepassingen in de software
Het testen van de code hebben we uiteindelijk door tijdgebrek niet kunnen uitvoeren. Wel is er een logisch testplan geschreven. Tests zouden later door de opdrachtgever of door een volgende project groep kunnen worden meegenomen.

Error handling en het wegschrijven naar een log bestand is volledig geïmplementeerd. De beheerder kan een niveau instellen wanneer er gemaild moet worden, een niveau wanneer de melding moet worden weggeschreven naar het logbestand en een niveau wanneer het als commentaar in het HTML of KML bestand moet worden geplaatst. De gebruikers merkt hier dus niets van.

Pieter Naber (s1054204)
David de Boer (s1013714)

9 Risico's uitgewerkt

Om de "formule" "Kans x Impact = Risico" beter in beeld te brengen zullen we dit doen in een tabel, waarbij al snel duidelijk wordt welke risico's het grootst zijn en zeker zullen moeten worden aangepakt.

9.1 Score per omschrijving

Rekenen met woorden gaat lastig, daarom gebruiken we de volgende punten verdeling:

Omschrijving:	Score:
Klein	1
Middel	2
Redelijk	3
Groot	4

9.2 Risico tabel

Beschrijving	Categorie	Kans	x	Impact	=	Risico
5.1. (Distributed) Denial-Of-Service	Netwerkbeveiliging	1	x	1	=	1
5.2. Uitval netwerk	Netwerkbeveiliging	1	x	1	=	1
5.3. Afhankelijkheid Google Maps	Netwerkbeveiliging	3	x	5	=	15
5.4. Afluisteren van netwerk	Netwerkbeveiliging	1	x	1	=	1
5.5. Onderschepping van data	Netwerkbeveiliging	1	x	2	=	2
5.6. Snooping	Netwerkbeveiliging	1	x	1	=	1
5.7. Analyse en traffic	Netwerkbeveiliging	1	x	1	=	1
6.1. Generieke data	Databeveiliging	3	x	4	=	12
6.2. Fysieke beveiliging	Databeveiliging	1	x	4	=	4
6.3. Beveiligen tegen technische mankementen	Databeveiliging	1	x	3	=	3
6.4. Databeveiliging bij database	Databeveiliging	1	x	4	=	4
7.1. Binnenhalen gegevens	Procesbeveiliging	2	x	2	=	4
7.2. Up-to-date gegevens	Procesbeveiliging	2	x	2	=	4
7.3. Offline raken kaarten software	Procesbeveiliging	2	x	4	=	8
7.4. Error handling	Procesbeveiliging	2	x	2	=	4
8.1. Testen van code	Applicatiebeveiliging	3	x	4	=	12
8.2. Log bestanden	Applicatiebeveiliging	1	x	4	=	4
8.4. Open Source	Applicatiebeveiliging	1	x	4	=	8

Pieter Naber (s1054204)
David de Boer (s1013714)