Afstudeerverslag

Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie bij Technolution BV. Gouda

Bart de Langen
15077071
HBO-ICT Network & System Engineering
De Haagse Hogeschool Delft
Versie 1.0
26 maar 2021

Voorwoord

Dit verslag bevat de procesomschrijving van de afstudeerstage die uitgevoerd werd in opdracht van Technolution BV. Gouda. In deze opdracht werd een hardware VPN-module ontwikkeld. Dit document omschrijft het proces dat doorgelopen is en de gemaakte keuzes die uiteindelijk tot het eindproduct hebben geleid.

Binnen Technolution wil ik graag een aantal mensen bedanken voor hun hulp en ondersteuning tijdens het project. Allereerst wil ik Roy Rietveld bedanken voor zijn betrokkenheid en geduld. Hij heeft mij gedurende het volledige afstudeerproject heeft geholpen, en tips heeft gegeven. Zelf als ik het door bomen het bos niet meer zag. Ik wil ook Jonathan Hofman bedanken voor de mogelijkheid om mijn afstudeertraject te mogen lopen bij Technolution.

Van de Haagse hogeschool wil ik graag de heren Hans de Vreught en Marinus Maris bedanken voor de hulp en begeleiding. Niet alleen tijdens het afstudeertraject, maar ook voor de lessen in de studiejaren daarvoor.

Als laatste wil ik een aantal studiegenoten bedanken: Willem Vos, Ruben van den Engel, en mijn vrienden in de studiegroep B&B: bedankt voor onze studenten tijd. Zonder jullie hulp en support was mijn studietijd een stuk saaier geweest.

Gouda, 1 maart 2020

Bart de Langen

Referaat

Er is steeds meer behoefte om data veilig tot in het end-point apparaat te brengen. Ook wel zero-trust networking genoemd. Daarnaast groeien de dreigingen. Zo geven de recente kwetsbaarheden in de hardware van pc-platformen voldoende aanleiding om niet meer volledig op de software-scheiding van besturingssystemen te kunnen vertrouwen. Om deze reden is er in opdracht van Technolution een proof of concept VPN-oplossing ontwikkeld die een hardware matige scheiding maakt tussen VPN en computer. Voor de ontwikkeling hiervan werd een USB Armory als platform gebruikt.

Tijdens het project werd er een hardware matige VPN-demonstratieproduct ontwikkeld. Hierbij is er een analyse gemaakt wat de hardware mogelijkheden van het platform waren. Hierna zijn de requirements opgesteld aan de hand interviews met de opdrachtgever. Door middel van de RAD methodiek is er iteratief een programma geschreven.

Aan het einde van het project is het proof of concept succesvol afgerond. Het kon op veilige wijze een VPN-verbinding opzetten die gedeeld kon worden met de host PC. Met python is de onderliggende code ontwikkeld die verantwoordelijk is voor het opzetten van de OpenVPN-NL tunnel, en het doorvoeren van systeemwijzigingen. Hierbij waar mogelijk gebruik gemaakt van bestaande packages. Als deze niet bestonden waren hier eigen libraries voor ontwikkeld.

Aanbevolen wordt voor de doorontwikkeling van de VPN-module een ander platform dan een USB Armory te gebruiken. De USB Armory bevat niet voldoende capaciteiten om op hogere snelheden een VPN-tunnel te onthouden. Ook essentiële zaken zoals een ingebouwde Wi-Fi module zijn niet te vinden in de USB Armory waardoor andere platformen beschikbaar.

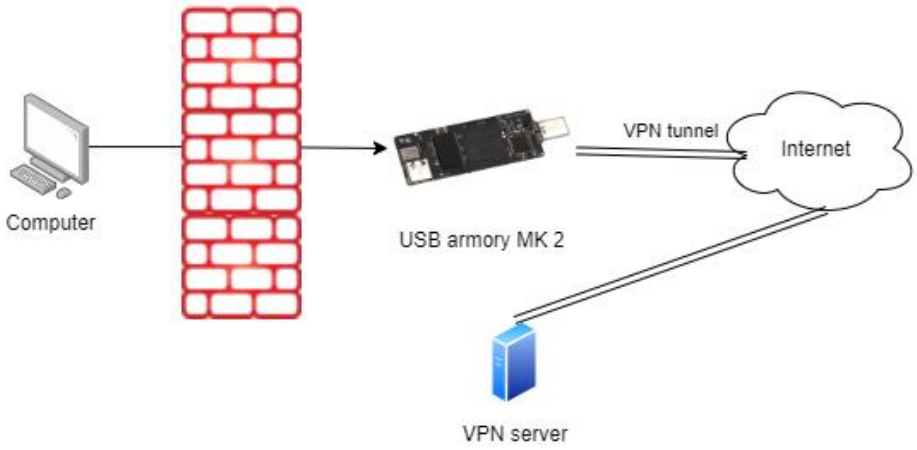
Lijst met figureren

Inhoudsopgave

1 Inleiding

Technolution is een technologie-integrator. Ze focussen zich op het ontwikkelen van gespecialiseerde software en product oplossingen voor bedrijven en overheden. Hierbij ontwikkelen zij zelf de elektronica, programmeerbare logica, embedded software en technische informatiesystemen.

Technolution ziet steeds meer behoefte bij het ontwikkelen van deze producten om data veilig tot in het end-point apparaat te brengen, ook wel zero-trust networking genoemd. Recente kwetsbaarheden in de hardware van de pc-platformen geeft voldoende aanleiding om niet meer volledig op de softwarescheiding van het besturingssystem te kunnen vertrouwen. Traditionelen VPN-oplossingen bieden niet meer voldoende bescherming tegen hoge dreigingen. Dit geeft Technolution voldoende aanleiding om een hardware geïsoleerde VPN-oplossing te creëren.



Figuur 1-1 visuele omschrijving voor

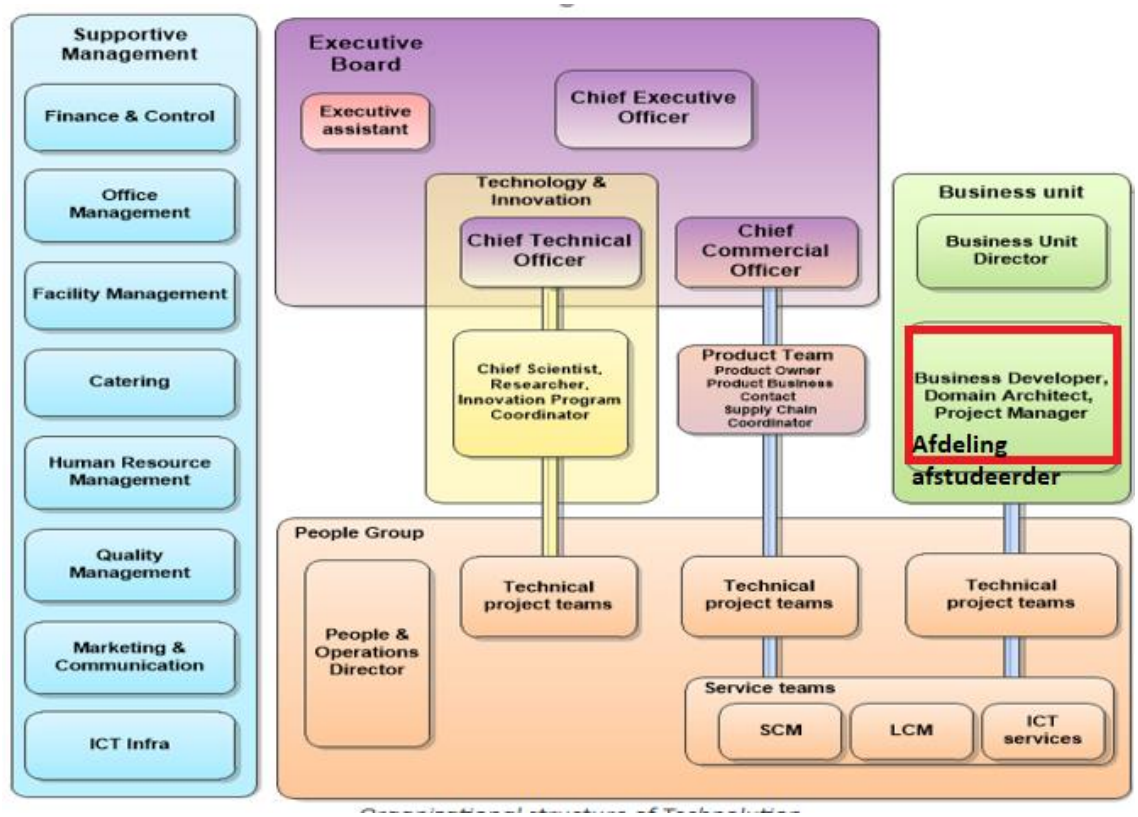
Het doel het deze opdracht is dan ook om een hardware VPN-module te creëren die een scheiding/muur creëert tussen het “onveilige internet” en een veilige VPN-verbinding. Het eindpunt hoeft op deze manier nooit blootgesteld te worden aan het internet. Dit zorgt voor een veilige verbinding.

In dit verslag is te lezen hoe het project en de iteraties (met behulp van de Rapid Application Development (RAD)) zijn verlopen. De specifieke opdrachtomschrijving en de organisatie komen in latere hoofdstukken aan bod.

2 Organisatie omschrijving

Technolution is een middelgroot bedrijf met ongeveer 230 medewerkers. Het bedrijf is in bezit van de oprichters en werknemers. Ze focussen zich op het leveren van gespecialiseerde software en productoplossingen. Hierbij vallen de afdelingen onder 4 categorieën in te delen.

- Executive board, bestaande uit management die de bedrijfsrichting bepaald.
- Technology & Innovation, investeert in R&D en de kennis in het bedrijf
- Business unit, voert de opdrachten voor klanten uit.
- People group, is verantwoordelijk is voor het verkrijgen, co-ordineren, en managen van de opdrachten (alle contact tussen, business unit en de klant)
- Supportive management, services voor het functioneren van bedrijf, HR, en catering.



Figuur 2-1 Bedrijfsorganigram Technolution (bron: technolution intranet)

Projecten bij Technolution worden ingedeeld onder, mobility Solutions, energy solutions, High tech & Big science, manufacturing solutions, en high-Assurance solutions. Ze hebben zowel bedrijven als overheden als klant. Sommige producten die in opdracht van een klant ontwikkeld waren zijn hierna als Technolution product ook aan andere klanten aangeboden. Hierdoor voert Technolution zowel externe projecten uit voor klanten en intern projecten om deze producten door te ontwikkelen.

De opdracht wordt uitgevoerd binnen de afdeling “Business Development” van “Business unit”. De business unit is verantwoordelijk voor de uitvoer en verkoop van projecten. Mijn opdrachtgever, Jonathan Hofman, is de domain architect voor de business unit. Mijn begeleider, Roy Rietveld, behoord als Architect tot de business unit. Omdat de afstudeeropdracht het ontwikkelen van software omvat is ervoor gekozen om deze in de business unit afdeling te plaatsen.

3 Opdracht

3.1 Project aanleiding

Er is steeds meer behoefte om data veilig tot in het end-point apparaat te brengen. Zo is afgelopen jaren steeds populairder geworden voor medewerkers om hun eigen apparaat te gebruiken voor werk gerelateerde zaken. Hier is een VPN een van de tools die bedrijven gebruiken om ervoor te zorgen dat bedrijfsmatig dataverkeer veilig bij de eindbestemming aankomt. Meestal wordt hiervoor een softwarematige VPN-client voor geïnstalleerd. Hiervoor is deze afstudeeropdracht opgesteld (zie **bijlage A: afstudeeropdracht**).

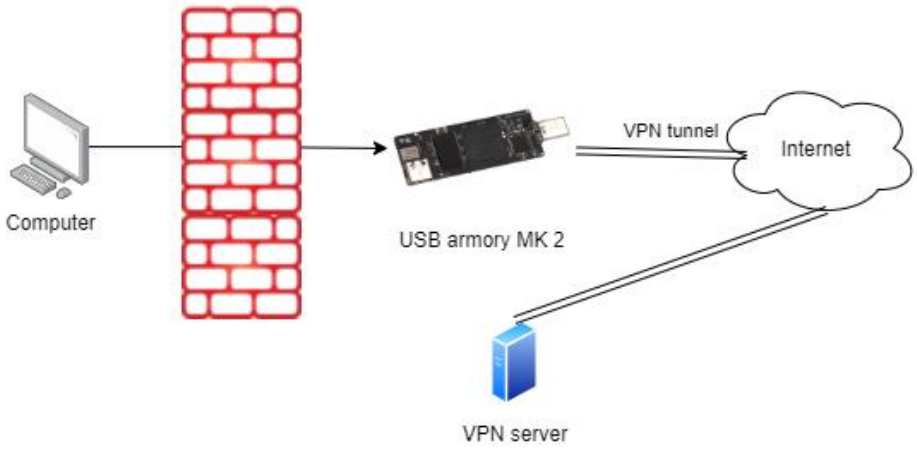
3.2 Probleemstelling

Recente kwetsbaarheden in de hardware van pc-platformen zorgen ervoor dat er niet meer volledig op de software-scheiding van de besturing systemen kan worden vertrouwd. Traditionele software VPN-oplossingen bieden niet meer voldoende weerstand tegen hoge dreigingen. Er is daarom behoefte voor Technolution om te onderzoeken wat de mogelijkheden zijn voor hardware geïsoleerde VPN-oplossingen.

3.3 Doelstelling

Het doel van dit project is het maken van een demonstratieproduct waarbij er een scheiding is aangebracht tussen de computer en het internet. Dit wordt bereikt door middel van een USB Armory die functioneert als VPN-module. De USB Armory vormt een soort muur tussen de computer en het internet.

Zodra de gebruiker de USB Armory in de computer steekt maakt deze verbinding met internet en probeert vervolgens een veilige VPN-tunnel op te zetten. Als de VPN-tunnel opgezet is deelt hij deze verbinding met de computer door middel van Ethernet Emulation. Op deze manier heeft de computer een veilige internetverbinding zonder dat hij verbonden is met het “onveilige internet.”



Figuur 3-1 visualisering doelstelling

3.4 Resultaat

Een proof of concept waarbij een USB Armory MK 2 als platform is gebruikt. Dit demonstreert dat de USB armory als hardware matige VPN-solution kan worden gebruikt om op veilige manier een VPN-tunnel op te zetten. Dit zonder dat de host computerverbinding hoeft te maken met het “onveilige internet”.

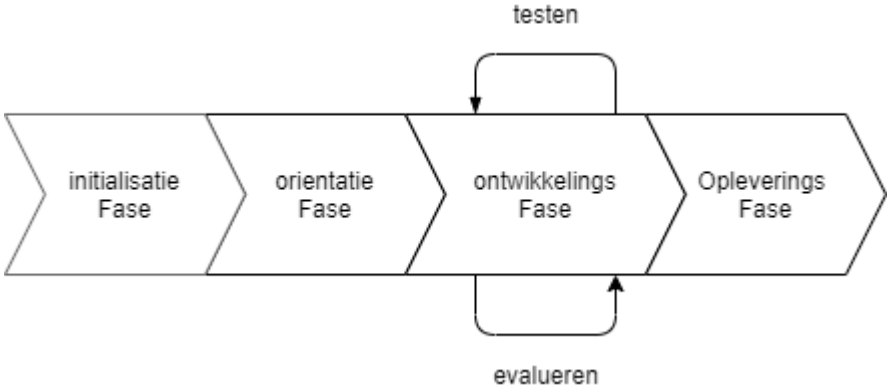
3.5 Gebruikte project methodiek

Om het project structureel in goede banen te leiden en de kwaliteit van de software te bewaken wordt er gewerkt volgens een project methodiek. Tijdens het afstudeertraject wordt er software ontwikkeld. Deze software moet met de hardware (in dit geval de USB Armory) kunnen communiceren. Er is er geen voorkennis over de mogelijkheden van de hardware en communicatiemogelijkheden voor de software. Hierdoor is het belangrijk om voldoende tijd in te plannen voor een vooronderzoek. Ook is het belangrijk om de requirements kunnen stellen als ze niet mogelijk blijken te zijn.

Voor dit project is er gekeken naar vier project methodieken die geschikt leken voor dit project. **Scrum, pair programming**, **RAD**, en de **watervalmethodiek**. Dit project wordt uitgevoerd in de corona-crisis. Er wordt afwisselend vanuit huis en vanaf kantoor gewerkt. Hierdoor ontstaat het risico dat de communicatie niet altijd ideaal verloopt. Om deze reden is het belangrijk bij de gebruikte methodiek ook flexibel gewerkt kon worden.

Voor de projectmethodiek is ervoor gekozen om een aangepaste versie van **RAD** (Rapid Application Development) te gebruiken. In eerste instantie ging de voorkeur uit naar SCRUM omdat ik hier al ervaring mee had via schoolprojecten. Echter omdat dit project door 1 persoon wordt uitgevoerd en er verder geen teamverband is viel deze methodiek af. Zaken als daily standups en SCRUM meetings met één persoon zijn niet nuttig. De watervalmethode is niet geschikt omdat er flexibel geïtereerd moet kunnen worden. Het risico bestaat dat door een miscommunicatie tijdens het project de eisen kunnen veranderen. Als er een wijziging in een voorgaande iteratie plaatsvindt moeten de vorige fases ook opnieuw uitgevoerd worden. Hierdoor valt de waterval methodiek af. Tijdens een contactmoment met de opdrachtgever gaf hij aan dat hij weinig tijd had voor pair programming. Daardoor viel deze methodiek ook af.

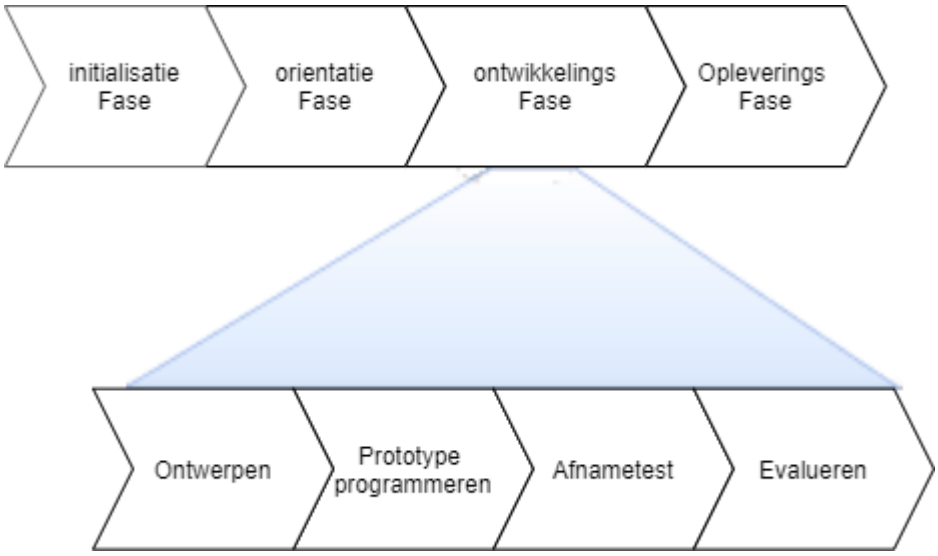
Bij de RAD methodiek is het de bedoeling dat er snel en flexibel werkende software opgeleverd wordt en erdoor geïtereerd wordt op de vorige fase. Bij RAD vindt er eerst een risico en requirement analyse plaats. Omdat er geen voorkennis is wordt er in dit project na de requirement analyse ook een vooronderzoek gedaan. Als de requirement analyse is afgerond wordt er een ontwerp en een prototype gemaakt. Aan de hand van de feedback wordt het ontwerp aangepast en op dit prototype geïtereerd in de volgende fase. Deze methodiek is ideaal voor het project omdat op deze manier snel en flexibel gewerkt kan worden. In dit project is weinig voorkennis, en door de corona lockdown zal er minder contact met de opdrachtgever zijn. Mocht het zo zijn dat er nieuwe kennis opgedaan wordt, of dat de requirements tussentijds veranderen kan er op deze manier in het project flexibel gereageerd worden.



Figuur 3-2 Uitvoering RAD binnen project

In figuur 3-2 is een visualisatie te zien van de RAD methodiek binnen dit project. De aangepaste RAD methodiek in dit project kent vier fases: de initialisatiefase, de oriëntatiefase, de ontwikkelingsfase, en de opleveringsfase. In de initialisatie fase wordt het probleemdomein, de requirements en de visie voor het project vastgelegd. Er wordt een globale planning gemaakt en een risicoanalyse uitgevoerd. De resultaten hiervan worden beschreven in een voorlopig plan van aanpak.

Vervolgens worden in de oriëntatiefase de visie en requirements van het probleemdomein onderzocht. Er vindt een onderzoek plaats naar de mogelijkheden om de visie voor project te kunnen realiseren en of dit überhaupt realistisch is. Deze worden vastgelegd in het analysedocument. De bevindingen worden voorgedragen aan de opdrachtgever, waarna de requirements eventueel aangepast worden. Vervolgens wordt er een globale planning gemaakt, Use-Cases omschreven, en de afbakening van het project vastgelegd. Dit wordt aangepast in het voorlopig plan van aanpak. Dit wordt vervolgens het **definitieve plan van aanpak**.



Figuur 3-3 verloop van de ontwikkelingsfase

De ontwikkelingsfase van de RAD methodiek voldeed niet volledig aan de visie hoe het project zou moeten verlopen. Het was vrij belangrijk dat er duidelijke fases van ontwerpen, programmeren en testen waren. Dit kwam niet voldoende voor in de RAD methodiek. Om deze reden is de ontwikkelingsfase in 4 subfases ingedeeld.

De ontwikkelingsfase omvat de werkelijke ontwikkeling van de applicatie. Deze fase vinden plaats in meerdere iteraties. Aan het begin van de iteratie wordt er één of meer eisen genomen uit de requirements. Er wordt een ontwerp gemaakt voor de realisatie van deze eisen. Hier wordt vervolgens een prototype voor geprogrammeerd. Voor dit prototype vindt een afname test plaats om te demonsteren dat het prototype aan de eisen voldoet. Aan de hand van de verkregen feedback wordt er geëvalueerd en wordt het prototype verbeterd. Hierna wordt de iteratie afgerond. In de opeenvolgende iteratie begint het proces opnieuw, maar dan wordt er op het prototype van de vorige iteratie geïtereerd.

Uiteindelijk zal er in de laatste iteratie van het prototype een versie 1.0 uitgebracht worden. Hiervoor vindt een demonstratie plaats waar getest wordt dat het aan alle requirements voldoet. Hierna is de ontwikkelingsfase afgerond.

Nadat de ontwikkelingsfase is afgerond vindt de opleveringsfase plaats. In deze fase zal de applicatie “productie klaar” worden gemaakt, en wordt de documentatie hiervoor overgedragen. Het doel van deze fase is om Technolution de mogelijkheid te geven om de applicatie te onderhouden en door te ontwikkelen. De resultaten hiervan worden omschreven in de **Applicatie documentatie**.

3.6 Methodiek voor visualiseren ontwerpen.

In dit project wordt object georiënteerd geprogrammeerd. Voor visuele representatie van een object wordt er gebruik gemaakt van aspecten van de Unified Modeling Languege(UML). Helaas is er niet genoeg kennis om exact juiste UML te kunnen toepassen. Ook is er geen persoon binnen Technolution die deze kennis kan delen. Om deze reden wordt er in dit project van een extreem losse variatie van UML gebruik gemaakt. Hierbij wordt het volgende gebruikt voor deze doeleinde:

- Gebruiks diagram.
    - Een visuele representatie wat het programma moet doen die itteratie (wat gaat er gebeuren in het systeem).
- Klassendiagram
    - Overzicht van alle klasse, met hun functies, parameters, variabelen en relaties tussen de klassen.
    - Het project is in python gemaakt. Python kent geen access modifiers (public, private, protected). Om deze reden worden zijn deze attributen niet opgenomen in de klassendiagrammen.
- Sequentie diagram.
    - Hoe de klasse met elkaar in sequentie functioneren.

3.7 Op te leveren producten per fase

Initialisatie fase:

Voor de initialisatie fase wordt de visie en eisen voor het project vastgelegd. Ook wordt er gekeken naar de risico's in het project. De bevindingen worden beschreven in het **voorlopig plan van aanpak**.

Oriëntatiefase:

In de oriëntatiefase wordt er gekeken naar de haalbaarheid van het project. Er wordt onderzoek gedaan naar de mogelijkheden van de huidige eisen. Deze worden vastgelegd in het **analysedocument**. De bevindingen worden overlegd met de opdrachtgever en de visie en requirements worden eventueel aangepast. Het voorlopig plan van aanpak wordt aangepast en wordt het **definitieve plan van aanpak.**

Ontwikkelingsfase:

De ontwikkeling van de applicatie vindt plaats in de ontwikkelingen fase. Deze fase vindt in meerdere iteraties plaats. Bij iedere iteratie wordt een ontwerp gemaakt, een prototype geprogrammeerd en een demonstratie gehouden. Aan de hand van de verkregen feedback wordt het prototype verbeterd. In de opeenvolgende iteratie wordt weer doorgebouwd op het prototype.

Opleveringsfase

In de opleveringsfase wordt het product "productie klaar" gemaakt. De code wordt becomment en onderhoudbaar gemaakt. Er wordt een advies uitgebracht en er wordt documentatie geschreven voor het onderhoudt en de doorontwikkeling van het eindproduct. De resultaten hiervan worden omschreven in bijlage C: **Applicatie documentatie**

3.8 Fasering

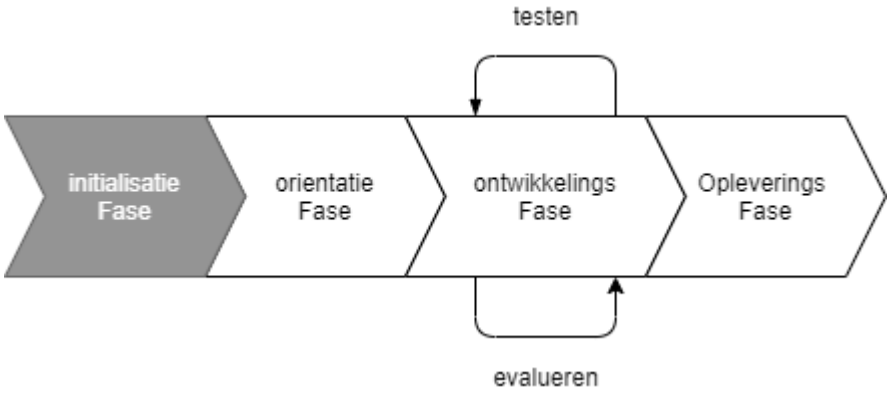
Dit afstudeerproject is verdeeld in 4 fases. Deze zijn weergeven in tabel 4.1. Een volledig overzicht van de planning kan gevonden worden in bijlage D: **plan van aanpak**

Naam	Doel	Tijdsduur
Initialisatiefase	Vergaren van de requirements. Maken risicoanalyse Opstellen voorlopig plan van aanpak	2 weken
Oriëntatiefase	Onderzoeksmogelijkheden USB Armory, Opstellen definities en plan van aanpak	2 weken
Ontwikkelingsfase	Ontwerpen van de software Maken prototype Demonstreren prototype en verbeteren aan de hand van feedback	12 weken in iteraties van 2 weken
Opleveringsfase	Final bugfixes, schrijven van documentatie voor de doorontwikkeling van het product.	2 weken

Figuur 3-4 fasering van project

4 Initialisatiefase

In dit hoofdstuk wordt de initialisatiefase van het project behandeld.



Tijdens de initialisatiefase van het project wordt het probleem domein en de voorlopige eisen opgesteld. In dit project is er geen voorkennis van de hardware omgeving en de mogelijkheden hiervoor. Hierdoor is de initialisatiefase de grondslag voor het project. Aan de hand van gesprekken met de opdrachtgever worden de randvoorwaarden en de voorlopige requirements opgesteld. In de oriëntatiefase wordt gekeken of deze ook haalbaar waren. Ook wordt er gekeken naar het risico van het project, en hoe deze voorkomen kan worden.

4.1 Visiegesprek met de opdrachtgever

Om de eisen te achterhalen werden er een drietal gesprekken gevoerd met de opdrachtgever. Hierbij werd overlegd wat er belangrijk is voor dit project; en welke zaken minder prioriteit hebben.

Uit de gesprekken met de opdrachtgever kwam naar voren dat hij graag een proof of concept zou willen zien waar een computer een VPN-tunnel zou kunnen opzetten zonder dat deze ooit direct verbonden te zou geweest met het onveilige internet. Recentelijk had hij een artikel gelezen over een flash drive sized computer: de USB Armory. Dit product was ontworpen voor security-achtige hardware solutions. Het proof of concept wou hij graag met de USB Armory als platform ontwikkeld zien.

Ook kwam er naar voren dat Technolution zelf al een VPN-solution had ontwikkeld die voor een netwerk een point to point VPN-tunnel kon opzetten: de Primelink 3015. De Primelink maakt gebruik van een geharde versie van OpenVPN genaamd OpenVPN-NL. Als het proof of concept bevalt zou deze mogelijk na afloop van het project worden doorontwikkeld bij Technolution. Hierbij is het wenselijk dat het aangesloten kan worden op bestaande Technolution producten. Daarom zou bij het ontwikkelen van de VPN-module gebruik gemaakt moeten worden van OpenVPN-NL voor de opgezet van de beveiligde VPN-tunnel.

Aan de hand van de gesprekken zijn de volgende voorlopige requirements opgesteld. Deze zouden nog aangepast kunnen worden na de oriëntatiefase.

Voorlopige requirements

ID	Omschrijving	Commentaar
FE01	De gebruiker hoeft enkel de USB VPN module in de computer te steken waarna het programma zichzelf start. De gebruiker hoeft verder geen fysieke handelingen met de module uit te voeren.	
FE02	De USB VPN module probeert een secure VPN-tunnel op te zetten.	
FE03	De computer hoeft geen directe verbinding te maken met het internet voor de USB VPN module om een VPN-tunnel op te zetten	
FE04	De VPN-module routeert het internetverkeer tussen de computer en het internet.	Anders kan die wel een VPN-verbinding opzetten, maar dan gebeurt er niets mee
FE05	De computer kan niet direct bij de VPN-module komen, configuratie moet op een indirecte manier plaatsvinden.	Bijvoorbeeld via bluetooth, of via een externe SSH-sessie. Hiervoor bijvoorbeeld van een GUI opzetten.
FE06	Wi-Fi hotspot selectie/verbinding vindt plaats op de VPN-module, de gebruiker mag wel indirect deze selecteren.	
FE07	De VPN-module moet een vpn tunnel op kunnen zetten met een Technolution PrimeLink	

4.2 Risicoanalyse

Om in het project zo goed mogelijk volgens de planning te kunnen werken werd er een risicoanalyse uitgevoerd. Hierbij werd er gekeken naar mogelijke factoren die vertraging zouden kunnen veroorzaken. Dervolgens werd hier een correctieve en een preventieve oplossing voor bepaald. In het schema hieronder is een beknopt overzicht van de belangrijkste risico's en acties terug te lezen. De volledige risicoanalyse valt terug te lezen in bijlage D: **het plan van aanpak**

Risico	Vertraging afstudeeropdracht door ziekte of corona
Actie(Preventief)	Vanuit huis werken, grote groepen mensen vermeiden.
Actie(Correctief)	Kijken of opdracht aangepast kan worden, kijken of verlenging mogelijk is.

Risico	Onvoldoende begeleiding door uitval bedrijfsmentor
Actie (Preventief)	Wekelijks contact met de bedrijfsmentor waardoor er altijd voldoende begeleiding is.

Actie (Correctief)	Contacteren bedrijfsbegeleider. Het vinden van een nieuwe bedrijfsmentor.

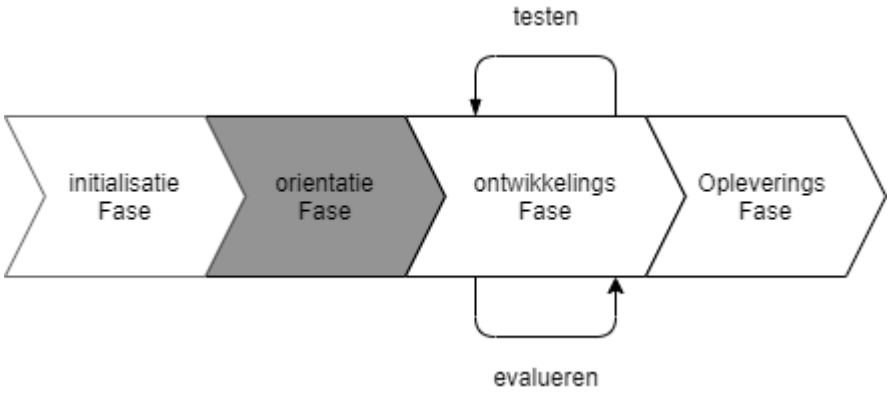
risico	Opdrachtgever/bedrijfsbegeleider valt weg.
Actie(Preventief)	Wekelijks contact met de bedrijfsbegeleider/opdrachtgever.
Actie(Correctief)	In overleg zou de bedrijfsmentor de taak van opdrachtgever/bedrijfsbegeleider kunnen overnemen. Wijziging doorgeven aan de HHS

risico	Hardware is niet leverbaar/ functioneert niet naar behoren
Actie(Preventief)	Hardware vroeg bestellen, analyse uitvoeren of de hardware geschikt is voor project
Actie(Correctief)	Opdracht uitvoeren op een ander Linux based platform (bijvoorbeeld raspberry pi). In een latere fase in het project de software overzetten naar de USB Armory

risico	USB Armory kan niet met Wi-Fi of ethernet verbinden
Actie(Preventief)	Checken of de juiste drivers beschikbaar zijn op gekozen Linux distributie zijn voor de te gebruiken hardware
Actie(Correctief)	Drivers handmatig installeren op Linux distributie, andere Wi-Fi dongel gebruiken, ergste geval andere Linux distributie uitkiezen waar de Wi-Fi dongel wel werkt

5 Oriëntatiefase

In dit hoofdstuk wordt de oriëntatiefase van het project behandeld

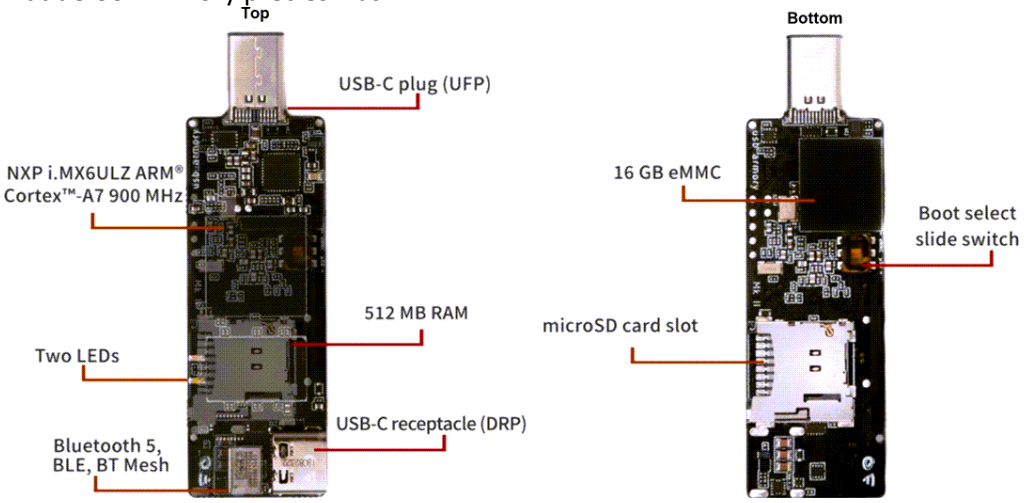


Tijdens de initialisatie fase is het probleemdomein, de requirements, en de visie voor het project opgesteld. In deze fase wordt er een analyse onderzoek uitgevoerd om te kijken wat de mogelijkheden zijn om deze visie te realiseren. Hierbij is in het analyse onderzoek gefocust om de gaten in kennis over de USB Armory en OpenVPN-NL te dichten. In dit hoofdstuk worden de resultaten van het onderzoek besproken. Het volledige analyserapport is terug te lezen in bijlage B: **het analysedocument**

Aan het einde van het analyse onderzoek is er een advies uitgegeven of de gestelde requirements haalbaar zijn of eventueel aangepast moeten worden. Hierna is er een planning gemaakt voor de ontwikkelingsfase. Hier wordt er per iteratie 1 of meerdere eisen toegekend waar tijdens die iteratie een prototype voor ontwikkeld wordt.

5.1.1 USB Armory

Tijdens het gesprek met de opdrachtgever gaf hij aan dat hij het proof of concept graag ontwikkeld zou willen zien worden op een USB Armory. De eerste stap van het analyse onderzoek was kijken wat de USB Armory precies was.



Figuur 5-1 USB Armory MK 2(bron: https://nl.mouser.com/new/f-secure/crowd-supply-usb-Armorymkii/)

De USB Armory is een opensource computerplatform ter grootte van een usb stick. Hij is ontwikkeld met Information Security Applications in mind. Het bijzondere van de USB Armory is dat hij een groot assortiment aan features heeft die ervoor zorgen dat het ideaal is voor het ontwikkelen van secure solutions. Enkele voorbeelden van deze features is Ethernet, USB storage en keyboard/mouse emulation, ARM trustzone, en een externe cryptographic co-processor.

USB Armory Hardware

Naam	Spec
CPU	NXP i.MX6UL 528MHz /i.MX6ULZ ARM® Cortex™-A7 900MHz
Werkgeheugen	512MB/1GB DDR3 RAM (afhankelijk van versie)
Videokaart	Geen
Opslag	Internal 16 GB eMMC + External SD kaart
Aansluitingen	DRP (Dual Role Power) receptacle + UFP (Upstream Facing Port) plug, USB 2.0 only (*no video support*), via externe debug card UART, GPIO, SPI, I²C, CAN breakout
Afmeting	65 x 19 x 6 mm
Besturingssysteem	Ondersteuning voor Linux en android.
Connectiviteit	u-blox ANNA-B112 Bluetooth module

Figuur 5-2 Hardware specificaties

De USB Armory komt is twee varianten. De MX6UL en de MX6ULZ. Het grootste verschil tussen deze twee versies is de kloksnelheid. De MX6UL heeft een kloksnelheid van 528MHz waar de MX6ULZ een kloksnelheid van 900 MHz heeft. In dit project moet er een OpenVPN-NL server gehost worden op de Armory. Hoewel OpenVPN geen zware CPU-footprint heeft is het niet wenselijk dat VPN verkeer vertraagdt wordt omdat de CPU het niet aan kan. Om deze reden alleen al is de MX6ULZ geschikter voor dit project dan zijn kleinere broertje.

Besturingssysteem mogelijkheden

Voor het besturingssysteem maakt de Armory gebruik van Linux. In theorie zou iedere distro hierop moeten kunnen draaien, maar zou deze wel eerst uitgerust moeten worden met de juiste drivers en programma's. Er zijn 2 kant en klare images beschikbaar die enkel op de Armory geïnstalleerd hoeven worden: **Arch Linux** en **Debian 10**. De Debian 10 variant is door de fabrikant van de Armory (F-Secure) gemaakt en krijgt maandelijkse updates. Arch Linux is in het verleden door de community opgezet. De laatste update die deze heeft gekregen was al bijna meer dan een jaar geleden. Omdat de Debian variant maandelijkse updates krijgt van de fabrikant is dit het meest geschikte besturing systeem om te gebruiken.

Wireless en wired connecties

De USB Armory heeft standaard geen Wi-Fi mogelijkheden. Via ethernet emulation kan er op de host PC een internetverbinding gedeeld worden met de Armory. De armory heeft wel een externe USB C poort. Via een losse Wi-Fi/ethernet dongel kan hier een wired of wireless verbinding opgezet worden. Via Ethernet emulation kan deze verbinding met de host PC gedeeld worden.

Cryptografische mogelijkheden

De armory heeft ook een externe cryptografische co-processor. Deze processor kan aangeroepen worden voor verscheidenen hashing en cryptografische functies. Omdat er in dit project gebruik gemaakt wordt van een bestaande package (OpenVPN-NL) met zijn eigen encryptie library zal de crypto co-processor niet gebruikt worden in het project. In theorie zou deze gebruikt kunnen worden om de encryptie library van OpenVPN-NL te vervangen. Maar dit valt buiten de scope van het project.

5.1.2 OpenVPN-NL

Tijdens het gesprek met de opdrachtgever in de initialisatie fase gaf de opdrachtgever aan dat Technolution voor hun eigen VPN-solution gebruik maakte van OpenVPN-NL. Hij zou graag zien dat het proof of concept ook met deze VPN-solution zou kunnen verbinden.

OpenVPN-NL is een aangepaste versie van OpenVPN die is aangepast om zoveel mogelijk beveiligs maatregelen te bevatten om in een *secure environment* (bijvoorbeeld een ministerie netwerk) te werken. Deze maatregelingen omvatten patches om OpenVPN te *harden* (extra beveiligen). Tevens zijn er ook verbeteringen in de documentatie aangebracht om evaluaties makkelijker te maken.

Deze versie van OpenVPN is ontwikkeld door Fox-IT, in samenwerking met de AIVD, nadat een aantal ministeries interesse toonde om OpenVPN te gebruiken. Echter voldeed OpenVPN niet aan de veiligheid eisen van de ministeries. Als antwoord werd OpenVPN-NL ontwikkeld.

Verschil tussen OpenVPN-NL en OpenVPN

OpenVPN-NL en OpenVPN zijn vrijwel identiek. Het grootste verschil is dat OpenVPN-NL een aantal encryptie types heeft “uitgeschakeld” (je krijgt een melding als je ze probeert te gebruiken). In tabel 5-2 is een overzicht van de grootste verschillenen te zien.

Afgezien van deze verschillen is de functionaliteit hetzelfde. OpenVPN en OpenVPN-NL zijn interchangeable. Hiermee bedoeld wordt dat een client van de een verbinding kan maken met de server van de ander (mits de gekozen crypto in beide producten beschikbaar zijn.). Configuratiebestanden kunnen onderling uitgewisseld worden.

	OpenVPN	**OpenVPN-NL**
onderhouder	OpenVPN community	Fox-it
Certificering	Geen	NLNCSA criteria Level 2, "Departementaal VERTROUWELIJK" (Dutch) if deployed in compliance with deployment advisory ("inzetadvies")
Funtionaliteit	Volledig	Onveilige opties verwijderd, of hardened, anders ongewijzigd
Cryptografische Library	OpenSSL	Mbed TLS
Default encryptie	BF-CBC / AES-256-GCM, SHA1	AES-256-CBC / AES-256-GCM, SHA256 (andere opties niet toegestaan)
Toegestanne groepen voor (EC)DH	DH: 1024-8192 bits, ECDH: any supported curve	DH: 2048-4096 bits, ECDH: P-256, P-384
Acceptabele size RSA	1024-8192 bits	1024-4096 bits (using less than 2048 bits is advised against, and requires explicitly setting "--tls-profile legacy")

Figuur 5-2 verschil OpenVPN en OpenVPN-NL (bron: https://OpenVPN.fox-it.com/about.html

5.2 Conclusie analyse

Aan het einde van de analyse is er gekeken of de requirements haalbaar zijn, of dat deze moesten worden aangepast naar iets reëlers. Uit de analyse is gebleken dat de requirements op eerste oogopslag prima haalbaar zijn en deze niet aangepast hoeven te worden.

De USB Armory heeft een goed assortiment heeft aan securityfeatures waar gebruik gemaakt van zouden kunnen worden voor het project. Zowel op hardware niveau als op softwareniveau zijn er verscheiden features die er gebruikt kunnen worden om het eindproduct veiliger of meer functionaliteit te geven. Voor de functionaliteit van een VPN-module is het heel handig dat hij standaard al Ethernet emulatie functionaliteit heeft.

Het grootste nadeel aan de Armory is dat hij wel een Bluetooth module heeft, maar geen Wi-Fi/ethernet module. Hiervoor zou er gebruik gemaakt moeten worden van een externe Wi-Fi dongel om de Armory van internet te voorzien.

Bij het onderzoek naar OpenVPN-NL bleek dat deze in vrijwel alle aspecten gelijk is aan een normale instantie van OpenVPN. Een OpenVPN client zou prima met een OpenVPN-NL server moeten kunnen verbinden en vis versa mits ze dezelfde securityprotocollen gebruiken.

5.3 Iteratie planning

Nadat bevestigt was dat de requirements haalbaar waren is er een planning opgesteld welke eisen uitgevoerd worden per iteratie. In tabel 5-3 worden deze weergeven

Iteratie	Eis.	Werkzaamheden
Iteratie 1	FE01	Opzetten USB Armory, kunnen draaien van programma als gebruiker module heeft ingeplugd
Iteratie 2	FE02/FE03	Opzetten van OpenVPN-NL secure tunnel.
Iteratie 3	FE04	Routering tussen de host computer en de VPN-tunnel
Iteratie 4	FE05	Maken van een GUI voor indirecte configuratie
Iteratie 5	FE06	Wi-Fi selectie, moet plaatsvinden op de Armory, maar op en of andere manier wel input nodig vanaf de computer
Iteratie 6	FE07	Armory moet een VPN-tunnel kunnen opzetten met een Technolution PrimeLink

Figuur 5-3 iteratieplanning tabel

6 Ontwikkelingsfase: iteratie 1

In iteratie 1 werd de volgende eis behandeld:

- FE01 De gebruiker hoeft enkel de USB VPN module in de computer te prikken waarna het programma zichzelf start. De gebruiker hoeft verder geen fysieke handelingen met de module uit te voeren.

Omdat dit de eerste fase van het project was moest ook de USB Armory hardware opgezet worden en benaderbaar gemaakt worden. Voordat er aan de functionele eis gewerkt kon worden moest eerst de hardware verkregen en ingesteld worden. Dit hield in:

- Het selecteren en installeren van een besturingssysteem op de USB Armory.
- Het kunnen benaderen en kunnen configureren van de USB Armory
- De USB Armory voorzien van een (externe) internetverbinding.

Zodra deze punten verwerkt waren kon er gewerkt worden aan de functionele eis FE01:

- Het draaien van een programma op de Armory.

6.1 Voorbereidingen

Voor dat er met de USB Armory gewerkt kon worden moest deze eerst ingesteld worden. In dit deel-hoofdstuk worden de gemaakte keuzes en werkzaamheden besproken.

6.1.1 Keuze besturingssysteem

De USB heeft geen standaard besturingssysteem dat wordt meegeleverd. In theorie moet iedere Linux distributie erop kunnen draaien. Echter zijn er dan wel losse drivers nodig om functies als Ethernet emulation, de LED lights, en de Crypto Co-processor te kunnen gebruiken. In het kader van tijdsbesparing wordt er gebruik gemaakt van een pre-compiled image.

Voor pre-compiled besturingssysteem waren er 2 keuzes. Debian 10 dat wordt onderhouden door de makers van de USB Armory, en Arch Linux welk onderhouden wordt door de community. In de tabel staan de grootste verschillen tussen deze Linux distributies.

	Debian (USB Armory)	Arch Linux (USB Armory)
Onderhouden door	F-Secure Foundry (makers USB Armory)	Arch Linux community
Update cycle (kernel/OS vesie)	+- 6 maanden	Onbekend (geen geschiedenis beschikbaar)
Support of forum	Google Groups	Geen
Package manager	Apt-get	Pacman
Moeilijkheid graad	Beginner vriendelijk	Wat Linux ervaring nodig

Figuur 6-1 verschillen arch Linux vs Debian

Er is gekozen voor Debian 10 (Buster). Deze image is gemaakt door de makers van de USB Armory en wordt maandelijks bijgewerkt. De hoofdreden hiervoor is dat op de google forums van de USB Armory wel ondersteuning voor de Debian variant wordt gegeven, en niet voor Arch Linux variant. Ook was ik meer bekend met de werking en package managers van Debian.

Nadat de image gedownload is kon deze door middel van het programma Etcher op de SD Kaart geïnstalleerd worden.

6.1.2 Benaderen/configuratie

Als de Debian image goed op de SD kaart is geïnstalleerd zou het blauwe lampje moeten gaan knipperen volgens de Github page. Ook start de image de Ethernet over USB emulation (CDC Ethernet) waarbij hij een ethernet verbinding emuleert.

Nadat de Armory begon met knipperen kreeg de computer via DHCP een adres in de 10.0.0.0/24 range. Hierna kon er verbinding gemaakt worden met de Armory op het ip 10.0.0.1. De standaard username is **usbarmory**, en het wachtwoord is **usbarmory**

6.1.3 Externe connectie.

Standaard heeft de USB Armory geen ethernet of Wi-Fi mogelijkheden. Door middel van een USB Wi-Fi dongel zou de USB Armory uiteindelijk zelfstandig verbinding moeten kunnen maken met het internet.

Om de USB Armory te kunnen updaten werd er gebruik gemaakt van Windows 10 Connection Sharing. Hiermee werd de internetverbinding van de host pc gedeeld met de USB Armory zodat de image geüpdatet kan worden.

Als volgende stap werd er getracht een TP-Link TL-WN823N Wi-Fi usb Dongel werkend te krijgen op de USB Armory. Door middel van de tool **LSHW** kon er geconstateerd worden dat hij standaard hier niet de driver voor geïnstalleerd had.

```
   *-usb UNCLAIMED
        description: Generic USB device
        product: 802.11n NIC
        vendor: Realtek
        physical id: 1
        bus info: usb@1:1
        version: 2.00
        serial: 00e04c000001
        capabilities: usb-2.10
        configuration: maxpower=500mA speed=480Mbit/s
```

Figuur 6-2 Output LSHW

De drivers op de website van de fabrikant ondersteunde max tot kernel versie 3.10. De huidige image draaide kernel 5.4.87. Echter veel chipsets worden als de fabrikant de drivers hiervoor niet meer update onderhouden door de opensource community. De chipset van de TP-link TL-WN283n was hierop geen uitzondering.

Door Debian package **"firmware-realtek"** te downloaden zouden alle opensource Realtek drivers geïnstalleerd worden. In eerste instantie was deze package niet beschikbaar via de standaard packagemanager **apt-get**. Na wat onderzoek bleek dat de package een **Non-free** package was (voldoet niet aan GNU license). Door de /etc/apt/sources.list aan te passen kon **apt-get** geforceerd worden om deze alsnog te installeren. Echter kon hij de Wi-Fi dongel nog steeds niet herkennen.

6.1.4 Externe connectie: Kernel recompileren voor driver.

De volgende stap om de driver werkend te krijgen was het recompilen van de kernel, en hierbij Realtek support aan de kernel toe te voegen. Hierbij werd er gebruik gemaakt van een tutorial van het internet.
Via de **make menuconfig** kon op simpele wijze de .config file aanpassen zodat de Realtek drivers mee geïnstalleerd zouden worden.
Een probleem met de guide was dat ervan uitgegaan werd dat de kernel op dezelfde computer gecompileerd werd. De USB Armory heeft een single core processor met 0.9 MHz processorkracht.

Het zou gemiddeld 9 uur duren om een kernel te compileren. Om de tijd te verkorten werd er gebruik gemaakt van de compilatie server van Technolution.

Bij het installeren van de kernel image was er de foutmelding "**Kernel header files not in any of the expected locations**". Bij nader onderzoek bleek dat normaal gesproken de headerfiles geïnstalleerd moeten kunnen worden via de de Package manager, apt-get. Echter waren deze enkel beschikbaar voor kernel versie 4.19.x. Hierdoor kon de gebouwde kernel niet geïnstalleerd worden.

Op de github pagina van de pre-compiled image werd ook een Makefile aangeboden om zelf een image te genereren. Deze Makefile bestond uit niks meer dan een lange lijst Linux commando's. Na een virtuele omgeving te hebben gemaakt om deze image kunnen te compileren werd de regel die de kernel .config file downloaden vervangen met mijn eigen .config file met Realtek support erin. Hierna kon er succesvol een image gegenereert worden.

Als de TP-Link TL WN821N Wi-Fi dongle ingeplugt wordt is er te zien met het commando **dmesg** dat de driver faalt om in te laden.

```
138.820690] usb 1-1: RTL8192EU rev B (SMIC) 2T2R, TX queues 3, WiFi=1, BT=0, GPS=0, HI PA=0
138.831593] usb 1-1: RTL8192EU MAC: c0:25:e9:10:59:dd
138.839009] usb 1-1: rtl8xxxu: Loading firmware rtlwifi/rtl8192eu_nic.bin
138.848022] usb 1-1: Direct firmware load for rtlwifi/rtl8192eu_nic.bin failed with error -2
138.859058] usb 1-1: request_firmware(rtlwifi/rtl8192eu_nic.bin) failed
138.867918] usb 1-1: Fatal - failed to load firmware
138.875252] rtl8xxxu: probe of 1-1:1.0 failed with error -11
138.883605] usbcore: registered new interface driver rtl8xxxu
```

Figuur 6-3 Output van dmseg met wireless

Bij het inpluggen van een merkloos Ethernet usb dongle is hij echter nu wel in staat om te herkennen (zie figuur 6-4).

```
usbarmory@usbarmory:~$ lshw -c network
WARNING: you should run this program as super-user.
  *-network:0
       description: Ethernet interface
       physical id: 1
       bus info: usb@1:1
       logical name: eth0
       serial: d0:37:45:be:5f:ed
       size: 1Gbit/s
       capacity: 1Gbit/s
       capabilities: ethernet physical tp mii 10bt 10bt-fd 100bt
       configuration: autonegotiation=on broadcast=yes driver=r8
ort=MII speed=1Gbit/s
```

Figuur 6-4 lshw output van merkloze ethernet usb dongel.

Het traject om de USB Armory van een externe internetverbinding te voorzien duurde ongeveer anderhalve week. Hoewel hij nog geen verbinding kan maken met Wi-Fi was hij nu wel voorzien van een internetverbinding via ethernet. Om deze reden was ervoor gekozen om de rest van de iteratie te werken aan de code voor de functionele eis. In een latere iteratie zou geprobeerd worden met een andere Wi-Fi usb Adapter een Wi-Fi verbinding werkend te krijgen.

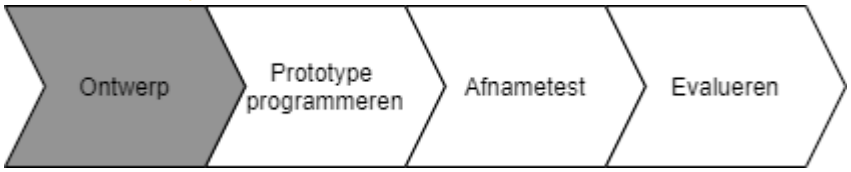
6.1.5 Keuze programmeertaal

De afstudeeropdracht is geprogrammeerd is python. De opdrachtgever gaf aan dat hij geen voorkeur had in welke programmeertaal het geprogrammeerd was. De keuze was volledig aan de afstudeerder. Er is voor python als programmeertaal gekozen om de volgende redenen:

- Python is een script taal. Er hoeft geen tijd gespendeerd te worden om een compiler omgeving opgezet op te zetten.
- Het is uitermate geschikt voor het schrijven van programma's binnen een Linux omgeving.
- Het heeft op het moment van schrijven meer dan 137.000 libraries en frameworks die gebruikt kunnen worden om het prototype te maken.

Out of the box is python en extreem geschikte taal om prototypes te maken. Het heeft voor vrijwel iedere functie een library. Ook is het een zeer syntax vriendelijke taal omdat er geen type controle plaatsvindt. Dit en in combinatie dat er al ervaring was met programmeren in python was dit genoeg reden om dit als programmeertaal voor het project te gebruiken.

6.2 Ontwerp



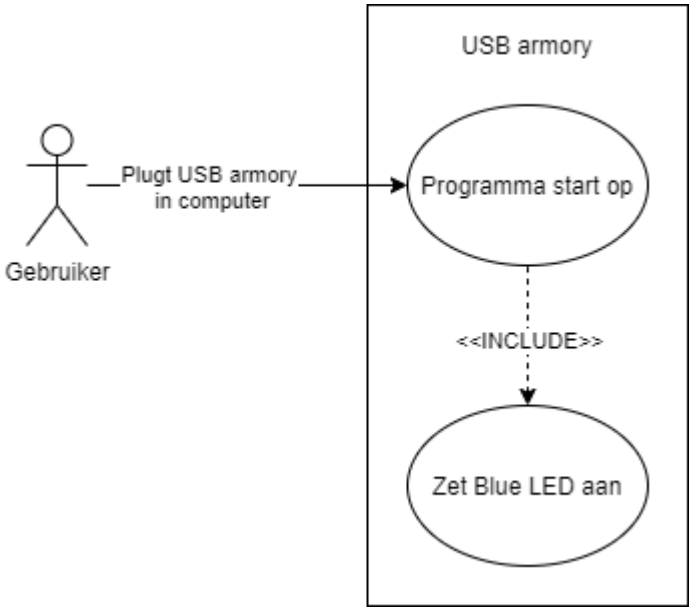
Voor het ontwerpen zijn de eisen eerst omgeschreven naar use-cases. Deze zijn verwerkt in een use-case diagram. Hierna is er per Use-Case een omschrijving van alle stappen gemaakt. Voor de code is een klasse diagram en sequentie diagram hoe de code eruit moet zien en moet functioneren

6.2.1 Gebruiksdiagram

De functionele eis die bij deze iteratie hoorde was als volgt:

- FE01 De gebruiker hoeft enkel de USB VPN module in de computer te prikken waarna het programma zichzelf start. De gebruiker hoeft verder geen fysieke handelingen met de module uit te voeren

Om dit te demonstreren werd er een programma gemaakt dat automatisch moet starten zodra de USB Armory opgestart is. Om een visuele demonstratie hiervan te geven werd het ledlampje van de USB Armory aan-en uitgezet.



Figuur 6-5 gebruiksdiagram

In het Gebruiksdiagram is te zien dat de gebruiker de USB Armory in zijn computer plugt. Zodra deze is opgestart start hij het programma. Het programma knippert vervolgens het blauwe LED licht.

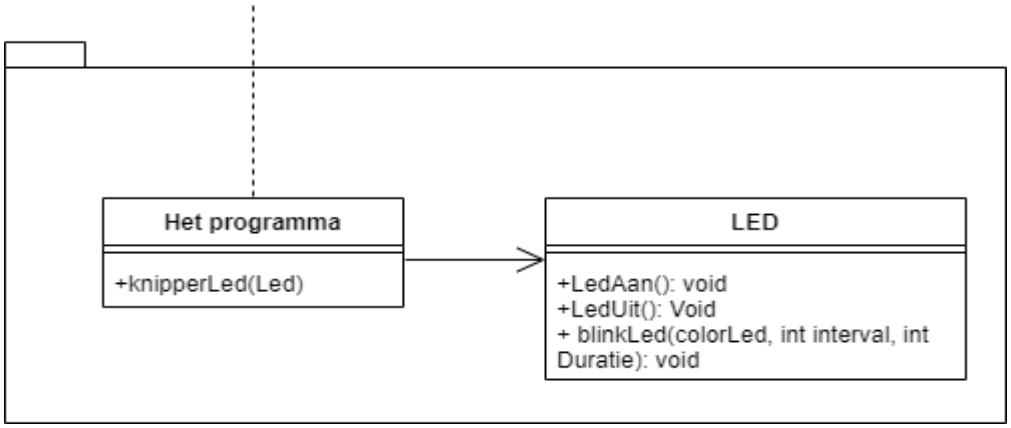
In tabel 6-6 wordt de use-case omschreven.

Id	Use-case 1
Use-Case	Start program na opstarten.
Brief Description	Automatisch starten van een programma als de Armory is opgestart
Primary Actors	Computergebruiker
Secundary Actors	USB armory
Pre-condition	Geen
Main flow	1. Gebruiker plugt USB Armory in computer 2. USB Armory start OS 3. USB Armory start programma 4. Programma zet knippert de LED op armory 5. Gebruiker ziet blauwe licht knipperen
Post conditions	1. USB Armory is opgestart 2. Programma draait 3. Blauwe led knippert
Alternative flows	Geen

Figuur 6-6 Use-case 1

6.2.2 Klassendiagram

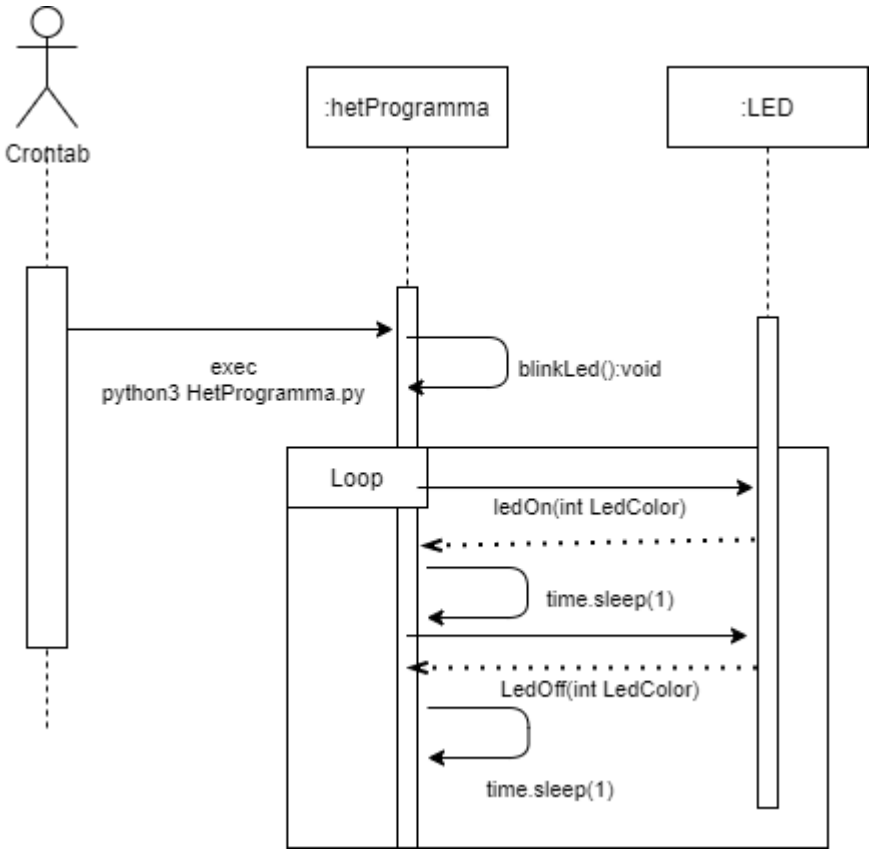
In dit hoofdstuk wordt het klasse diagram van use-case 1 opgesteld (figuur 6-7). De klasse “Het programma” stuurt de functie “knipper led” het de blauwe ledlamp aan. Hierbij is de klasse “Het Programma” een dummy klasse die enkel bedoeld is voor de demonstratie van de use-case en zal zich niet in het eindproduct bevinden.



Figuur 6-7 klasse diagram Use-case 1

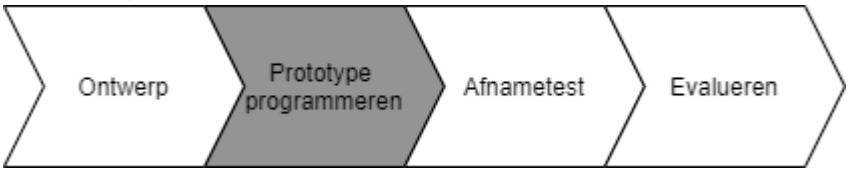
6.2.3 Sequentie diagram

In figuur 6-8 is het sequentie diagram voor het programma te zien. Als de USB Armory opgestart is start de entry in de crontab een python proces van HetProgramma.py. Hierna maakt het programma een object van “Het programma” aan en roept de functie blinkLed() aan.



Figuur 6-8 Sequentie diagram iteratie 1

6.3 Programmeren prototype



In dit deelhoofdstuk worden de gemaakte klasse besproken.

Led klasse
De led klasse stuurt de witte en de blauwe led aan. Hierbij is ervoor gekozen om gebruik te maken van de **Armoryctl** package omdat directe waardes naar GPIO-pinnen wegschrijven in /sys/class/ledsdevices onvoorspelbaar gedrag vertoonde.

De LedAan en LedUit functies in de roepen een childprocess aan die het commando uitvoeren, waarna het childprocess weer verdwijnt.

```
def LedOn(self, color):
    if(color == 1): #white
        subprocess.run("armoryctl led white on", shell=True)
    if(color == 2):
        subprocess.run("armoryctl led blue on", shell=True)
```

Figuur 6-9 broncode ledOn functie

De functie blinkLed() functioneert iets anders. Zoals te zien in figuur 6-9 creëert de functie een nieuw proces. Dit process voert de led blink uit met de duratie en het interval die zijn meegegeven in de parameters. Hij spawnt hier een los process omdat een subprocess die in de shell draait *blocking* is. Hierdoor zou het hoofdprogramma moeten wachten tot de ledlampen klaar zijn met blinken voor het verder kan gaan met de volgende taken.

Momenteel wordt er dus voor iedere keer dat blinkLed() aangeroepen een nieuw process gespawnt die zijn functie uitvoert en zichzelf hierna terminate. Mogelijk zou het efficiënter zijn om de klasse led in een permanent los proces te draaien die reageert op functie calls. Echter de ledlampen worden enkel aan het begin van het programma aangestuurd. Dit om e weergeven of het programma is opgestart. Om deze reden is deze manier efficiënter.

Het programma klasse
De “Het programma” klasse is een dummy klasse die enkel gebruikt wordt bij deze iteratie om de Led klasse te demonstreren. Zoals te zien in figuur 6-10 maakt het een instantie van de led klasse aan. Hierna komt het in een while loop die de led laat blinken tot het programma beëindigd wordt.

```
import led, time

class HetProgramma:
    def blinkLed():
        l = Led()
        while True:
            l.ledOn(l)
            time.sleep(l)
            l.ledOff(l)
            time.sleep(l)

if __name__ == "__main__":
    p = HetProgramma()
    p.blinkLed()
```

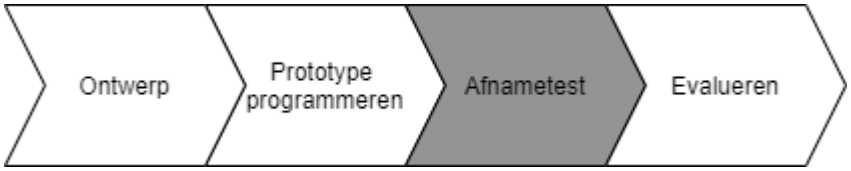
Figuur 6-10 broncode “het programma” klasse.

Automatisch starten van programma
Voor het starten van het programma wordt er gebruik gemaakt van crontab. Door de volgende regel toe te voegen aan de crontab draait het programma iedere keer dat de armory opgestart wordt.

```
@reboot python /home/usbarmory/scripts/hetprogramma.py &
```

Hierbij zorgt de **@reboot** flag ervoor dat het uitgevoerd wordt bij iedere reboot. Python draait het programma dat opgeslagen staat onder /**home/usbarmory/scripts/hetprogramma.py**. Aan het einde is een **&** teken toegevoegd zodat het programma op de achtergrond wordt uitgevoerd en de armory blijft doorstarten.

6.4 Afnametest



Binnen dit increment is het prototype gedemonstreerd aan de opdrachtgever. Hierbij testen hij of het prototype voldeet aan de eis die gesteld was in deze sprint. Hierbij bestond de testomgeving uit de usb armory, en een testcomputer die was opgestart.

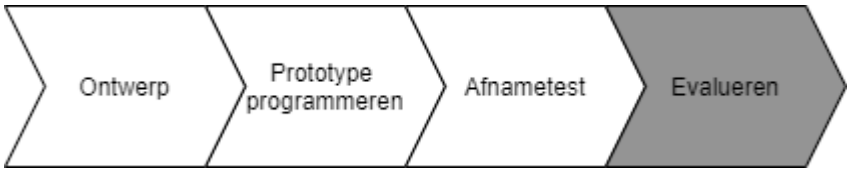
De opdrachtgever kon de USB armory in de testcomputer steken. Hierna starten na enkele ogenblikken het programma, welk de blauwe LED lampjes liet knipperen. Ook werd er via SSH met de USB armory verbonden. Via het **top** commando aangetoond worden dat het programma automatisch gestart was op de computer. In figuur 6-11 is een samenvatting van de testprocedure.

Test	Use-case 1
Use-Case	Start program na opstarten.
Brief Description	Automatisch starten van een programma als de Armory is opgestart
Pre-condition	De computer is opgestart, en geeft stroom aan de USB-poort
Test process	1. Steek de USB armory in de computer 2. Wacht ongeveer 30 seconden tot de USB armory is opgestart
Verwachte uitkomst	Het blauwe LED van de USB armory ging knipperen. Via het commando **top** kon geverifieerd worden dat het python programma automatisch gestart was
Uitkomst	Uitkomst zoals verwacht

Figuur 6-11 afnametest procedure

In dit increment is aan de eisen voldaan. Er kan automatisch een programma gestart worden op de USB Armory. De resultaten werden met de opdrachtgever besproken. De opdrachtgever maakte de opmerking dat er momenteel nog weinig te zien was om te demonstreren. Ook werkte de Wi-Fi verbinding nog steeds niet. Hierbij was hij het wel eens om een andere Wi-Fi dongel te bestellen en in een latere fase deze te implementeren in het prototype.

6.5 Evalueren afnametest



Bij de demonstratie van iteratie 1 was nog vrij weinig te zien. Het grootste deel van de tijd was gespendeerd om de drivers voor de Wi-Fi/ethernet verbinding op orde te krijgen.
Buiten de opmerking dat het jammer was dat de Wi-Fi nog niet werkte waren er verder geen op- of aanmerkingen van de opdrachtgever.

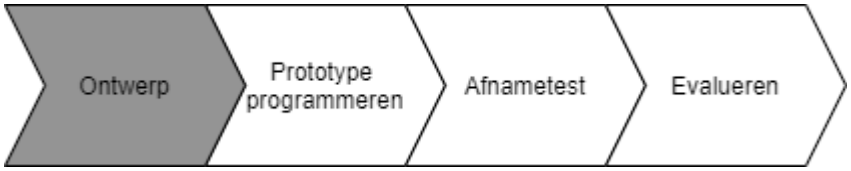
7 Ontwikkelingsfase: Iteratie 2/3

In iteratie 2 werden de volgende eisen behandeld:

- FE02 De USB VPN module probeert een secure VPN-tunnel op te zetten.
- FE03 De computer hoeft geen directe verbinding te maken met het internet voor de USB VPN module om een VPN-tunnel op te zetten
- FE04 De VPN module routeert het internetverkeer tussen de computer en het internet.

In eerste instantie zou FE04 uitgevoerd worden in iteratie 3. Echter tijdens iteratie 2 bleek dat deze eis veel simpeler te implementeren was dan gedacht. Om deze reden is deze eis ook meegenomen in de ontwikkeling van deze iteratie.

7.1 Ontwerp

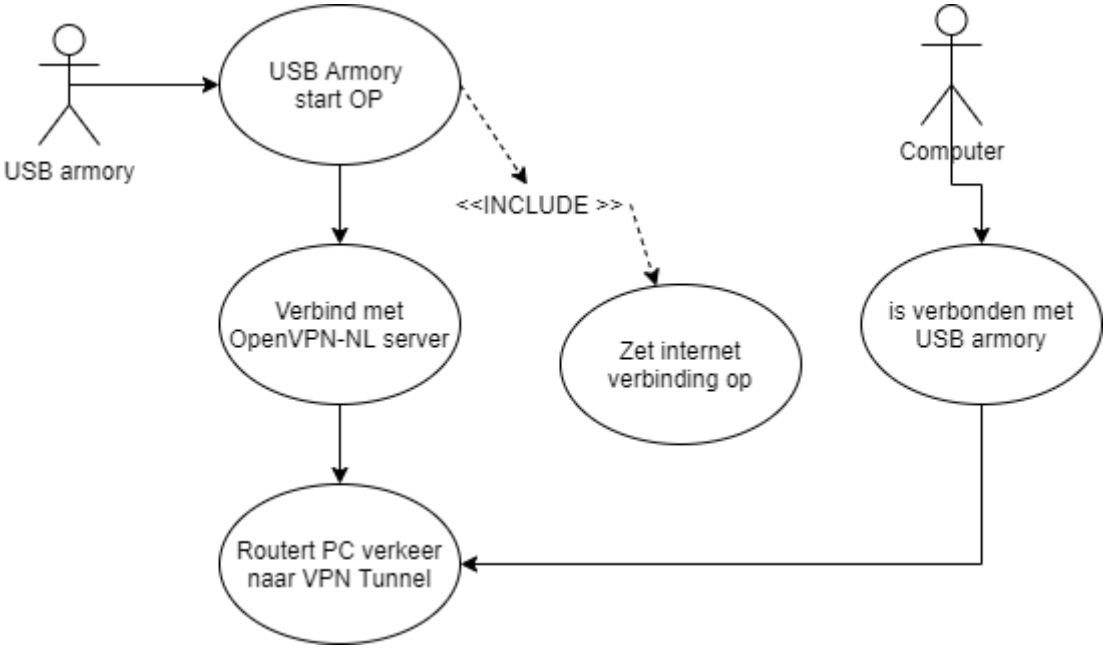


Voor het ontwerpen zijn de eisen eerst omgeschreven naar use-cases. Deze zijn verwerkt in een use-case diagram. Hierna is er per Use-Case een omschrijving van alle stappen gemaakt. Voor de code is een klasse diagram en sequentie diagram hoe de code eruit moet zien en moet functioneren.

7.1.1 Gebruiksdiagram

Om de functionele eisen te halen is het de bedoeling dat de computer een VPN-tunnel verbinding kan krijgen zonder dat hij verbinding moet maken met het “onbeveiligde internet”.

Om dit te bereiken zet de USB Armory zodra die opgestart is (en een internetverbinding heeft) een verbinding op met de OpenVPN-NL server. Hierbij zet hij een beveiligde VPN Tunnel op waar al het verkeer van de PC over gerouteerd kan worden. De stappen hiervoor zijn gevisualiseerd in figuur 7-1



Figuur 7-1 Gebruiksdiagram iteratie 2/3

In de tabel hieronder zijn de use-cases omschreven voor iteratie 2 en 3

Id	Use-case 2
Use-Case	De USB VPN module probeert een secure VPN-tunnel op te zetten.
Brief Description	De USB Armory verbindt met de OpenVPN-NL Server, en zet een VPN-tunnel op.
Primary Actors	USB Armory
Secundary Actors	Geen
Pre-condition	- USB Armory is opgestart - USB Armory heeft een internetverbinding.
Main flow	1. USB Armory leest OpenVPN-NL config file in 2. USB Armory zet VPN-tunnel op 3. USB Armory routeert computerverkeer over VPN-tunnel.
Excepties	2a. Armory kan geen verbinding maken met server 2b. wordt foutmelding weergeven.
Post conditions	- USB Armory heeft Actieve VPN tunnel. - Computer heeft internetverbinding. - USB Armory routeert al het verkeer van de computer over de VPN-tunnel
Alternative flows	Geen

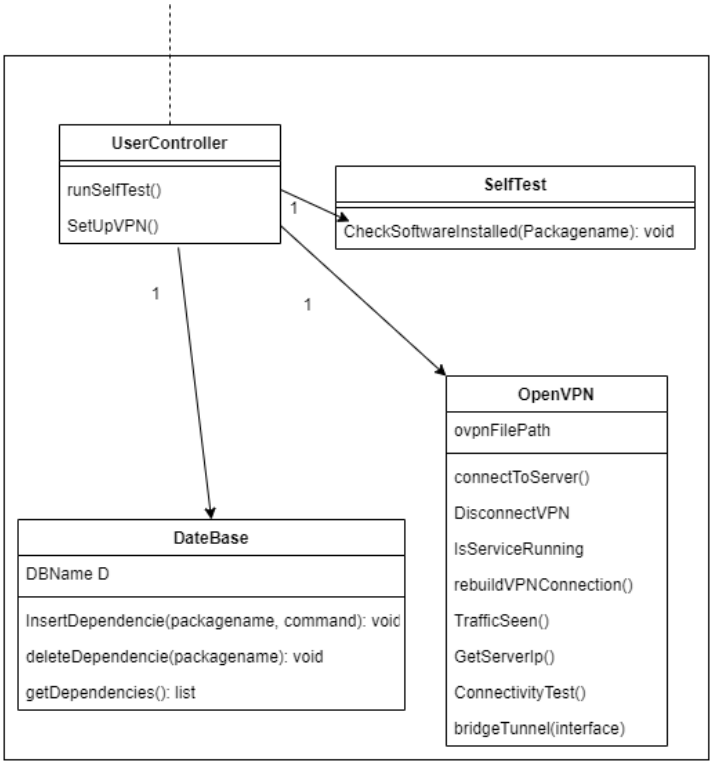
Figuur 7-2 Use-case 2

Id	Use-case 3
Use-Case	De computer hoeft geen directe verbinding te maken met het internet voor de USB VPN module om een VPN-tunnel op te zetten.
Brief Description	De computer hoeft nooit verbonden te zijn met het onbeveiligde internet om een veilige VPN-internetverbinding op te zetten.
Primary Actors	Computer
Secundary Actors	USB Armory
Pre-condition	- Computer is verbonden met de USB Armory. - USB Armory heeft beveiligde VPN-tunnel verbinding opgezet.
Main flow	1. Computer is verbonden met USB Armory via ethernet emulation 2. Computer stuurt internet verkeerd naar USB Armory 3. USB Armory routeert verkeer naar VPN-tunnel interface
Excepties	Geen
Post conditions	- Computer verkeerd wordt gerouteerd over de VPN-verbinding van de USB Armory

	- Internetverkeer wordt via een beveiligde VPN-verbinding verstuurd.
Alternative flows	Geen

Figuur 7-3 use-case 3

7.1.2 Klassendiagram



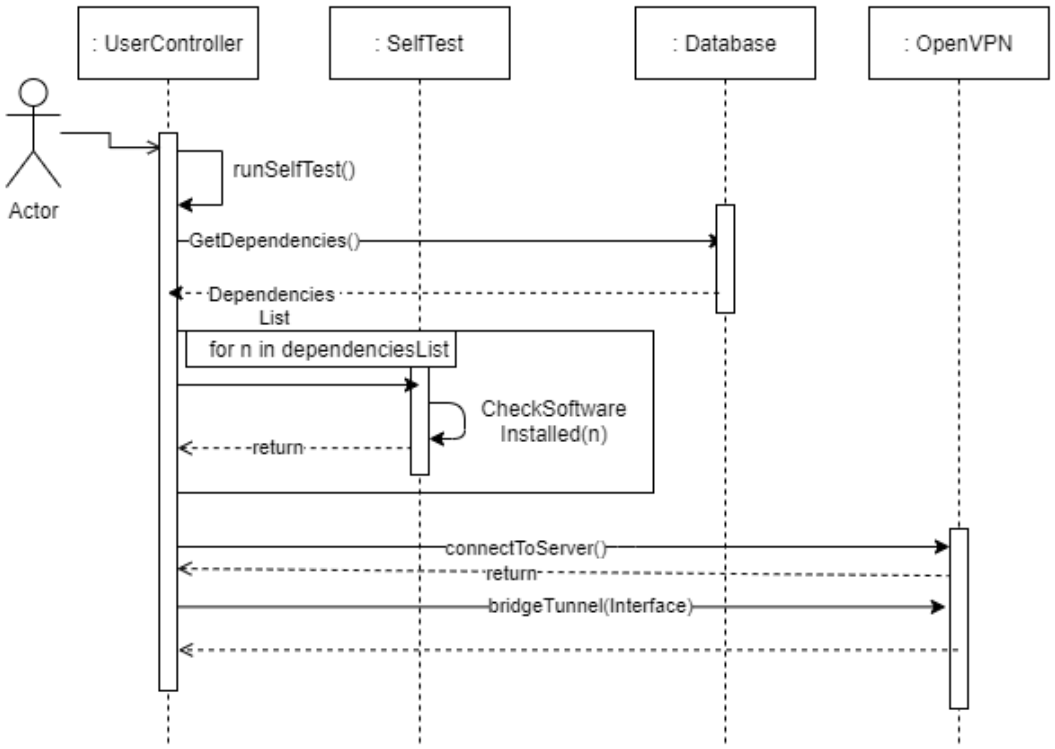
Figuur 7-4 klassendiagram iteratie 2/3

In figuur 7-4 is het klasse diagram te zien voor deze iteratie. Hierbij is de belangrijkste klasse OpenVPN die de OpenVPN-NL verbinding gaat beheren. Deze klasse is verantwoordelijk voor het inlezen van de OpenVPN-NL configuratie, en het op- en afbouwen van de VPN-verbinding.

Omdat het programma veel gebruik maakt van externe libararies (en als deze niet beschikbaar zijn externe packages) wordt er een simpele unitTest klasse gemaakt: “SelfTest”. Deze klasse controleert of alle packages op het systeem geïnstalleerd zijn, als deze niet geïnstalleerd zijn geeft hij een error en sluit het programma. Hierbij wordt gebruik gemaakt van de database klasse om de lijst met dependencies op te vragen.

Het programma is geprogrammeerd volgens het Model View Controller (MVC) model. Hierbij is de usercontroller de binden controller klasse die alle losse “models” aanstuurt. In latere iteraties zal deze klasse/controller uitbreidt worden met meer functies die model-klasse aanstuurt.

7.1.3 Sequentie diagram



Figuur 7-5 Sequentie diagram voor iteratie 2

In figuur 7-5 is het sequentie diagram voor iteratie 2 te zien. De actor start het programma waarna de UserController start. Hierna vraagt deze eerst de Dependencies list op aan de database klasse met de functie GetDependencies(). Deze return deze in de vorm van een list object.

Hierna loept de UserController over de list met dependencies. Voor iedere dependencie roept hij de functie CheckSoftwareInstalled() functie aan van de SelfTest klasse. Als deze een false returnt (software niet geïnstalleerd) beëindigt de UserController het programma.

Als de CheckSoftwareInstalled() succesvol is doorlopen start hij de VPN service met ConnectToServer(). Hierbij leest deze functie zijn config file in, en maakt verbinding met de VPN-server. Als deze verbinding succesvol is opgezet koppelt hij de VPN-interface aan de interface van de computer, waarna deze een beveiligde internetverbinding heeft.

7.1.4 Gemaakte keuzes opslagmedia

Tijdens het project moeten er verschillende zaken zoals settings, benodigdheden en login data opgeslagen worden. Deze moeten iedere keer als het programma opstart is benaderd kunnen worden. In deze iteratie wordt voor het eerst gebruik gemaakt van een opslagmedia. Bij de ontwerpfase is gekeken naar de verschillende opties hiervoor. Hierbij was de optie:

- Als bestand /file based
    - Bijvoorbeeld XML of JSON, hierbij wordt de data naar weggeschreven naar een textbestand volgens een bepaald standaard gegevens formaat (bijvoorbeeld json.
- Database service
    - Op de server is een database geïnstalleerd (bijvoorbeeld MongoDB, of MySQL). Via SQL queries kan data hiernaar gelezen en weggeschreven worden.
- Single file SQL
    - Een Single file SQL is een enkel bestand en library die geen eigen process of management interface nodig heeft.

Hierbij zijn de volgende verschillen:

Features	File based	Database Service	Single File databata
Read/write	Traag, veel lees operaties nodig bij grotere hoeveelheid data	Snel	Tot 10000 Records snel.
Hoe bestaat op systeem	Als losse files	Geïnstalleerd als package/service	Enkele Database file
Schaalbaarheid	Slecht, performance verslecht naar mate bestand groter wordt	Goed.	Tot 10000 entries goed, hierna is er performance verlies
Benaderbaarheid	File inlezen/parsen	SQL Queries	SQL Queries
Beveiliging	Geen	Mogelijkheid te beveiligen op accountniveau	Mogelijkheid te beveiligen met wachtwoord

Figuur 7-6 verschillen tussen opslagmethodes

Uiteindelijk is er gekozen voor een single File Database. Specifiek voor SQLLite. De hoofdreden hiervoor is dat er niet extreem veel data opgeslagen gaat worden, dus een traditionele sql database was niet logisch. Ook heeft de USB Armory een beperkte snelheid met een 900 MZ clock snelheid op de CPU. Een single file database heeft hier de beste performance van de drie opties.

7.1.5 Keuzes aansturing OpenVPN

In eerste instantie was het plan om OpenVPN aan te sturen door een los child shellproces te creëren dat door middel van signalen aangestuurd kon worden. Echter naarmate de iteratie vorderde kwamen er meer en meer problemen naar voren met deze aanpak.

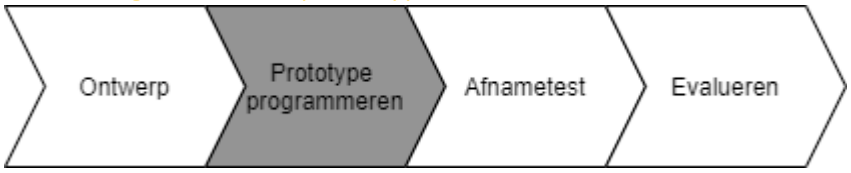
Het grootste probleem was dat de cliënt niet op consistente wijze gestopt kon worden. Als de OpenVPN verbinding onderbroken moest worden zou het process met een Terminate signaal beëindigt moeten worden. Echter is sommige gevallen reageerde het proces niet op een terminate signaal, waardoor het volledige programma via een Kill signaal beëindigt moest worden. Hierdoor bleef er rommel zoals routing entries achter die ervoor zorgde dat de armory eerst opnieuw opgestart moest worden voor deze weer goed functioneerde.

Een ander probleem wat naar voren kwam dat deze methode een CPU intensief was. De USB Armory heeft een single 900 MHz CPU core. Het multiprocessing process zorgde ervoor dat als het programma idle was 40% van de CPU verbruikt werd.

Het laatste probleem was dat het lastig was om de status van de OpenVPN client uit te lezen. Foutmeldingen werden naar de error pipeline van het subprocess geleid en naar de logs weggeschreven. Er was geen makkelijke manier om deze foutmeldingen te interpreteren in het programma.

Uiteindelijk werd ervoor gekozen om OpenVPN als service op de achtergrond te draaien. Hierbij zou er gebruik gemaakt worden van system resources om de benodigde informatie uit te lezen, en OpenVPN aan te sturen. Welke resources hiervoor gebruikt waren valt terug te lezen in hoofdstuk 7.2

7.2 Programmeren prototype



In dit hoofdstuk wordt beschreven hoe de klassen in deze iteratie ontwikkeld zijn.

selfTestklasse

Bij het ontwikkelen van het programma wordt er getracht zo veel mogelijk bestaande APi's en libraries te gebruiken. Echter bestaan packages met de gewenste functionaliteit soms niet, of zijn deze niet up to date. In dat geval wordt er met behulp van bestaande packages (Linux variant van programma's) gewerkt. Het doel van de SelfTest klasse is te controleren dat deze packages ook geïnstalleerd zijn.

```
def checkSoftwareInstalled(self, packageName):
    try:
        devnull = open(os.devnull, "w")
        retval = subprocess.call([packageName], stdout=devnull, stderr=subprocess.STDOUT)
        devnull.close()
        return True
    except FileNotFoundError:
        return False
```

Figuur 7-7 implementatie checksoftware()

Om er zeker van te zijn dat het programma goed functioneert test de SelfTest klasse of de package geïnstalleerd is. Dit doet het door een system call te doen met het commando naar de shell (zie figuur 7-7). Als het commando niet bestaat/goed geïnstalleerd is zal er een FileNotFoundError exception plaatsvinden en weggeschreven worden naar de error pipeline. Als dit het geval is returnt de functie een False en beëindigt het programma zichzelf.

```
equinox@apollo:~/hi$ pwd
/home/equinox/hi
equinox@apollo:~/hi$ pwdd

Command 'pwdd' not found, did you mean:

  command 'pwd' from deb coreutils
  command 'pwdx' from deb procps
  command 'pdd' from deb pdd
  command 'pldd' from deb libc-bin

Try: sudo apt install <deb name>

equinox@apollo:~/hi$
```

Figuur 7-8 goed commando versus slecht commando

OpenVPN klasse

Standaard is er niet een API (of andere makkelijke manier) voor het aansturen van de status van de OpenVPN client. De opties hiervoor waren:

1. OpenVPN client starten met de juiste parameters als los proces

2. OpenVPN client als service draaien en hierbij de juiste parameters uit een .ovpn config file laten lezen.

Uiteindelijk is er gekozen voor de tweede optie. De hoofdredenen hiervoor waren betere prestaties en het makkelijker kunnen aansturen van OpenVPN. De volledige redenering hiervoor valt terug te lezen in hoofdstuk 7.1.5, keuzes aansturing OpenVPN.

Om de service aan te sturen wordt er gebruik gemaakt van een combinatie van:

- Systemctl;
  - De Linux servicemanager, dezewordt gebruikt om de OpenVPN Service te starten, te stoppen, en de status uit te lezen;
- OpenVPN management client,
  - Een local interface die via SSH benadert kan worden. Hier kan beperkte informatie uitgelezen worden, zoals VPN metric's;
- OpenVPN config file.
  - Wordt door de OpenVPN service gebruikt om te zien met welke server en parameters die moet verbinden.

```
def DisconnectVPN(self):
    print("stopping openvpn...")
    devnull = open(os.devnull, "w")
    valret = subprocess.run("systemctl stop openvpn", stdout=devnull, stderr=subprocess.STDOUT, shell=True)
    devnull.close()
    if(valret.returncode):
        raise Exception("unable to stop VPN server, errorcode: ", valret.returncode)
```

Figuur 7-9 DisconnectVPN functie broncode

In figuur 7-9 is een voorbeeld te zien hoe er gebruik gemaakt wordt van system resources om de OpenVPN client aan te sturen. Door een systemcall te doen naar de Service Manager van Linux (systemctl) wordt er door het systeem een signaal gestuurd dat de service zichzelf moet stoppen. De OpenVPN service is dan zelf verantwoordelijk dat de service op goede wijze beëindigd wordt.

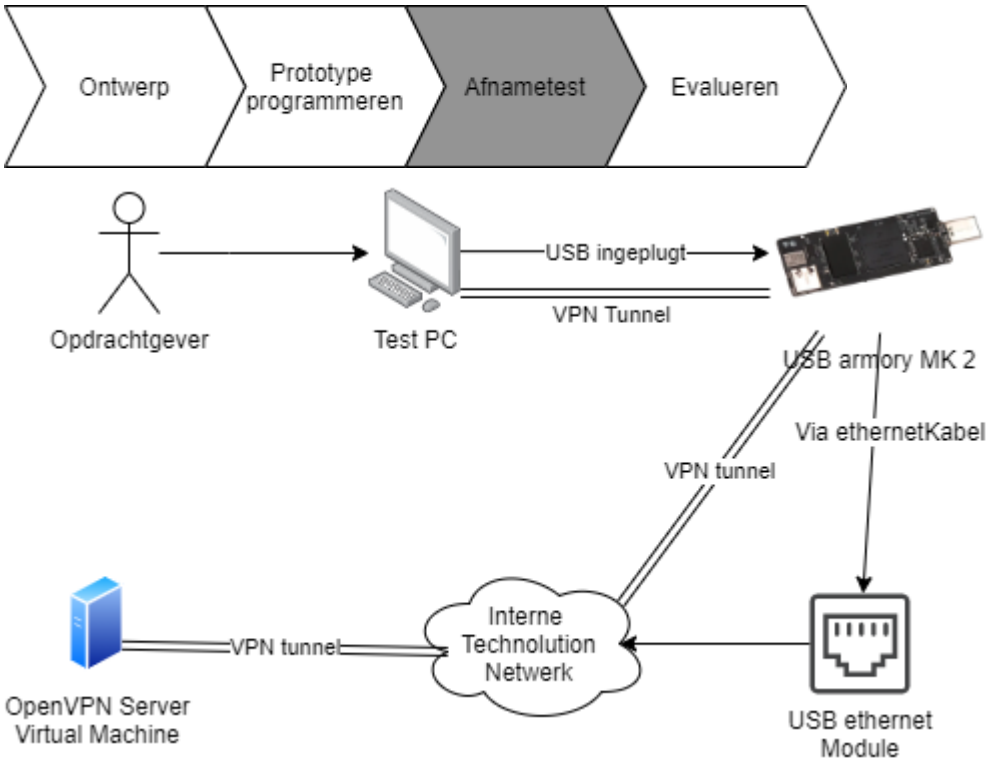
Nadat de OpenVPN tunnel is opgezet kan de USB armory deze delen met de hostcomputer door de **iptables** aan te passen. De implementatie hiervan is te zien in figuur 7-10.

```
devnull = open(os.devnull, "w")
# TODO check commands if valid and if bridging still works.
commands= ['sysctl -w net.ipv4.ip_forward=1',
           'iptables -F; iptables -t nat -F; iptables -t mangle -F',
           'iptables -t nat -A POSTROUTING -o {} -j MASQUERADE'.format(self.vpnInterface),
           'iptables -I FORWARD -o {} -s 10.0.0.0/24 -j ACCEPT'.format(self.vpnInterface),
           'iptables -I INPUT -s 10.0.0.0/24 -j ACCEPT',
           'iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT',
           'iptables -A INPUT -m state --state NEW ! -i  {} -j ACCEPT'.format(self.vpnInterface),
           'iptables -P INPUT DROP   #only if the first two are succesful',
           'iptables -A FORWARD -i {} -o {} -j REJECT'.format(self.InternetEdgeInterface,
                                                              self.InternetEdgeInterface),
           ]
print("bridging edge interface to vpn interface")
for x in commands:
    # x = x.replace("{}", interface)
    print('running command:', x)
    subprocess.run(x, shell=True, check=True)
devnull.close()
```

Figuur 7-10 implementatie bridgeTunnel()

De commando's om het verkeer van de host netwerkinterface naar de VPN netwerk interface door te sturen staan in de list **commands**. Deze worden via een loop met subprocess .run()uitgevoerd. De **check=true** flag zorgt ervoor dat er een exceptie wordt aangemaakt als één van deze commando's zou falen.

7.3 Afnametest



Figuur 7-11 testopstelling iteratie 2/3

In figuur 7-11 is de testopstelling te zien. Hierbij was de USB Armory aangesloten via USB aangesloten op een lokale computer. Via een USB Ethernet Module was hij verbonden met het interne Technolution netwerk. Op een virtual machine draait de OpenVPN-NL server waar de USB Armory mee verbond.

Tijdens de demonstratie plugde de opdrachtgever de USB amory in op de test pc. Zoals te zien in figuur 7-12 kwam er na enkele momenten er een nieuwe VPN-verbinding binnen op de openVPN-NL server.

```
ovpn-server[698]: 172.16.111.3:39523 [client01] Peer Connection Initiated with [AF_INET]172.16.111.3:39523
ovpn-server[698]: client01/172.16.111.3:39523 MULTI_sva: pool returned IPv4=10.8.0.2, IPv6=(Not enabled)
ovpn-server[698]: client01/172.16.111.3:39523 MULTI: Learn: 10.8.0.2 -> client01/172.16.111.3:39523
ovpn-server[698]: client01/172.16.111.3:39523 MULTI: primary virtual IP for client01/172.16.111.3:39523: 10.8.0.2
ovpn-server[698]: client01/172.16.111.3:39523 PUSH: Received control message: 'PUSH_REQUEST'
ovpn-server[698]: client01/172.16.111.3:39523 SENT CONTROL [client01]: 'PUSH_REPLY,dhcp-option DNS 176.103.130.130,dhcp-
y subnet,ping 10,ping-restart 120,ifconfig 10.8.0.2 255.255.255.0,peer-id 0,cipher AES-128-GCM' (status=1)
ovpn-server[698]: client01/172.16.111.3:39523 Outgoing Data Channel: Cipher 'AES-128-GCM' initialized with 128 bit key
ovpn-server[698]: client01/172.16.111.3:39523 Incoming Data Channel: Cipher 'AES-128-GCM' initialized with 128 bit key
systemd[1]: pcscd.service: Succeeded.
```

Figuur 7-12 output logfiles VPN Server

Via het commando **ip addr show dev tun0** kon geverifieerd worden dat de USB Armory diegene was die de VPN-tunnel op had gezet

```
usbarmory@usbarmory:~$ ip addr show dev tun0
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
    link/none
    inet 10.8.0.2/24 brd 10.8.0.255 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::3b57:7764:e4e9:3015/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Figuur 7-13 output ip addr show dev tun0

De opdrachtgever probeerde hierna via de test pc een website te bezoeken. Om te verifiëren dat het verkeer ook echt via de VPN verstuurd werd draaide er een Wireshark packet scan op de VPN-interface van de VPN-server. In tabel 7-13 en 7-14 is de samenvatting van de afname test te zien.

Test	Use-case 2
Use-Case	De usb VPN-module probeert een secure VPN-tunnel op te zetten
Brief Description	De USB Armory verbinding met de OpenVPN-NL server, en zet een VPN-tunnel op
Pre-condition	- USB Armory heeft een internetverbinding
Test process	1. Steek de USB armory in de computer 2. Wacht ongeveer 30 seconden tot de USB armory is opgestart 3. USB Armory zet VPN-tunnel op 4. USB Armory is verbonden met OpenVPN server
Verwachte uitkomst	Op de OpenVPN-NL server is te zien dat de USB armory verbinding heeft gemaakt en er verkeer over de VPN-tunnel plaats kan vinden.
Uitkomst	Uitkomst zoals verwacht

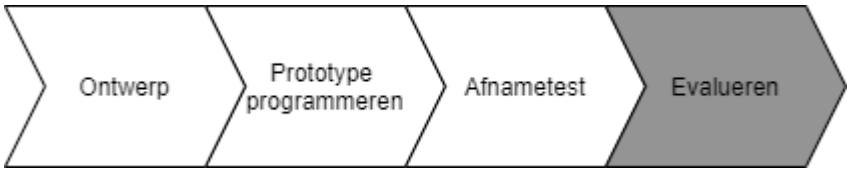
Figuur 7-14 afnametest use-case 4

Test	Use-case 3
Use-Case	De computer hoeft geen directe verbinding te maken met het internet voor de USB VPN module om een VPN-tunnel op te zetten.
Brief Description	De computer hoeft geen directe verbinding te maken met het internet voor de USB VPN module om een VPN-tunnel op te zetten.
Pre-condition	- De testcomputer is opgestart,
Test process	1. Steek de USB armory in de computer 2. Wacht ongeveer 30 seconden tot de USB armory is opgestart 3. Check of de computer de armory herkent als Ethernet adapter 4. Verifieer dat de USB Armory een actieve internetverbinding geeft door een willekeurige website te bezoeken.
Verwachte uitkomst	Nadat de USB armory een VPN-tunnel heeft opgezet heeft de testcomputer een actieve internetverbinding. Op de OpenVPN server is het webverkeer van de testcomputer te zien.
Uitkomst	Uitkomst zoals verwacht

Figuur 7-15 afnametest use-case 3

De afname test verliep zoals verwacht. De testcomputer was voorzien van een actieve internetverbinding zonder dat deze eerst met het “onveilige internet” hoefde te verbinden.
De opdrachtgever kon succesvol een website op de testcomputer bezoeken, en het verkeer hiervan was live te zien op de OpenVPN server via wireshark. Om deze redenen is deze iteratie succesvol afgerond.

7.4 Evalueren afnametest



Aan het einde van de iteratie wordt het prototype verbeterd aan de hand van de feedback verkregen bij de demonstratie. De opdrachtgever was tevreden met het resultaat. Hij had verder geen op- of aanmerkingen.

8 Ontwikkelingsfase: Iteratie 4

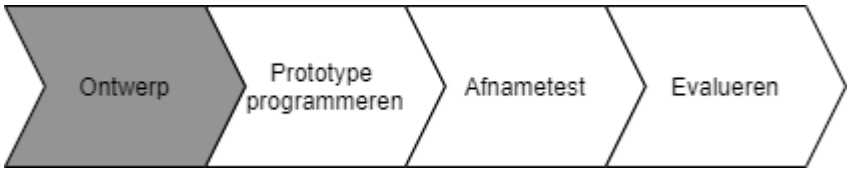
In iteratie 4 werd de volgende eis behandeld.

- FE05, de computer kan niet direct bij de VPN-module komen, de configuratie moet op een indirecte manier plaatsvinden.

Bij deze eis is het belangrijk dat er een gebruikersvriendelijke manier is om de VPN-module te kunnen configureren, zonder dat de gebruiker direct verbinding moet maken met het systeem (via bijvoorbeeld een SSH-connectie). Uiteindelijk is er gekozen om de configuratie plaats te laten vinden via een webapp. Tijdens deze iteratie konden de werkzaamheden in twee delen gesplitst worden.

- Het maken van een GUI/web app voor het aansturen en configureren van de USB Armory
- Het maken van de klasse die de GUI aanroept om de Armory de configureren.

8.1 Ontwerp



Om de eis te realiseren moest er iets verzonnen worden zodat de Armory op een indirecte en veilige manier geconfigureerd kon worden. In voorgaande gesprekken gaf de opdrachtgever aan dat hij liever geen software hiervoor wilde installeren. Het eindproduct moet zo veel mogelijk plug-and-play blijven zonder dat er extra stappen aan te pas zou komen.

Hierbij moest de configuratie methode dus:

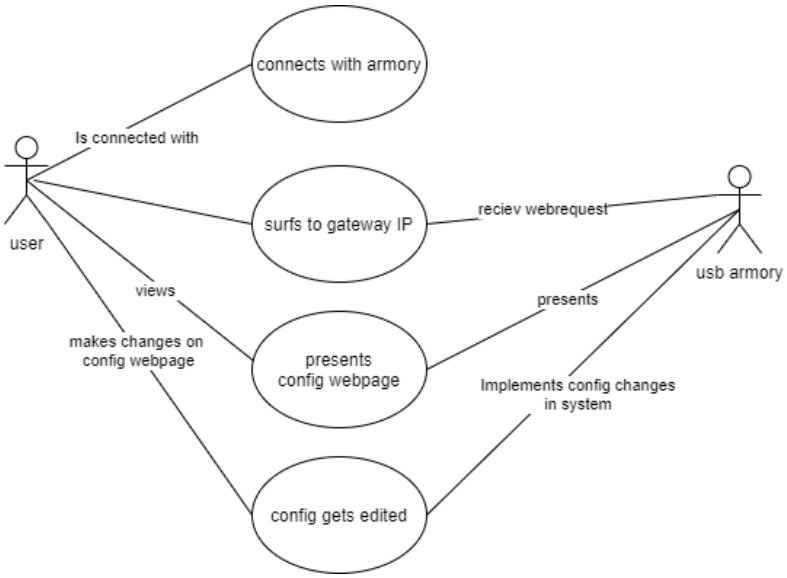
- Geen direct verbinding met de USB Armory hebben;
- Gebruikersvriendelijk zijn;
- Geen extra software-installatie vereist zijn.

Uiteindelijk is ervoor gekozen om de configuratie via een web app GUI te laten verlopen. Dit is de meest voor de hand liggende oplossing. De gebruiker hoeft hiervoor geen extra software te installeren, en kan op een makkelijke gebruikersvriendelijke manier de configuratie aanpassen. Ook is de computer van de gebruiker al verbonden met de USB Armory. Een webserver draaien is daarom geen probleem

8.1.1 Gebruiksdiagram.

Een representatie hoe de configuratie van de web app moet functioneren is te zien in figuur 8-1. De gebruiker verbind met de USB Armory en surft naar het gateway ip (ip van de armory). Hier wordt en

configuratie pagina gepresenteerd. De wijzigingen die de gebruiker hier invoert en submit wordt volgens door de USB Armory geïmplementeerd.



Figuur 8-1 Gebruiksdiagram iteratie 4

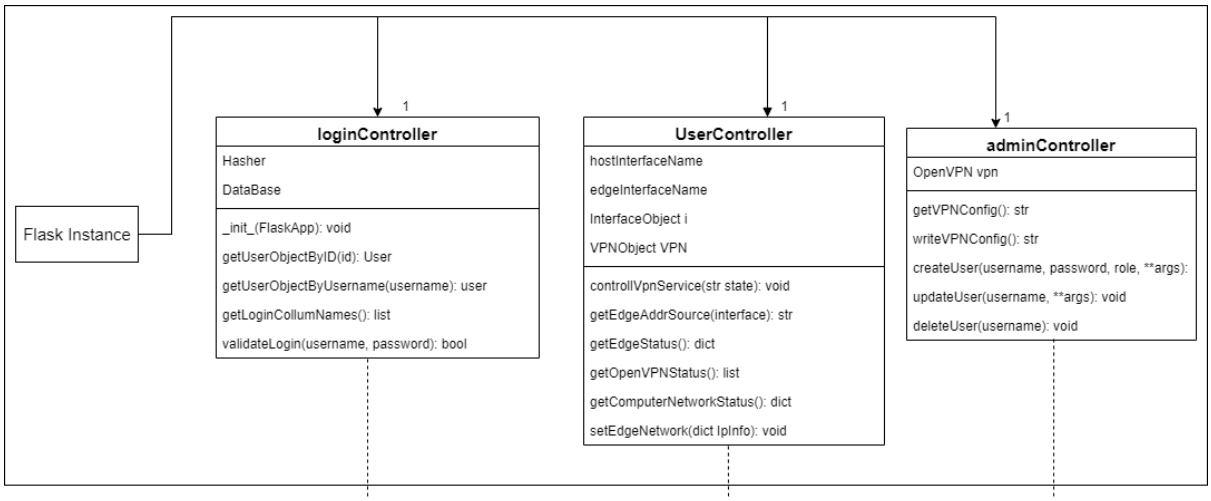
In de tabel hieronder de is de use-case voor FE05.

Id	Use-case 4
Use-Case	De computer kan niet direct bij de VPN-module komen, configuratie moet op een indirecte manier plaatsvinden.
Brief Description	Een indirecte manier van configuratie in de vorm van een webapp
Primary Actors	Eindgebruiker
Secundary Actors	USB Armory
Pre-condition	USB Armory is opgestart en verbonden met computer
Main flow	1. Eindgebruiker surft naar default gateway IP in webbrowser 2. Eindgebruiker logt in op Web app. 3. Eindgebruiker wijzigt configuratie op Armory 4. USB Armory voert configuratie wijzigen door in systeem
Alternative flow	2a. Authenticatie faalt. 2b. Gebruiker wordt teruggestuurd naar login screen met error
Post conditions	Configuratie op USB Armory is aangepast.
Alternative flows	Geen

Figuur 8-2 use-case 4

8.1.2 Klassendiagram view/controllers

Voor het ontwikkelen van de webapp wordt gebruik gemaakt van Flask. In Flask wordt ontwikkeld volgens een **Model View Controller model (MVC).** In de web app komen alle klasse van de vorige iteraties samen. Ook worden er meerdere nieuwe klassen geïntroduceerd. In figuur 8-3 een representatie van de View en Controllers (zonder hun models). Hierbij is de Flask instance verantwoordelijk voor de view weergaves. Voor de aansturing van de models en het leveren van informatie voor de views wordt gebruik gemaakt van een drie-tal Controller klassen.

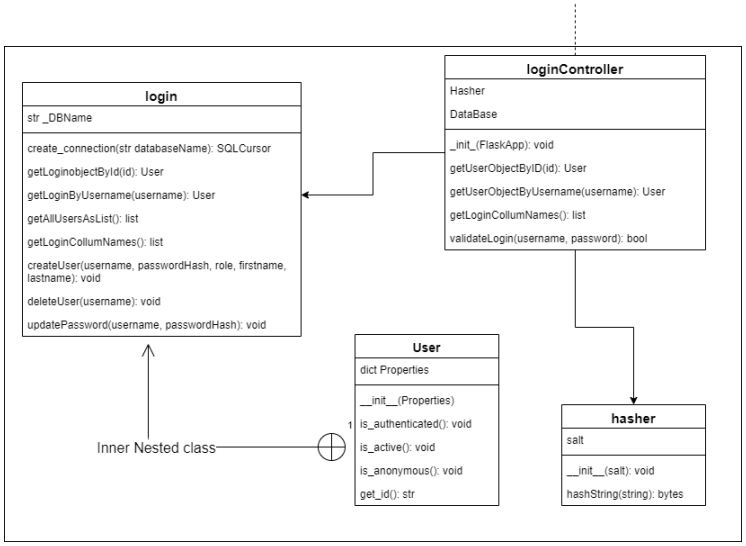


Figuur 8-3 view en controllers

Hierbij zijn de controllers verantwoordelijkheid voor het volgende:

- De **loginController** is verantwoordelijk voor alle functies gerelateerd aan het login proces zoals authenticatie.
- De **userController** is verantwoordelijk voor het weergeven van data en functies die een normale gebruiker zou kunnen zien en uitvoeren, zoals het weergeven van het dashboard en start/stoppen van de vpn service.
- De **adminController** is verantwoordelijk voor alle functies die enkel beschikbaar zijn voor beheerders. Bijvoorbeeld het wijzigen van de VPN-configuratie.

8.1.3 Klassendiagram: Logincontroller



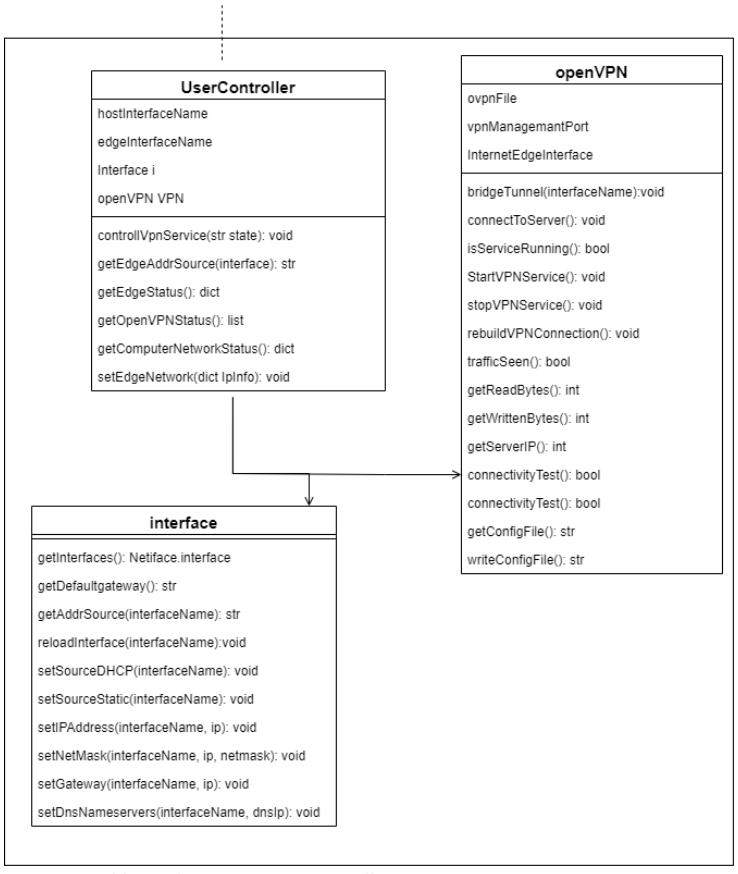
De loginController is verantwoordelijk voor het login/authenticatie proces. Hij heeft twee models: login en hasher.

De Login model bevat de functies voor authenticatie en het managen van de users.

De Hasher Model wordt gebruikt voor het hashen van het wachtwoord.

Figuur 8-4 klassendiagram loginController

8.1.4 Klassendiagram: userController



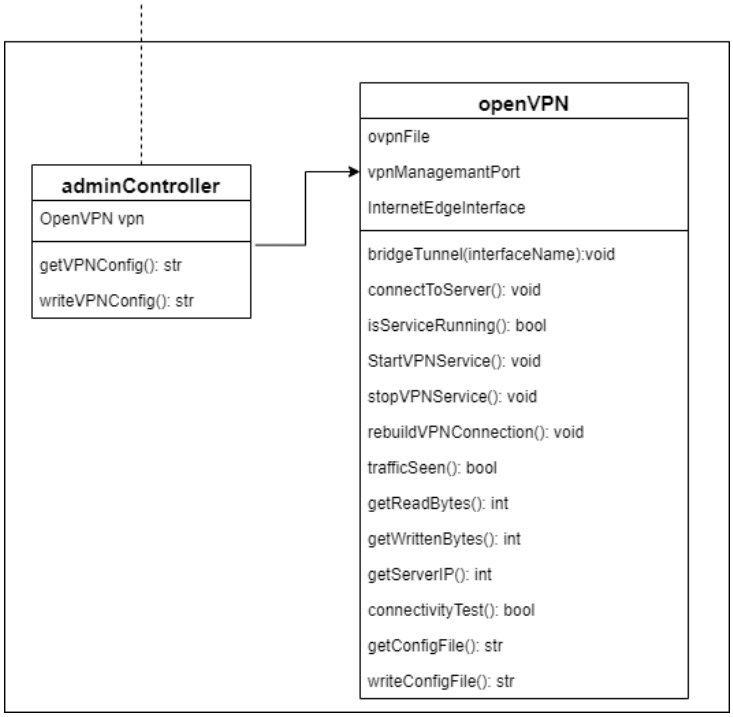
De userController is verantwoordelijk voor alle functies die een gebruiker kan uitvoeren. Deze functies komen terug in de statusPage view. Hiervoor heeft het twee models.

De OpenVPN klasse beheert het OpenVPN process. De hoofdtaak hierin is het stoppen en starten van OpenVPN-NL en het weergeven van de status

De interface klasse beheert de IP-instellingen. Het kan de huidige IP-instellingen uitlezen en weergeven op de statusPage view. Voor de networksettingsView heeft het functies als setIPaddress() om de huidige IP-instellingen aan te passen.

Figuur 8-6 klassediagram userController

8.1.5 Klassendiagram: adminController



Figuur 8-7 klassendiagram adminController

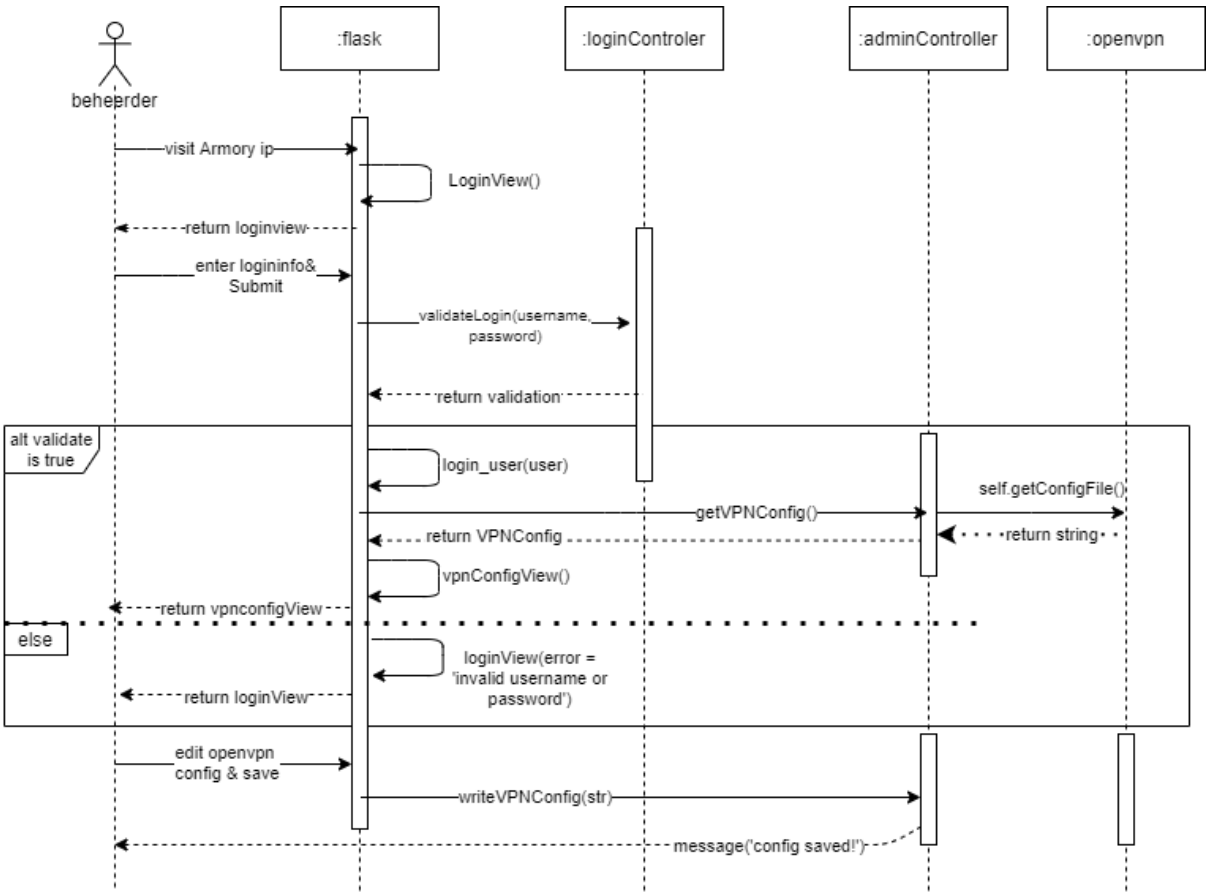
De adminController klasse is verantwoordelijk voor het opvragen en aanpassen van configuratie van beheerders-pagina's.

In deze iteratie wordt de klasse momenteel enkel gebruikt om de OpenVPN-NL configuratie aan te passen. Als er in een latere iteratie behoefte is aan meer admin functionaliteiten dan kan deze klasse hiervoor uitgebreid worden.

8.1.6 Sequentie diagram

In de onderstaande afbeelding is een sequentie diagram te zien van een beheerder die inlogt op de armory, en de OpenVPN-NL configuratie aanpast. De webserver verzoeken komen binnen bij het Flask object. Hierbij is er een alternatieve flow voor als het login proces mislukt. In dit geval stuurt Flask de gebruiker terug naar de login pagina met een error.

Als de login succesvol is vraagt Flask de juiste informatie op bij het OpenVPN object via de adminController. Deze wordt vervolgens getoond in de OpenVPN Config view.



Figuur 8-8 sequentie diagram iteratie 4

8.1.7 Keuze programmeren model view controller

Het programmeren van de web app kan snel onoverzichtelijk worden. Er wordt gewerkt met veel losse onderdelen die allemaal met elkaar moeten kunnen samenwerken. Om het programma overzichtelijk te houden en de kwaliteit van de code te kunnen garanderen wordt er gebruik gemaakt van een Model View Controller model (vanaf nu MVC genoemd).
Een korte omschrijving van wat een MVC inhoudt.

- De **Model** zijn de configuratie klassen die acties uitvoeren aan de hand van de commando's verkregen van de **Controller;**
    - Bijvoorbeeld**:** controller stuurt DeleteUser(username) naar Database model, model verwijderd vervolgens user in de database.
- De **View** (webpage) genereert output voor de gebruiker gebaseerd op veranderingen in het model;
    - Bijvoorbeeld: Controller stuurt de status van de OpenVPN service naar de View, de view (in de vorm van de webpagina) weergeeft deze vervolgens aan de gebruiker.
- De Controller handelt als een tussenstuk tussen de **View** en de **Models.** Het stuurt commando's naar de View om de informatie die aan gebruikers wordt gepresenteerd te wijzigen. Het stuurt ook commando's naar de **Models** om hun acties uit te laten voeren.

8.1.8 Keuze framework web app

Voor het ontwikkelen van de web app moest een framework gekozen worden om python code en HTML/CSS samen te laten komen. Hierbij zijn er voor python twee populaire web frameworks:

- **Django:** Django is een full website framework voor Python. Het is een high-level webframework voor rapid development. Alle functionaliteiten die je nodig hebt voor een website zitten “out of the box” verwerkt in Django. Het primaire doel van dit webframework is om complexe database gestuurde websites te maken.
- **Flask,** Flask is een python micro-framework dat basis functionaliteiten van een webapp biedt. Standaard heeft het vrij minimalistische functionaliteiten. Door middel van libraries kan het uitbreidt worden met extra functionaliteiten. Het hoofddoel van Flask is om op snelle en flexibele wijze een website te bouwen.

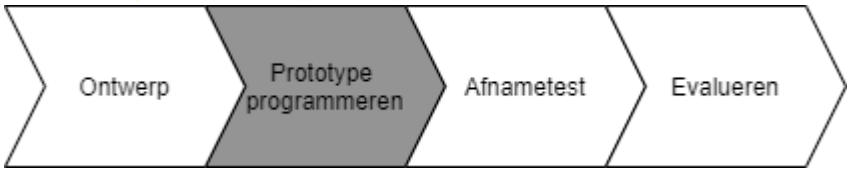
Er is gekozen om Flask te gebruiken. Beide webframeworks functioneren vrijwel hetzelfde. Bij het uittesten bleek de learning curve van Django vrij hoog te zijn. Met geen ervaring in het programmeren met webframeworks zou het enkele weken duren voor hier genoeg handigheid in behaald zou zijn. Flask heeft een simpelere API en lagere learning curve. Hierdoor kon in kortere tijd een functionele web app gemaakt worden.

8.1.9 Keuze vormgeving Web app.

Een volledige website vanaf de grond opbouwen zou zeer tijd consumerend zijn. Om tijd te besparen is ervoor gekozen om gebruik te maken van de front-end toolkit Bootstrap en een kant klare bootstrap template. Bootstrap is een CSS-engine waarmee de CSS automatisch afgehandeld wordt door forms, tables, en dergelijke te wrappen in een bootstrap class.

Voor de template is er gekozen voor SB admin. Dit is een gratis voorgebouwde template die aangepast kon worden naar de wensen en eisen van het project.

8.2 Programmeren prototype



Voor het programmeren van het prototype werden alle klasse die gemaakt waren in de vorige iteraties geïmplementeerd. In dit hoofdstuk wordt het verloop hiervan besproken

Flask en de webtemplate

Voor het opzetten van Flask hoefde weinig te gebeuren. Via de python package manager **pip** kon de Flask package geïnstalleerd worden. Door de het starten van het Flask example script kon er geverifieerd worden dat Flask out of the box werkte.

8.2.1 Views

Voor deze iteratie werden er drie views gemaakt (webpages): De login page, statuspagina, en de Network settings. Voor een volledig overzicht van deze views zie **bijlage G: web app pagina.**

Login pagina view

De loginpagina van de bootstrap template was iets te opzichtig. Om deze reden was er een simpele pagina gemaakt met het logo van Technolution. De login pagina bestaat uit een Flask-wtf form met een tekst field voor de login en een password field voor het wachtwoord. Hierbij wordt gebruik gemaakt van de wtf-Flaskform package. Hierbij vereist wtf-flaskform dat het formulier als losse klasse geïmplementeerd worden (zie figuur 8-9). In de loginForm implementatie wordt er met wtf.validators ervoor gezorgd dat de de zowel het login als het passwordfield verplicht in moet vullen.

```
class loginForm(FlaskForm):
    username = StringField('Name:', validators=[validators.required()])
    password = PasswordField('Password:', validators=[validators.required(), validators.Length(min=3, max=35)])
```

Figuur 8-9 implementatie loginForm

Zodra de gebruiker het login form submit stuurt Flash deze naar de LoginController. Met de functie Auth.validateLogin() (zie figuur 8-10) valideert hij of de login gegevens goed zijn. Als dit het geval is redirect de loginView de gebruiker naar de statuspagina. Als dit niet het geval is dan redirect hij de gebruiker terug naar de loginpagina met een foutmelding.

```
if form.validate_on_submit():
    print(form.errors)
    print(request.form)
    username = request.form['username']  # request username from form
    password = request.form['password']  # request password from form
    # validate the username and password
    if Auth.validateLogin(username, password): #checks if password corrospo
        login_user(Auth.getUserObjectByUsername(username))  # authenticate
        UC.bridgeVPNTunnel()  # bridge the VPN tunnel
        return redirect(url_for('statuspage'))
    else:
        error = 'Invalid username or password'
# return login view, if error is not None, returns that one to login view
return render_template('login.html', error=error, form=form)
```

Figuur 8-10 loginView implementatie.

Status Pagina view.

De status pagina view is de hoofdpagina van de webapp. In één oogopslag kan de status van de VPN, de internetverbinding, en de verbinding tussen de armory en de hostcomputer uitgelezen worden. De informatie hiervoor verkrijgt hij via de userController. Onderaan de pagina zijn een drietal knoppen die de OpenVPN service kunnen starten, stoppen, of de pagina kunnen herladen.

Om deze gegevens goed te kunnen weergeven geeft Flask een aantal lists mee. Zoals te zien in figuur 8-11 verkrijgt hij deze via userController functies. Hierna worden ze als parameters meegegeven aan de view.

```
return render_template('StatusPage.html',
                       openVpnStatus=UC.getOpenVPNStatus(),
                       openVpnColor=UC.getVpnStatusColor(),
                       computerNetworkStatus=UC.getComputerNetworkStatus(),
                       EdgeNetworkStatus=UC.getEdgeStatus(),
                       EdgeNetworkColor=UC.getEdgeNetworkColor(),
                       popupHeader='captive portal!', popupMessage=message
                       )
```

Figuur 8-11 statuspage view in Flask

In de html pagina loopt hij vervolgens met behulp van een jinja2 loop (wordt gebruikt door Flask om html pages te renderen) door de list heen. Vervolgens maakt hij voor iedere item in de list een table entry aan.

```
<tbody>
    {% if EdgeNetworkStatus is not defined %}
    {% set EdgeNetworkStatus = {} %}
    {% endif %}
    {% for key, value in EdgeNetworkStatus.items() %}
    <tr>
        <th> {{ key }} </th>
        <td> {{ value }} </td>
    </tr>
    {% endfor %}
```

Figuur 8-12 jina2 loop

Netwerkinstellingen view

De netwerkinstellingen view is de pagina waar de netwerkinstellingen gewijzigd kunnen worden. Voor dit instellingen form wordt er gebruik gemaakt van de library **Flask-wtf**. Deze functies worden dan ook via het Flask jinja2 template “geïnjecteerd” als de HTML-code aan de gebruiker gepresenteerd wordt. De settings als IP en DHCP vragen de huidige IP-instellingen op via de UserController en geven deze als placeholder in de velden weer. Als de gebruiker settings probeert door te voeren worden door middel van regulaire expressies gecontroleerd of het een valide IP is. Als dit niet geval is krijgt de gebruiker de pagina weer terug met een foutmelding.

8.2.2 Models

Hasher

Wachtwoorden opslaan in plain tekst is een slechte gewoonte. Iedereen die toegang zou krijgen tot de database kan deze wachtwoorden zo uitlezen. Om een extra beveiliging laag toe te voegen worden de wachtwoorden opgeslagen via een ‘one way hash’ methode. De klasse **hasher** stelt de loginController instaat om wachtwoorden gehasht op te slaan in de database.

Zoals te zien in figuur 8-13 wordt hierbij gebruik gemaakt van de package hashlib. De functie hashString neemt een wachtwoord als input. Deze wordt vervolgens 100000 iteraties gehasht met het **sha256** algoritme. Om het wachtwoord te beschermen tegen een woordenlijst aanval wordt er gebruikt gemaakt van salt. Bij 'salt' wordt willekeurige data toegevoegd aan een hashfunctie. Hierbij gebruikt hij de secret string van de Flask app als salt.

```python
def hashString(self, string):
    key = hashlib.pbkdf2_hmac(
        'sha256',  # The hash digest algorithm for HMAC
        string.encode('utf-8'),  # Convert the password to bytes
        self.salt,  # Provide the salt
        100000  # It is recommended to use at least 100,000 iterations of SHA-256
    )
    return (key).decode('ISO-8859-1')
```

Figuur 8-13 implementatie hasher

Login en User Klasse

Voor het login functionaliteit in Flask wordt er gebruikt gemaakt van de extension **Flask-login.** Voor deze extensie moeten enkele functies geïmplementeerd worden. In eerste instantie zouden de attributen uit de database (user id, passwordhash, username) als dictonary item aan de functies doorgegeven worden. Benodigde attributen kon dan makkelijk opgevraagd worden (bijvoorbeeld Dict['id'] returnt ID, dict['username'] returns username). Als er in een latere fase van het project extra attributen aan de database moeten worden toegevoegd kunnen deze dynamisch aan het object worden toegevoegd.

Echter bleek dat **Flask-login** een vast aantal functies nodig heeft bij het userobject om te controleren of die actief is en om hem uit loggen. Om deze reden werd er een sub-klasse gemaakt genaamd User (zie figuur 8-14). Deze klasse functioneerde als een shell. Database attributen konden nog steeds dynamisch aan de **property** list toegevoegd worden. **Flask-login** kan de benodigde functionaliteit via het User object aanroepen.

```python
class User:
    property = None
    def __init__(self, params):
        if type(params) is dict:
            self.property = params
        else:
            raise TypeError('User constructor: params is not a dict!')
    def is_authenticated(self):
        return True
    def is_active(self):
        return True
    def is_anonymous(self):
        return False
    def get_id(self):
        return str(self.property['id'])
```

Figuur 8-14 implementatie user

OpenVPN klasse

De OpenVPN klasse verzorgt de aansturing van de OpenVPN-NL cliënt. OpenVPN heeft geen betrouwbare API of python library die gebruikt kon worden om de OpenVPN verbinding aan te sturen. In de eerste iteratie van deze klasse OpenVPN aangestuurd worden door middel van een child process. OpenVPN-NL zou in het child process opgestart worden met de juiste parameters (zoals configuratie). Om het process te stoppen zou het child process beëindigd worden door middel van een flag.

Echter bleek dit in de praktijk niet haalbaar te zijn. Het process reageerde soms niet goed op het signaal om hem te beëindigen waardoor hij via een SIGKILL (kill -9 <PID>) beëindigt moest worden. Hierna moest de armory eerst weer opnieuw opgestart worden om alle restanten van het OpenVPN process op te ruimen. Ook was er het probleem dat OpenVPN als child process geen betrouwbare methodes had om errors en de status van de VPN-verbinding uit te lezen.

In de tweede iteratie van de OpenVPN klasse draait de OpenVPN verbinding als losse service op het systeem. Door middel van creatief gebruik van system resources wordt de OpenVPN service aangestuurd. Zo maakt het gebruik van:

- systemctl, om het OpenVPN process te starten , stoppen en de status uit te lezen;
- Editen van de ovpn config file om de config aan te passen;
- Uitlezen van managementpoort om te controleren of er verkeer heeft plaatsgevonden.

In figuur 8-15 is hier een implementatie voor te zien. Om de VPN-service te starten wordt er een subprocess gespawnt die het commando **systemctl start OpenVPN** uitvoert. Als systemctl een andere code dan 0 (succes) returnt dan wordt er een exception opgeworpen met de foutcode.

```
def StartVPNService(self):
    """
    starts the VPN service.
    """
    print("starting openvpn...")
    devnull = open(os.devnull, "w")
    valret = subprocess.run("systemctl start openvpn", stdout=devnull, stderr=subprocess.STDOUT, shell=True)
    devnull.close()

    if(valret.returncode):
        raise Exception("unable to start VPN server, errorcode: ", valret.returncode)
```

Figuur 8-15 implementatie voor vpn start

Interface klasse

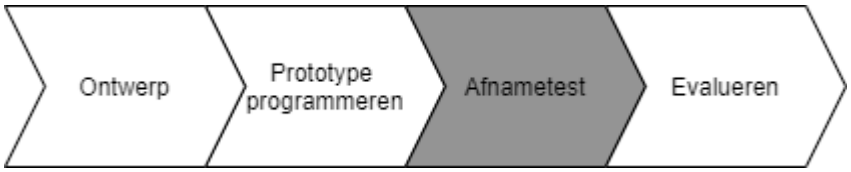
De interface klasse is verantwoordelijk voor het veranderen en uitlezen van de IP-instellingen van zowel de host interface, als de edge interface. Bij het maken van deze klasse was het kritisch dat er op betrouwbare wijze het IP-adres uitgelezen kon worden. Andere klasse zouden op deze verkregen informatie door kunnen bouwen.

Voor het uitlezen van de IP-instellingen zou er gebruik gemaakt worden van een python library. Tijdens onderzoek bleken veel van de python packages die deze functionaliteit verzorgde al een decennia niet meer bijgewerkt te zijn. Uiteindelijk bleek de package **Netifaces** het ip adres betrouwbaar uit te kunnen lezen.

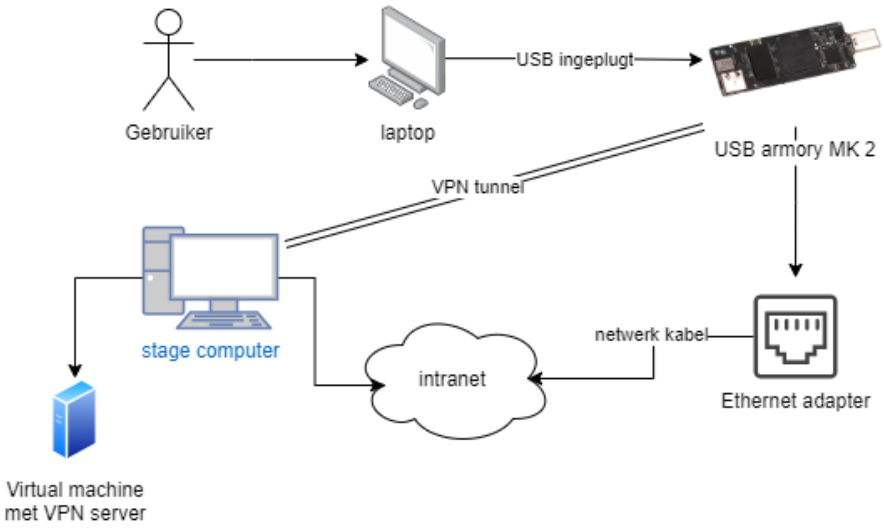
Voor het configureren van het IP-adres was ervoor gekozen om de interfaces file in /etc/networks/interfaces aan te passen. De python package **debinterfaces** kon de interfaces betrouwbaar parsen, en configuratie wijzigingen wegschrijven.

In deze iteratie werkt de interface klasse momenteel enkel met een bekabelde verbinding. In iteratie vijf wordt de klasse uitgebreid zodat de gebruiker ook een draadloos netwerk kan selecteren.

8.3 Afname test



Op 5 januari vond de demonstratie voor de opdrachtgever plaats. Voor de demonstratie was de USB Armory ingeplugd in een test laptop. In figuur 8-16 is een afbeelding te zien van de testopstelling. Op de werk pc draaide een virtualmachine waar een VPN-server op geïnstalleerd was. De USB Armory zetten een VPN-tunnel op via het interne netwerk.



Figuur 8-16 test opstelling voor iteratie 5

Tijdens de demonstratie kon de opdrachtgever wel alle netwerkinstellingen (ip, subnet, gateway, en dns) aanpassen, maar niet individuelen instellingen. De dag ervoor was er form validatie geïmplementeerd welk dit blokkeerde.

Toen de opdrachtgever naar de user managementpagina wilde gaan zorgde dit voor een error. Dit kwam omdat de route in Flask niet goed stond ingesteld. Na de afname test is deze bug verholpen.

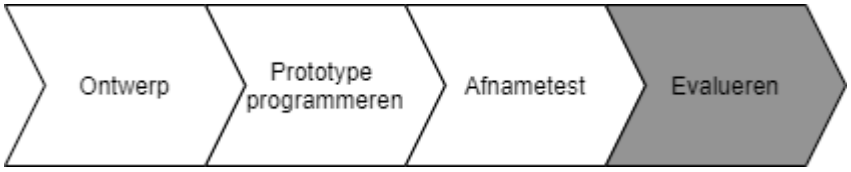
Afgezien van deze bug was de opdrachtgever tevreden met de voortgang. De eisen van deze iteratie zijn succesvol gedemonstreerd. De iteratie is met succes afgesloten. In tabel 8-17 is een samenvatting terug te lezen.

Test	Use-case 4
Use-Case	De computer kan niet direct bij de VPN-module komen, configuratie moet op een indirecte manier plaatsvinden.
Brief Description	Een indirecte manier van configuratie in de vorm van een webapp
Pre-condition	- USB Armory heeft een internetverbinding - USB Armory is in test pc geplugd - USB Armory is opgestart
Test process	1. Eindgebruiker surft naar default gateway IP in webbrowser 2. Eindgebruiker logt in op Web app. 3. Eindgebruiker wijzigt configuratie op Armory (in dit geval, veranderen van ip addres)

	4. USB Armory voert configuratie wijzigen door in systeem
Verwachte uitkomst	Als de configuratie wijziging doorgevoerd wordt is ook op het systeem terug te vinden (ip wijzigingen te controleren via het commando **ip adres)**
Uitkomst	De opdrachtgever kon vanwege een validator bug niet alle velden op het netwerk configuratie

Figuur 8-17 afnametest use-case 5

8.4 Evalueren afnametest



Bij de demonstratie kwamen enkele bug's naar voren. Deze werden na de meeting opgelost. Ook gaf de opdrachtgever een aantal suggesties voor verbeteringen aan die die graag geïmplementeerd zou zien.

De grootse aanmerking die hij had was dat het product momenteel genoeg technische opties (OpenVPN kunnen wijzigen, ip, DNS), en mogelijk zelfs iets te veel. Voor een niet-technische gebruiker zou het in 1 opslag duidelijk moeten zijn wat de status is. Hiervoor raadde de opdrachtgever aan om de statusbalken in het dashboard aan te passen met kleuren. Deze verbetering werd in de opeenvolgende iteratie geïmplementeerd.

9 Ontwikkelingsfase: Iteratie 5

In iteratie 5 wordt de volgende eis behandeld:

- FE06: Wi-Fi hotspot selectie/verbinding vindt plaats op de VPN-module, de gebruiker mag wel indirect deze selecteren.

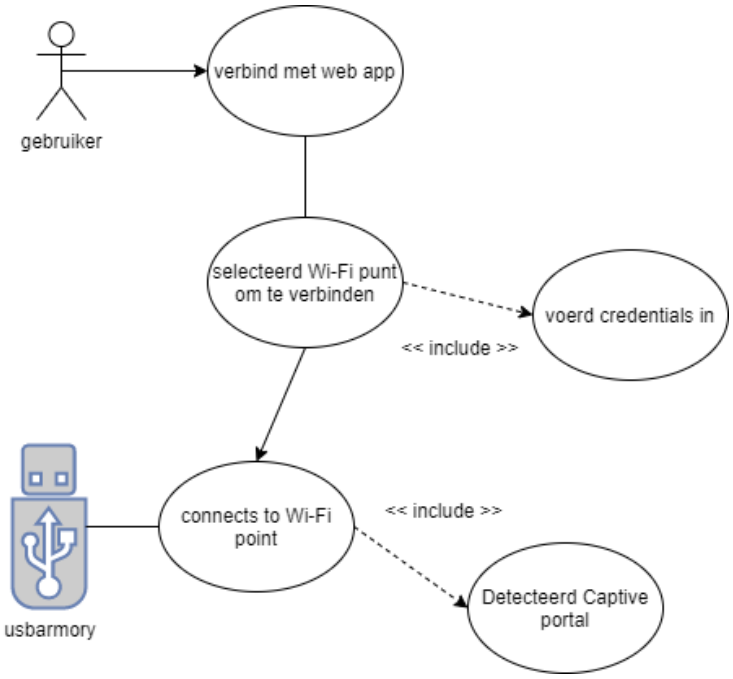
In deze spring zou deze eis gerealiseerd worden door op de GUI een Wi-Fi config pagina te maken. Hier kan de gebruiker een Wi-Fi punt selecteren om mee te verbinden. Tijdens de dagelijkse bespreking met de bedrijfsmentor kwam er een extra eis naar voren die niet opgenomen was in het oorspronkelijke plan van aanpak. Hierdoor is eis FE6.01 tot stand gekomen:

- FE06.1, De Wi-Fi Hotspot selectie moet ook succesvol plaats kunnen vinden als er spraken is van een zogeheten “captive portal”.
    - Een captive portal is een webpagina die weergegeven wordt met een webbrowser. Dit wordt getoond aan gebruikers voor zij volledige toegang tot het netwerk krijgen. (denk bijvoorbeeld aan de hotspot pagina als je verbind met de Wi-Fi in de trein of in de Mcdonalds.)

9.1 Ontwerp

Om aan deze eis te voldoen wordt er een pagina voor de web app gemaakt waar de gebruiker de Wi-Fi verbinding kan selecteren en hier de credentials kan invoeren. Bij de selectie van de Wi-Fi moet er rekening gehouden worden of de Wi-Fi Portal gebruik maakt van een captive portal. Als dit het geval is dan moet er iets uitgevoerd worden waardoor het captive portal omzeilt kon worden.

9.1.1 Gebruiksdiagram



Figuur 9-1 Gebruik diagram sprint 5

In figuur 9-1 is het Gebruiksdiagram te zien. De gebruiker verbindt met de web app, en navigeert naar de Wi-Fi selectie pagina. Hier selecteert de gebruiker het Wi-Fi punt om mee te verbinden. Hierna voert hij (indien nodig) zijn credentials in.

De Armory probeert vervolgens verbinding te maken het Wi-Fi punt. Daarna detecteert het of de Wi-Fi over een captive portal beschikt die de internetverbinding blokkeert.

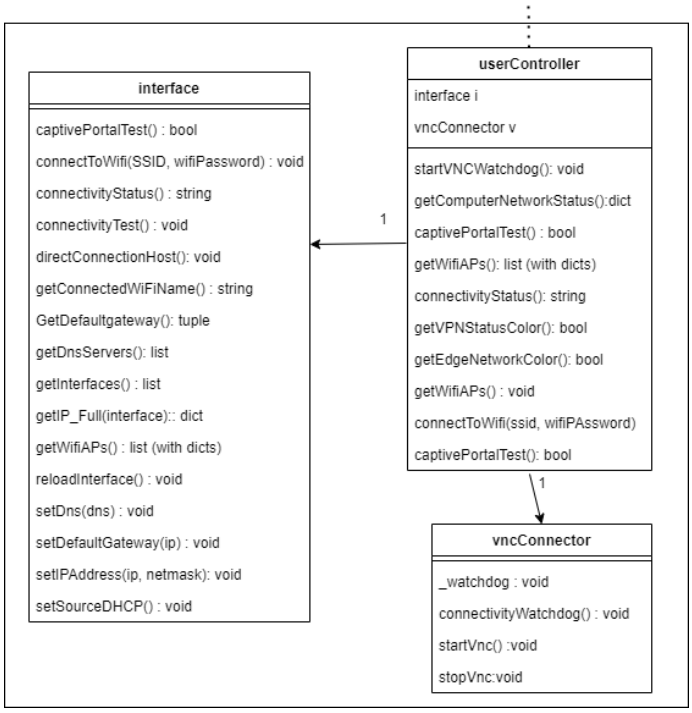
Als dit het geval is dan start de USB Armory de VNC-server. Hierover wordt later meer verteld in dit hoofdstuk

In tabel 9-2 de use-case diagram voor FE06.

Id	Use-case 6
Use-Case	Wi-Fi hotspot selectie/verbinding vindt plaats op de VPN-module, de gebruiker mag wel indirect deze selecteren.
Brief Description	Via web app kan de gebruiker een Wi-Fi verbinding selecteren
Primary Actors	Eindgebruiker
Secundary Actors	USB Armory
Pre-condition	- USB Armory is opgestart en in computer geplugd - USB armory heeft netwerkverbinding
Main flow	1. Eindgebruiker surft naar default gateway IP in de webbrowser 2. Eindgebruiker logt in op de Web app. 3. Eindgebruiker selecteert de Wi-Fi punt om mee te verbinden op Wi-Fi selectie pagina 4. USB Armory verbindt met de Wi-Fi punt. 5. De eindgebruiker heeft volledige netwerk toegang.
Alternative flow	2a. Wi-Fi Authenticatie faalt. 2b. Gebruiker krijgt pagina met foutmelding 4a. USB Armory detecteert Captive portal 4b. USB Armory start VNC-server op 4c. USB Armory presenteert pagina met melding/instructies afhandelen captive portal
Post conditions	USB Armory is verbonden met Wi-Fi.

Figuur 9-2 Use-case iteratie 5

9.1.2 Klassendiagram

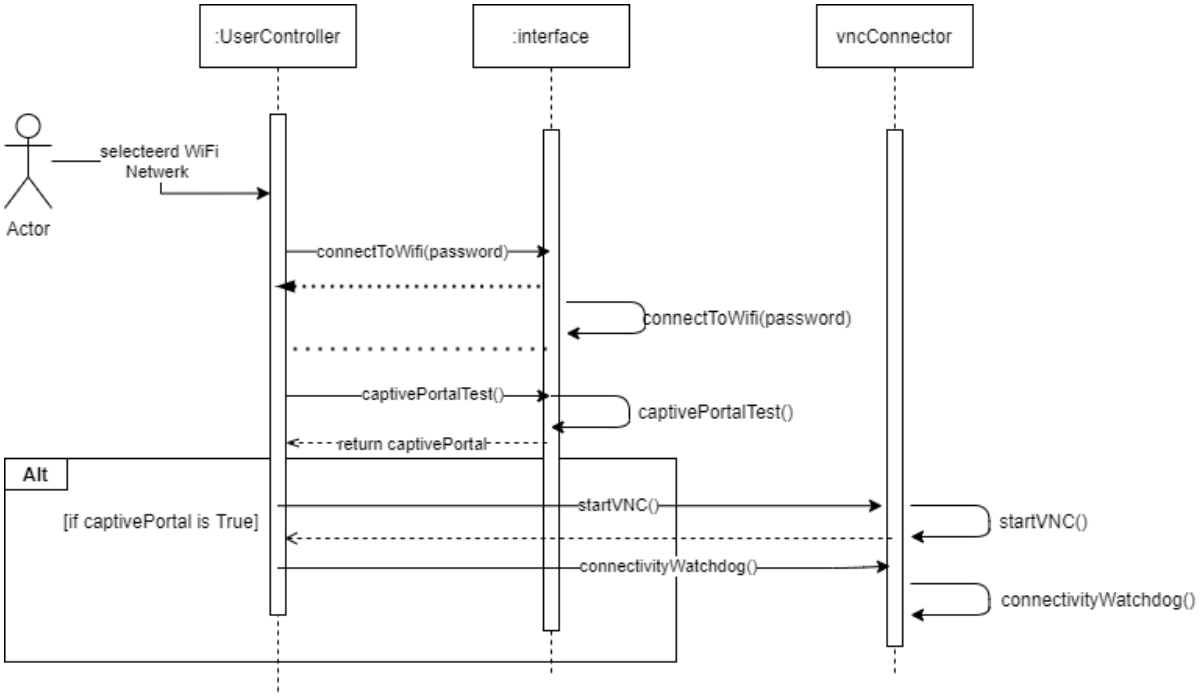


Figuur 9-3 klassendiagram iteratie 5

In figuur 9-3 is het klassendiagram van iteratie 5 te zien. Voor het aansturen van de VNC-server is de vncConnector klasse aangemaakt. Dit is een relatief simpele klasse die de VNC-server handmatig kan starten en stoppen, of een watchdog kan starten die de VNC-server automatisch stopt als er internetverbinding gedetecteerd wordt.

Omdat het aansturen van de vncController onder gebruiker acties valt wordt de vncConnector model aangestuurd vanuit de userController klasse.

9.1.3 Sequentie diagram



Figuur 9-4 sequentie diagram iteratie 5

In figuur 9-4 is het sequentie diagram voor iteratie 5 te zien. De gebruiker selecteert een Wi-Fi netwerk. Hierna verzorgt de UserController klasse dat de USB Armory met het netwerk verbindt. Vervolgens voert het met de functie captivePortalTest() een test uit of het Wi-Fi netwerk achter een captive portal zit. Als dit het geval is start hij de VNC-server, en de connectivity watchdog.

9.1.4 Keuze aansturing Wi-Fi netwerk

In deze iteratie werd er pagina gemaakt voor het selecteren en verbinden met de Wi-Fi. Onder Linux zijn er 3 manieren hoe je met een Wi-Fi netwerk kan verbinden.

- /etc/network/interfaces aanpassen;
- Via het commando **iwlist** de Wi-Fi netwerken detecteren, via **iwconfig** verbinden, en via **dhclient** een ip requesten;
- Via de Debian netwerkmanager tool. (**nmcli).**

Alle drie de opties werden getest. Het probleem van het aanpassen van de interface file was dat foutmeldingen uit een logfile gelezen moesten worden. Via de Linux commando's kon met vier commando's verbinding gemaakt worden met een Wi-Fi network. Met de netwerkmanager kan er door middel van éé commando al verbinding gemaakt worden met een netwerk.

Om deze reden is ervoor gekozen om het netwerk via de Debian netwerkmanager aan te sturen.

9.1.5 Keuze detecteren en doorlopen captive portal

Voor het detecteren en doorlopen van de captive portal zijn er naar meerdere mogelijkheden gekeken. In eerste instantie werd er naar een script gekeken dat de pagina automatisch doorliep door alle html buttons te selecteren. Echter was dit relatief onsuccesvol. Iedere captive portal is weer anders. Er kon geen stuk code gevonden worden wat dit 100% succesvol kon doorlopen. Mogelijk moest er een wachtwoord, form of enquête ingevuld worden. Hiervoor zouden handelingen van de gebruiker nodig zijn.

Om in aan de eis FE06 (configuratie moet op indirecte manier plaatsvinden) te voldoen werd gekeken hoe er op een veilige manier de captive portal aan de gebruiker gepresenteerd kon worden. Hier waren volgende 3 ideeën voor:

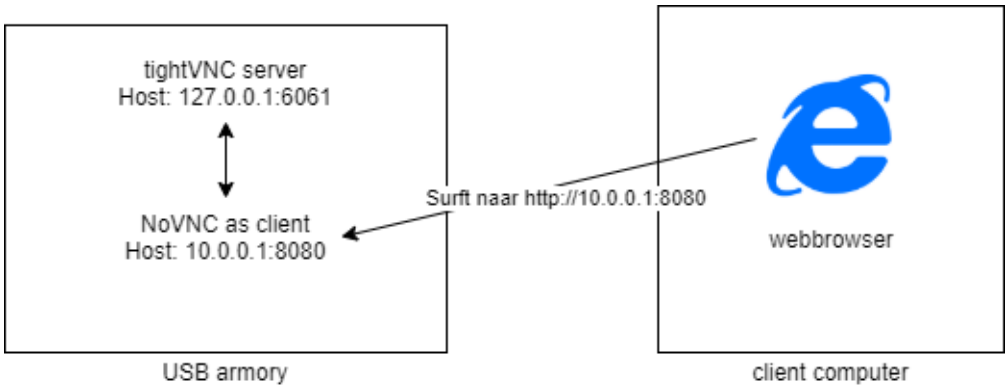
- De captive portal downloaden naar de USB Armory, en alle clientside code (javascript, ajax) verwijderen of filteren. Deze aanbieden via webserver van armory;
- RDP achtige oplossing waarbij de gebruiker via de webbrowser verbindt met de armory, en vervolgens op de armory de captive portal page invult;
- Een tijdelijke directe verbinding tussen client en netwerk, die gesloten wordt zodra er een internetverbinding gedetecteerd wordt.

De eerste optie was niet haalbaar. Veel captive portals maken gebruik van client side code zoals javascript om te functioneren. Hierdoor bleven optie 2 en 3 over. Voor beide werd een proof of concept gemaakt om te demonstreren aan de opdrachtgever.

Voor de RDP achtige oplossing werd er gebruik gemaakt van VNC. In figuur 9-5 een visualisatie hiervan. Op de USB Armory draait een VNC-server. NoVNC is een webservice die als VNC-client gebruikt wordt. De gebruiker hoefde alleen naar het IP van de service te surfen via zijn webbrowser om met de USB Armory te verbinden. Hierna werd automatisch een webbrowser in zijn webbrowser geopend met de captive portal landing page.

Voor de demo van de directe verbinding werd eerst gedetecteerd of er een captive portal was. Dit werd gedaan door een webpagina te downloaden (in dit geval http://captive.apple.com/). Als de

body niet de verwachte content bevatte (in dit geval ‘succes’) was er sprake van een captive portal. De client computer werd hierna direct verbonden met het netwerk. Zodra er een actieve internet connectie gedetecteerd werd zou de verbinding weer verbroken worden en worden vervangen door de VPN-verbinding.



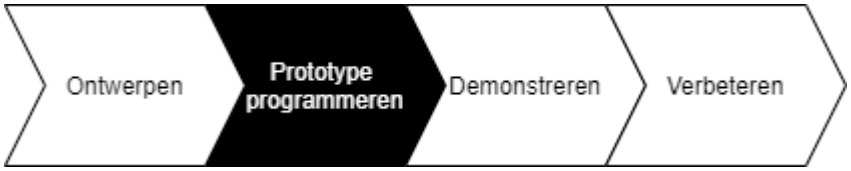
Figuur 9-5 werking poc remote desktop

Tijdens het maken van de demo kwam naar voren dat het inladen van webpagina’s vrij traag ging. De USB Armory heeft een single core processor van 900 MHz. In de demo moest het een grafische omgeving renderen, twee services (noVNC, en tightVNC) hosten, en een webbrowser engine draaien. Het duurde gemiddeld 10 seconden voor een webpagina was ingeladen.

Er is uiteindelijk voor gekozen om de VNC-oplossing te implementeren. Tijdens de demonstratie voor de opdrachtgever werd de directe verbinding demo gelijk afgekeurd. Het was niet wenselijk dat de host computer een directe verbinding met het netwerk zou maken.

Bij de VNC-demo vond de opdrachtgever het niet erg dat het 10 seconden duurde voor een pagina was ingeladen. Zijn redenering hiervoor was dat enkel de captive portal pagina geladen zou worden. Ook verwachtte hij dat dit proces geoptimaliseerd zou kunnen worden in de toekomst.

9.2 Programmeren prototype



Voor het programmeren van het prototype werd de vncConnector klasse gemaakt en de interface klasse aangepast zodat deze verbinding kon maken met een Wi-Fi netwerk. Ook werd een Wi-Fi selectie view voor de web app ontwikkeld. In dit hoofdstuk wordt het verloop hiervan besproken.

9.2.1 Views

In deze iteratie werd de Wi-Fi settings view gemaakt (zie bijlage G webpage views: Wi-Fi setting pagina.) Als de view de geladen wordt vraagt hij via userController alle Wi-Fi netwerken in de buurt op as een list. Zoals te zien in figuur 9-6 loopt hij door deze list heen en maakt een html table entry voor ieder netwerk met alle attributen erin. In de table entry wordt ter per netwerk een losse form aangemaakt die gebruikt kan worden om met dat specifieke netwerk te verbinden.

```
{% if WifiAPs %}
<table class="table table-striped">
   <thead>
      <tr>
         {% for name, value in WifiAPs[0].items() %}
         <th>{{ name }}</th>
         {% endfor %}
      </tr>
   </thead>
   <tbody>
      {% for ap in WifiAPs %}
      <tr>
         {% for name, value in ap.items() %}
         <td>{{ value }}</td>
         {% endfor %}
         <td><button onclick="myFunction('{{ ap["SSID"] }}')">connect</button></td>
      </tr>
      {% endfor %}
   </tbody>
</table>
```

Figuur 9-6 Wi-FiSettings.html loop

In figuur 9-7 is te zien dat als er op een connect button geklikt wordt er via javascript een pop up aangeroepen die de gebruiker vraagt om een WPA2 wachtwoord in te voeren; of het veld leeg te houden als er geen wachtwoord is. Zodra de gebruiker dit submit wordt de connectToWi-Fi() functie van de usercontroller aangeroepen in de Flask instance. De output hiervan (succesfully connected, of fail to connect) wordt hierna aan de gebruiker weergeven via de popup model.

```
<script>
   function myFunction(ap) {
     var pass = prompt("WPA2 password ? (leave empty if none needed)");

      document.getElementById("apToConnect").value = ap;
      document.getElementById("wpa2key").value = pass;
      document.getElementById("connectToApForm").submit();
   }
</script>
```

Figuur 9-7 popup voor Wi-Fi password

Een probleem met de view is dat voor hij ingeladen kan worden hij eerst alle Wi-Fi netwerken in de buurt moet scannen. Deze vraagt hij op bij de userController klasse, maar dit proces duurt ongeveer 15 seconden. Hierdoor lijkt het alsof de pagina niet reageert. Dit is momenteel opgelost door eerst een draaiend laad icoon te weergeven zodra de gebruiker klikt op de Wi-Fi setting knop in het menu.

9.2.2 Models

In deze iteratie werd de vncConnector view gemaakt en de interface klasse aangepast. In dit hoofdstuk wordt het verloop hiervan besproken.

Interface klasse

In de Interface klasse moesten twee zaken aangepast worden. Allereerst moet ervoor gezorgd worden dat deze met Wi-Fi netwerken kon verbinden. Voor het tweede deel moest er een functie aangemaakt worden die kon detecteren of de verbinding zich achter een captive portal bevond.

In voorgaande iteraties verbond de USB Armory met het internet door de /etc/network/interfaces file aangepassen en de netwerkverbinding te resetten. Foutmeldingen voor het verbinden met Wi-Fi netwerken konden hier lastig voor uitgelezen worden. Om dat te veranderen moesten alle **set** functies (setDns(0, setIP(), setDefaultGateway()) die voorheen de interface file aangepaste vervangen worden.

Voor het aansturen van de Debian networkManager waren er meerdere python packages beschikbaar. Echter tijdens het testen hiervan bleken ze of niet bijgewerkt te zijn, of een bug te bevatten die ze onbruikbaar maakte. Uiteindelijk werd gekozen om gebruik te maken van de package **pynmcli.** Deze python package is een zogeheten **wrapper package**. Dit houdt in dat de package iedere individueel networkManager commando in een functie call wikkelt (zie figuur 9-5). Hierdoor is de errorhandeling vrijwel niet bestaand, maar hierdoor was het niet nodig zelf een nieuwe package te maken.

```
def connectivityStatus(self):
    return pynmcli.NetworkManager.Networking().connectivity().execute()
```

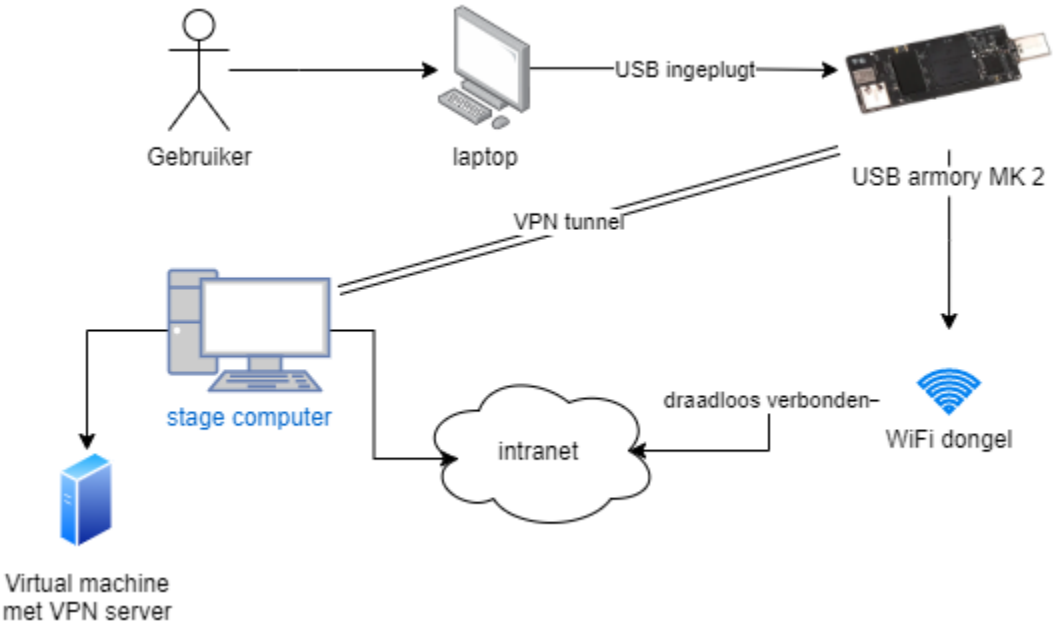
Figuur 9-8 implementatie connectivity test

Voor het detecteren van de captive portal werd de huidige netwerk status via de networkManager uitgelezen (zie figuur 9-9). De output string hiervan werd vergeleken, en aan de hand daarvan werd een boolean gereturnd.

```
def captivePortalTest(self):
    if(self.connectivityStatus() == 'portal\n'):
        return True
    else:
        return False
```

Figuur 9-9 implementatie captivePortalTest()

9.3 Afname test

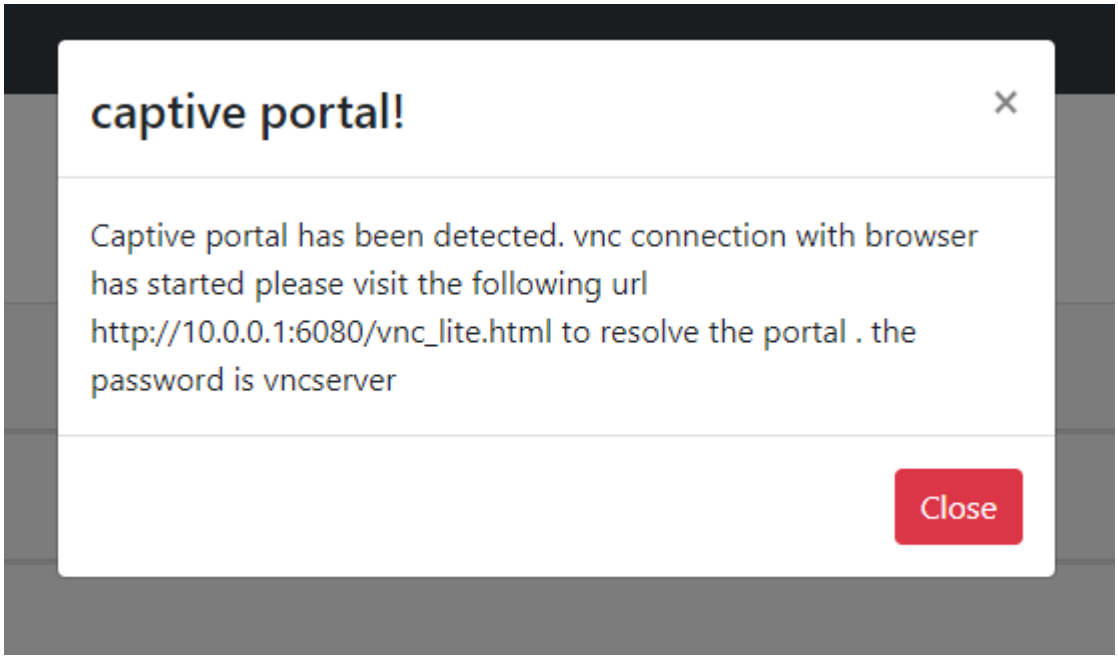


Figuur 9-10 testopstelling voor iteratie 5

Bij deze afname test werd de opstelling in figuur 9-10 gebruikt. Op de testlaptop was de USB armory ingeplugd. Via de Wi-Fi dongel kon de USB armory draadloos met het intranet verbinden. Op de stage computer draaide een virtual machine met een OpenVPN server geconfigureerd voor de USB armory om mee te verbinden. De stagecomputer was via een kabel verbonden met het intranet.

Voor de test logde de opdrachtgever via de test pc in op het login portal van de USB Armory. Hierna navigeerde hij naar de Wi-Fi configuratie pagina. Hier kon hij de Wi-Fi van Technolution selecteren en hier vervolgens mee verbinden. Het duurde iets langer dan normaal voor de Wi-Fi configuratie pagina ingeladen werd. Dit had te maken met de grote hoeveelheid accespoints in het gebouw die de USB armory moest scannen.

Om de captive portal te demonstreren moest de software helaas aangepast worden om dit te triggeren. Het was niet gelukt om een captive portal op te zetten, hiervoor was geen geschikt opensource of gratis pakket voor; enkel betaalde. Om dit toch te demonstreren werd de code aangepast dat de USB Armory altijd een captive portal detecteerde. De opdrachtgever logde in waarna een melding weergeven (figuur 9-11) werd dat er een captive portal gedetecteerd werd. In de melding kon de opdrachtgever op de link drukken, waarna hij naar de noVNC webpage werd geleid. Hier kon hij webbrowsen via no noVNC pagina, in zijn browser.



Figuur 9-11 melding captive portal

In deze iteratie is voldaan aan de eisen FE 6, en FE6.1. In tabel 9-12 en 9-13 een samenvatting van de afname testen. De computer kon succesvol met de Wi-Fi verbinden, en een captive portal omzeilen. Eis FE6.1 kon niet helemaal goed getest worden omdat er geen captive portal software beschikbaar was. Echter kon met een aanpassing het proces wel gedemonstreerd worden. Beide demonstraties demonstreerde dat de eisen behaald waren

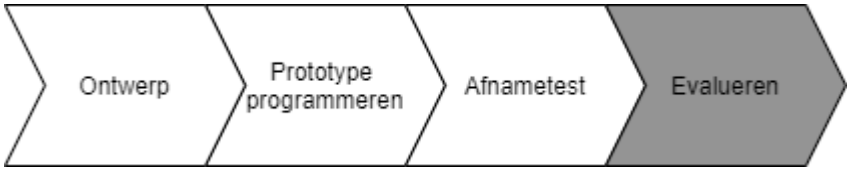
Test	Use-case 6
Use-Case	Wi-Fi hotspot selectie/verbinding vindt plaats op de VPN-module, de gebruiker mag wel indirect deze selecteren.
Brief Description	Via web app kan de gebruiker een Wi-Fi verbinding selecteren
Pre-condition	- USB Armory heeft een internetverbinding - USB Armory is in test pc geplugd - USB Armory is opgestart
Test process	1. Eindgebruiker surft naar default gateway IP in webbrowser 2. Eindgebruiker logt in op Web app. 3. Eindgebruiker gaat naar de Wi-Fi configuratie pagina 4. Eindgebruiker selecteert Wi-Fi om mee te verbinden 5. Eindgebruiker voert wachtwoord in 6. Eindgebruiker verifieert dat de Armory verbonden met Wi-Fi op de Status pagina.
Verwachte uitkomst	USB armory is verbonden met Wi-Fi netwerk.
Uitkomst	Uitkomst was zoals verwacht

Figuur 9-12 afnametest use-case

Test	Use-case 6.1
Use-Case	De Wi-Fi Hotspot selectie moet ook succesvol plaats kunnen vinden als er spraken is van een zogeheten “captive portal”.
Brief Description	Computer kan “captive portal” verhelpen
Pre-condition	- USB Armory is in test pc geplugd - USB Armory is opgestart - USB Armory is verbonden met netwerk met captive portal
Test process	1. Eindgebruiker surft naar default gateway IP in webbrowser 2. Eindgebruiker logt in op Web app. 3. Eindgebruiker gaat naar de Wi-Fi configuratie pagina 4. Eindgebruiker selecteert Wi-Fi om mee te verbinden 5. USB Armory detecteert captive portal 6. Usb armory start noVNC, vncserver, en webbrowser 7. USB Armory geeft melding met link naar noVNC 8. Gebruiker verbindt met NoVNC 9. Gebruiker vult captive portal in 10. USB Armory stop vncserver zodra er volledige netwerk connectie is
Verwachte uitkomst	Captive portal verholpen
Uitkomst	Kon niet getest worden omdat captive portal software niet beschikbaar was.

Figuur 9-13 Afnametest use-case 6.1

9.4 Evalueren afnametest



De opdrachtgever was tevreden over de resultaten die bereikt waren tot nu toe met het project. De hoofdeisen voor het project waren bereikt. Voor de volgende sprint zou het prototype verbonden worden met een Technolution Primelink. Echter mijn bedrijfsbegeleider en opdrachtgever waren het er allebei over eens dat dat gewoon zou werken, en hier geen iteratie aan besteed te hoeven worden. Ook zou dit enkel testen zijn, en zou er geen software voor ontwikkeld hoeven worden.

10 Oplevering en aanbevelingen

Voor de afronding is een overdracht document geschreven voor Technolution. Hierin is omschreven hoe de resultaten die in dit project bereikt zijn en gereproduceerd kunnen worden. Het volledige document is te vinden in bijlage c: **applicatie document**

10.1 Opgeleverde code

Alle geschreven python functies zijn voorzien van een omschrijving wat ze doen. Als ze een input hadden hebben ze ook een input type en output type in de omschrijving staan (zie figuur 10.1).

```
def hashString(self, string):
    """
    input: string to encrypt
    description: turns the input string in a encrypted hash
    output: string with hashed string.
    """
```

Figuur 10-1 voorbeeld docstring

De html view bestanden hebben hun omschrijving met een jinja2 comment veld omschreven in plaats van een html comment veld. Dit zorgt ervoor dat als de webpagina gerendered wordt het commentaar eruit verwijderd wordt en niet te zien is in de broncode.

De code is goedgekeurd door de bedrijfsmentor en geüpload naar de Gitlab van Technolution.

10.2 Conclusie

In dit project was het de bedoeling om een VPN-module te maken die een beveiligde verbinding kon opzetten voor de host pc. Dit zonder dat de host pc ooit hoefde te verbinden met “onbeveiligd internet”. Voor dit project was de USB Armory als platform gekozen.

Hoewel het concept van een hardware VPN-module een goed concept is zijn er tijdens het ontwikkelen van de module verschillende problemen naar voren gekomen waardoor de USB Armory geen geschikt platform blijkt voor deze doeleinden. Voor een VPN-module is het belangrijk dat hij op een makkelijke manier een externe verbinding met het internet kan maken. Eventueel via een ethernet kabel of via een Wi-Fi verbinding. De USB Armory heeft standaard geen ingebouwde Wi-Fi chip. Dit moest opgelost worden met een externe Wi-Fi dongel, wat ervoor zorgde dat de USB Armory groot en onhandig was in gebruik. Ook heeft de USB Armory enkel usb C poorten. Voor de meeste computers is USB B nog steeds de standaard. Dit zorgde ervoor dat er gebruik gemaakt moest worden van een USB B naar USB C converter om de USB Armory te gebruiken, wat hem nog groter en onhandiger in gebruik maakte.

Het tweede argument waarom de USB Armory geen geschikt platform is om een VPN-module mee te maken is dat hij hier niet voor ontwikkeld is. De USB Armory is ontwikkeld voor security based applicaties. Het heeft een externe crypto grafische co-processor, ARM trustzone, usb device emulation, en vele andere securityfeatures die geen direct nut voor het proof of concept hebben. Ze kunnen gebruikt worden om de beveiliging van het eindproduct te verbeteren, maar vaak zijn er softwarematige oplossingen die dit beter kunnen implementeren.

Het laatste argument is dat de prijs door de extra securityfeatures een stuk duurder dan nodig is. De USB Armory kost 170 euro, en komt met een 900 MHz Cortex CPU, en 512 mb aan ram. Voor dezelfde prijs is een W5 pro mini stick te koop (https://www.amazon.nl/W5-Stick-Windows-ondersteuning-Bluetooth/dp/B07MB9TC33) met vier keer zoveel computerkracht. Op deze computer kunnen zaken als disk encryptie ook gewoon geïmplementeerd worden.

Als er een gelijk soort product ontwikkeld wordt moet er ook goed over nagedacht worden hoe de OpenVPN processen aangestuurd worden. Standaard heeft OpenVPN voor de client geen manier om de configuratie live aan te passen. De OpenVPN service moet herstart worden om de wijzigingen door te voeren. Dus als de module bijvoorbeeld meerdere vpn tunnels heeft, en 1 hiervan wordt gewijzigd moeten alle VPN tunnels opnieuw starten.

10.3 Advies voor doorontwikkeling

Als het proof of concept door ontwikkeld wordt zijn er nog een aantal bugs en verbeteringen die doorgevoerd zouden kunnen worden:

- De applicatie draait momenteel onder root, om de service veiliger te maken zou er een speciale vpn module groep moeten komen die de benodigde commando's kan draaien.
- De webGUI is momenteel niet alleen via de host pc bereikbaar, maar ook via de Wi-Fi connectie. Dit zou opgelost kunnen worden door apache alleen te laten luistern naar eth0
- Een bug die momenteel erin zit is dat de VNC-server crasht als een connectie van full internet teruggaat naar een captive portal
- De applicatie is ontwikkeld met Flask, de built in Flask server is niet geschikt voor productie. De app zou op een WSGI-container gehost moeten worden

- De VNC-solution voor de captive portal is een slecht uitgewerkt idee. Hier zou iets op maat gemaakt voor moeten worden geprogrammeerd. Voor demonstratiedoeleinden werkt het voldoende.

11 Evaluatie

11.1 Proces

In dit hoofdstuk wordt er geëvalueerd op de gemaakte keuzes er werkwijze tijdens het afstudeerproject. Er wordt beoordeeld wat er goed ging en fout is gegaan in het traject; en hoe in de toekomst het project anders uitgevoerd had kunnen worden.

Bij het uitvoeren van het project werd er een aangepaste versie van de RAD project methodiek gebruikt. Bij het onderzoeken voor een geschikte project methodiek is er naar de verschillende voor- en nadelen van de meest gebruikte project methodieken gekeken. Van al deze methodieken leek RAD voor het grootste deel te voldoen aan de eisen van het project. De ontwikkel fase voldeed niet geheel aan de verwachtingen. Om deze reden werd ervoor gekozen om deze aan te passen met non-RAD 'subfases'. Hoewel de methodiek hierna meer voldeed aan het beeld hoe er te werk gegaan moest worden leidde dit op bepaalde momenten tot verwarring. Op deze momenten van de subfases (hoofdzakelijk het testen van de iteratie) vergat ik soms delen van de bedachte structuur, en moest bijvoorbeeld het ontwerp een tweede keer gemaakt worden. In de toekomst zou ik meer proberen te werken volgens de hoofdlijnen van de gebruikte methodiek.

Bij de analyse van het project is er stapsgewijs te werk gegaan om de gaten in de kennis te vullen voordat er beslissingen en keuzes gemaakt moesten worden. In dit project was er geen voorkennis wat de mogelijkheden van het platform waren, en hoe deze toegepast kon worden. Om deze reden werd eraan het begin van iedere fase en iteratie de tijd genomen om relevante kennis op te doen. Hierdoor kon ik als ik een bespreking had met de opdrachtgever goede suggesties maken of een idee wel of niet haalbaar was. Hierdoor ging het opstellen van de requirements relatief vlot en is er weinig opvallends gebeurd tijdens de analyse.

Het ontwerpen van de software ging goed. Er werd structureel gewerkt met UML. In iedere iteratie werden de eisen omgezet naar use-cases, waarna het klassendiagram werd gemaakt. Daarna is aan de hand van de use-cases en het klassendiagram een sequentiediagram opgesteld. Voor de eerste iteraties nam ik de tijd om een rapport met deze diagrammen en grafieken op te leveren, maar de opdrachtgever gaf al snel aan dat hij hier niet geïnteresseerd in was. Hij vond het resultaat vooral belangrijk. Hierna maakte ik de diagrammen hoofdzakelijk om voor mijzelf om het overzicht te bewaren. Wat ik wel merkte was dat ik weinig ervaring had in het maken van ontwerpen. Veel ontwerpen moest ik schrappen en opnieuw maken. Ik zou hierdoor dan ook als ik het opnieuw had moeten doen meer hulp van mijn bedrijfsmentor hebben gevraagd.

Het ontwikkelen en testen van de software ging goed. Aan het eind van iedere iteratie was er een contact moment met de opdrachtgever waarbij het prototype gedemonstreerd werd. Aan de hand van de feedback die hij gaf kon het project in de volgende iteratie waar nodig bijgestuurd worden. Een van de punten die terugkwam was dat ik de tijd spendeerde aan features die de opdrachtgever niet relevant vond. Een voorbeeld hiervan was dat ik uitgebreide form validatie had toegepast in de web app zodat enkel valide gegevens ingevoerd konden worden. De opdrachtgever herinnerde mij eraan dat ik een prototype aan het ontwikkelen was. Deze tijd had ik beter kunnen spenderen om nieuwe features te maken. Ook liep ik er soms tegenaan dat ik te veel bij één softwareoplossing bleef hangen en niet naar de alternatieven keek. Mijn bedrijfsmentor suggereerde in deze gevallen vaak een alternatief dat veel beter was dan de softwareoplossing waar ik aan dacht. In de toekomst zou ik dan ook ideeën altijd eerst met een collega bespreken voor ik deze zou uitvoeren.

In zijn geheel was ik tevreden met het project. In de toekomst zou ik een soortgelijk project op dezelfde manier uitvoeren met kleine aanpassingen. Op deze manier zouden keuzes beter gemaakt kunnen worden.

11.2 Beroepstaken

In dit hoofdstuk wordt genoemd waarom er is voldaan aan de beroepstaken die zijn opgesteld in het afstudeerplan.

A1 Analyseren probleemdomein & opstellen probleemstelling
Bij het analyseren van het probleem domein heb ik de visie en de eisen achterhaald die nodig waren om de afstudeeropdracht te voltooien. Hierbij is gekeken naar de visie van de opdrachtgever, de mogelijkheden om deze visie te realiseren, en of dit ook realistisch is. In hoofdstuk 3.2 is de probleemstelling opgesteld, en in hoofdstuk 3.3 de doelstelling hiervoor.

A2 Informatie vergaren, analyseren, beoordelen & verwerken
In dit project hebt ik in iedere fase informatie vergaard, en geanalyseerd. In de initiatiefase heb ik een informatie vergaard in de vorm van het visie gesprek met de opdrachtgever. Dit is terug te lezen in hoofdstuk 4.1. In de oriëntatiefase heb ik de informatie die ik uit de initiatiefase hebt verkregen geanalyseerd, en beoordeeld dat de visie haalbaar was. Dit valt terug te lezen in hoofdstuk 5.2.

C6 Ontwerpen Software
In de ontwikkelingsfase heb ik in iedere iteratie als eerste stap een ontwerp gemaakt. Eerst heb ik de eis omgezet naar een use-case. Aan de hand van de use-case maakte ik een klassendiagram hoe het programma eruit zou zien. Vervolgens maakteik een sequentie diagram hoe het programma in sequentie functioneerde. Dit valt terug te lezen in hoofdstuk 6.2, 7.1, 8.1 en 9.1

D1 Realiseren van software
Tijdens het project is de ontworpen software gerealiseerd. Hierbij is gebruik gemaakt van frameworks en bestaande libraries. Deze frameworks en libraries zijn gecombineerd met mijn eigen methodes om de gewenste functionaliteit te behalen. In hoofdstuk 6.3, 7.2, 8.2, en 9.2 wordt er besproken hoe de geschreven code tot stand is gekomen. In hoofdstuk 6.1.5 omschrijf ik waarom er python gebruikt is in dit project.

D2 testen & evalueren.
Aan het einde van iedere iteratie in de ontwikkeling fase is er voor iedere eis een afname test geschreven en uitgevoerd. Deze tests zijn omgezet naar een testrapport. Aan het einde van de test is er geëvalueerd hoe de iteratie verlopen was. Een voorbeeld van de test is terug te lezen in hoofdstuk 6.4. Een uitwerking van een evaluatie is terug te lezen in hoofdstuk 6.5

Gc Kritisch, onderzoekend en methodisch werken
Tijdens het afstudeerproject is er gebruik gemaakt van een aangepaste variant van de RAD methodiek. Hierbij is de conceptfase die bij RAD hoort methodisch gevolgd. Iedere fase bouwde voort op de fase die ervoor kwam. Dit valt in het volledige verslag terug te lezen

Ge creatief en innovatief werken
Binnen Technolution was er de vraag of een bestaande technologie (OpenVPN-NL) in een nieuw platform (USB Armory) verwerkt kon worden. Hierbij zijn er verschillende technologieën, frameworks, en libraries gebruikt om op (soms op extreem creatieve wijze) een nieuw product te ontwikkelen. In iedere iteratie kwamen er eisen naar voren die op creatieve wijze omgezet moesten worden in software.

Literatuurlijst

admin. (sd). *How to configure and Manage Network Connections using nmcli*. Opgehaald van thegeekdairy: https://www.thegeekdiary.com/how-to-configure-and-manage-network-connections-using-nmcli/

arlow, j. (sd). *Uml and the Unified Process and Uml: Practical Object-Oriented Analysis and Design.* booch jacobson rumbaugh.

Barisani, A. (sd). *USB armory wiki*. Opgehaald van github: https://github.com/f-secure-foundry/usbarmory/wiki

BV, T. (sd). *Bedrijfsorganogram Technolution BV.* Opgehaald van intranet Technolution.

Desktop Environment : noVNC Client. (sd). Opgehaald van server world: https://www.server-world.info/en/note?os=Ubuntu_18.04&p=desktop&f=7

fossil. (sd). *Accurately detecting a captive portal in python*. Opgehaald van stackoverflow: https://stackoverflow.com/questions/42404750/accurately-detecting-a-captive-portal-in-python

Frank. (sd). *Flask web forms*. Opgehaald van pythonspot: https://pythonspot.com/Flask-web-forms/

Gite, V. (sd). *How to compile and install Linux Kernel 5.6.9 from source code*. Opgehaald van cyberciti: https://www.cyberciti.biz/tips/compiling-Linux-kernel-26.html

Grinberg, M. (sd). *The Flask Mega-Tutorial*. Opgehaald van miguelgrinberg: https://blog.miguelgrinberg.com/post/the-Flask-mega-tutorial-part-iii-web-forms

Hasanov, R. (sd). *Flask Form Validation with Flask-WTF*. Opgehaald van stackabuse: https://stackabuse.com/Flask-form-validation-with-Flask-wtf/

How to guide: setting up & configuring OpenVPN client/server. (sd). Opgehaald van OpenVPN: https://OpenVPN.net/community-resources/how-to/

İsmail Baydan. (sd). *How To Install and Access TightVNC Remote Desktop In Linux?* Opgehaald van poftut: https://www.poftut.com/how-to-install-and-access-tightvnc-remote-desktop-in-Linux/

Jim Anderson. (sd). *An Intro to Threading in Python*. Opgehaald van realpython: https://realpython.com/intro-to-python-threading/

Mocan, T. (sd). *What Is OpenVPN & How Does OpenVPN Work?* Opgehaald van cactusvpn: https://www.cactusvpn.com/beginners-guide-to-vpn/what-is-OpenVPN/

Network Portal Detection. (sd). Opgehaald van The Chromium Projects: https://www.chromium.org/chromium-os/chromiumos-design-docs/network-portal-detection

Projectmanagement-training.nl. (sd). *Wat is Agile?* Opgehaald van projectmanagement-training: https://www.projectmanagement-training.nl/wat-is-agile/

Team, L. C. (sd). *4 Phases of Rapid Application Development Methodology*. Opgehaald van lucidchart.com: https://www.lucidchart.com/blog/rapid-application-development-methodology

Afkortingenlijst

Afkorting	Betekenis
VPN	Virtual Private Network
Harden	Extra beveiligen
Secure environment	Beveiligde omgeving
Package manager	De windows app store voor Linux
Dependenties	Libraries waar een programma afhankelijk van is
Distro	Een Linuxdistributie of ook wel distro is een besturingssysteem bestaande uit de Linuxkernel en andere software die op elkaar is afgestemd.
GUI	Graphical user interface

Bijlage

Bijlage	Titel
A	Afstudeerplan
B	Analyse document
C	Applicatie documentatie
D	Plan van aanpak
E	Volledige klassendiagram
G	Webpage views

Bijlage A: afstudeerplan

Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie bij Technolution BV. Gouda

Bart de Langen
15077071
HBO-ICT Network & System Engineering
De Haagse Hogeschool Delft
Versie 0.3
26 maar 2021

Afstudeerblok	2020-2.1
Startdatum uitvoering afstudeeropdracht	**15 oktober 2020**
Inleverdatum afstudeerdossier volgens jaarrooster	

Studentnummer	15077071
Achternaam	De Langen
Voorletters	BJS
Roepnaam	Bart
Adres	Brasserskade 63
Postcode	2612ca
Woonplaats	Delft
Telefoonnummer	
Mobiel nummer	0630097523
Privé emailadres	b.delangen@live.nl

Differentiatie	NSE
Afstudeerprogramma	IoT
Domein*	IoT
Locatie	Delft
Variant	Voltijd

**geef hier aan binnen welk domein jouw opdracht valt, m.a.w. wat voor type opdracht het is*

Naam studieloopbaanbegeleider	
Naam begeleidend examinator	
Naam expert examinator	

Bedrijf

Naam	Technolution B.V.
Afdeling	
Bezoekadres	Burg. Jamessingel 1

Postcode	2803 WV
Plaats	Gouda
Telefoonnummer	0182-594000
URL	www.technolution.eu

Opdrachtgever

Achternaam	Hofman
Voorletters / Voornaam	Jonathan
Titel / Opleiding	Ir./MSc - TU Delft Electrotechniek Bachelor/Computer engineering Master
Functie	Domein Architect
Telefoonnummer (werk)	0182-594000
Emailadres (werk)	jonathan.hofman@technolution.nl
is de opdrachtgever ook de bedrijfsmentor?	NEE[1]

Bedrijfsmentor

Achternaam	Rietveld
Voorletters / Voornaam	Roy
Titel / Opleiding	HBO elektrotechniek richting TCK
Functie	Architect
Telefoonnummer (werk)	0182-594000
Emailadres (werk)	roy.rietveld@technolution.nl

[1] Indien JA dan 'Bedrijfsmentor' leegmaken en niet invullen

Titel afstudeeropdracht

Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie voor werkplekken met een hoog dreigingsprofiel.

Opdrachtomschrijving

1. Bedrijf

Technolution is een technologie integrator. Ze focussen zich op het ontwikkelen van gespecialiseerde software en product oplossingen voor bedrijven en overheden. Hierbij ontwikkelen zij zelf de elektronica, programeerbare logica, embedded software en technische informatie systemen. Het bedrijf is in bezit van de oprichters en de werknemers. Hun kantoor is te vinden in Gouda. Het bedrijf bestaat uit ongeveer 230 medewerkers.

2. Probleemdomein

We zien steeds meer behoefte om de data veilig tot in het end-point device te brengen. Ook wel zero-trust networking genoemd. Daarnaast groeien de dreigingen. Zo geven de recente kwetsbaarheden in de hardware van PC platformen voldoende aanleiding om niet meer volledige op de software-scheiding van operating systemen te kunnen vertrouwen. Traditionele software VPN oplossingen bieden niet voldoende weerstand tegen hoge dreigingen. Dit geeft voldoende aanleiding om hardware geïsoleerde VPN-oplossingen te creëren.

3. Doelstelling van de afstudeeropdracht

De opdracht is het ontwikkelen van een demonstratieproduct waarbij er een scheiding is tussen het pc-platform en het internet is aangebracht. Hierbij wordt er gebruik gemaakt van een security USB-computer die recent op de markt is gekomen, de USB armory mk 2 (hierna genoemd de USB armory). De USB armory heeft ethernet emulation mogelijkheden. De eindgebruiker zou alleen de USB armory in hoeven te pluggen Hierna zal de USB armory eventuele authenticatie afhandelen, en zou USB armory een secure tunnel opzetten.

Bij het ontwikkelen van het product zijn er verschillende uitdagingen. Zo moet de authenticatie en selectie van WIFI accesspoints plaatsvinden zonder dat de computer verbinding heeft met het onvertrouwde deel van het netwerk, er moet een goede beveiligde opslag zijn van de gebruikerscertificaten, en is het gewenst dat het prototype met de bestaande VPN-producten van Technolution kan verbinden.

4. Resultaat

Als eindresultaat zou er een demonstratie moeten komen waarbij de USB armory ervoor kan zorgen dat een PC een secure VPN verbinding kan opzetten zonder dat er software voor geïnstalleerd hoeft te worden. Ook wil Technolution bekend worden met technieken die beschikbaar zijn om op de USB armory gebruikt kunnen worden om de beveiliging van hun eigen producten te verhogen.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Globale fases:

- Oriëntatie van de opdracht -2 weken
- Literatuuronderzoek VPN mogelijkheden onder Linux based environments – 1 week
- Onderzoek naar functionele specificaties van de USB Armory mk 2 – 1 week
- Proof of concept ontwikkelen voor eerste demo – 4 weken
- Ontwikkelen van een prototype. - 3,5 weken
- Testen Prototype – 1 week
- Product productierijp maken - 3,5 weken
- Afronden documentatie voor doorontwikkeling van product. - 1 week

Activiteiten

- Experts Technolution raadplegen ontwikkelen op embedded platform – 5 dagen
- Vergaren van Eisen – 2 dagen
- Overleg bedrijfsmentor ontwikkelingen binnen project – 7 dagen
- Schrijven van plan van aanpak - 2 dagen
- Literatuur onderzoek VPN solutions in linux based omgevingen - 3 dagen
- Onderzoek naar mogelijkheden USB armory mk 2 - 1 dag
- Opzetten distribution USB armory mk 2 – 3 dagen.
- Proof of concept ontwikkelen + uittesten mogelijkheden USB Armory – 10 dagen
- Demonstratie proof of concept. – 1 dag
- Aanvullend onderzoek en ontwerpen prototype -5 dagen
- Bouwen prototype – 15 dagen
- Documentatie schrijven prototype 3 dagen
- Demonstratie/testen prototype – 1 dag.
- Aanpassen ontwerp aan de hand van feedback – 5 dagen
- Productierijp maken/bugfixing -15 dagen
- Documentatie voor doorontwikkeling afronden. – 3 dagen
- Demo productierijp product – 1 dag

- Tussen deze activiteiten door zal ik aan mijn stageverslag werken

6. op te leveren (tussen)producten

- Plan van eisen
- Plan van aanpak
- Proof of concept USB armory
- Prototype end-to-end encryption USB Armory MK 2
- Documentatie end-to-end encryption solution.
- USB Armory 2 custom Linux image en product code.

7. Te demonstreren competenties en wijze waarop

- **A1 Analyseren probleemdomein & opstellen probleemstelling**

Tijdens de afstudeerstage zal ik het probleemdomein van mijn opdracht verder definiëren en een probleemstelling schrijven

- **A2 Informatie vergaren, analyseren, beoordelen & verwerken**

Onderzoek naar de functionaliteiten van de USB armor mk 2, VPN solutions in linux, en het ontwikkelen in een embedded omgeving.

- **C6 Ontwerpen Software**

Voor de end-to-end encryptie solution zal er programmatuur gemaakt moeten worden. Het ontwerpen van deze programmatuur zorgen ervoor dat ze gestructureerd opgebouwd worden

- **D1 Realiseren van software**

Het doel van de opdracht is het opleveren van een product. Hiervoor schrijf ik de programmatuur

- **D2 testen & evalueren.**

Gedurende deze opdracht zullen er een aantal proof of concepts opgeleverd worden. Deze zullen getest worden op fouten waarna deze in de volgende fase opgelost worden.

- **Gc Kritisch, onderzoekend en methodisch werken**

Gedurende deze zal ik werken via een methodische aanpak. Deze zal worden omschreven worden in het plan van aanpak. Ook zal ik aan het einde van iedere fase kritisch reflecteren naar hoe het gegaan is, en de feedback hiervan meenemen naar de volgende fase.

- **Ge creatief en innovatief werken**

Ik ontwikkel het product vanaf de grond af aan. Een van de doelen is om de mogelijkheden van het ontwikkelingsplatform te demonstreren. Hierdoor zal ik creatief moeten omgaan in de ontwikkeling van de software om dit goed naar voren te laten komen.

Bijlage B: Analyse document

Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie bij Technolution BV. Gouda

Bart de Langen
15077071
HBO-ICT Network & System Engineering
De Haagse Hogeschool Delft
Versie 0.3
26 maar 2021

12 Inhoudsopgave

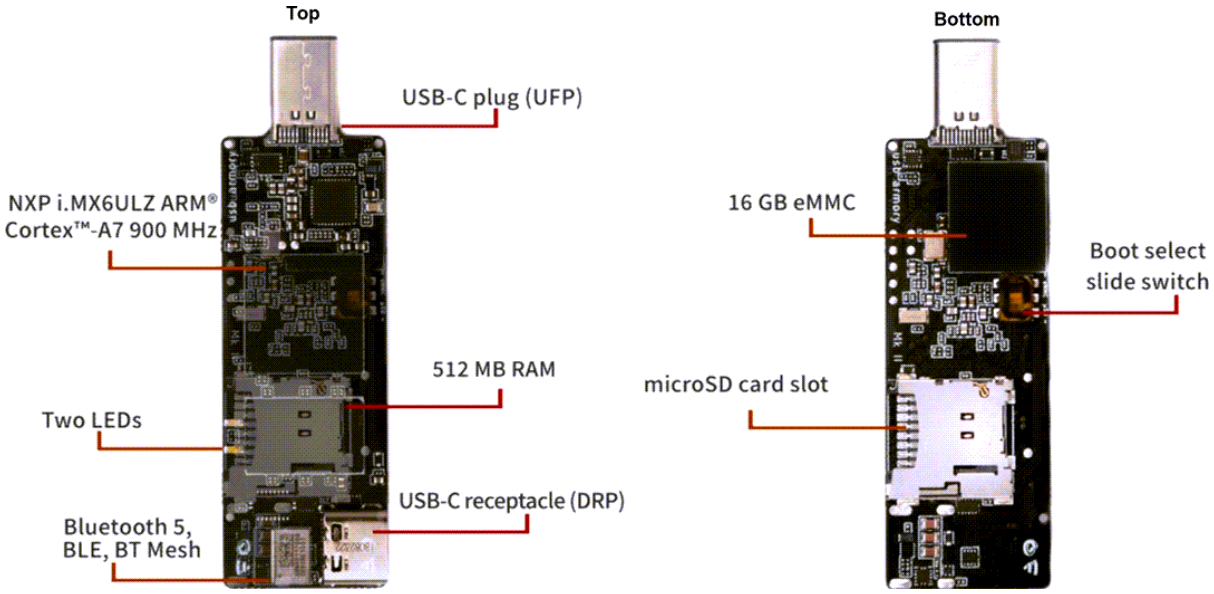
1. Inleiding

We zien steeds meer behoefte om de data veilig tot in het end-point device te brengen. Ook wel zero-trust networking genoemd. Daarnaast groeien de dreigingen. Zo geven de recente kwetsbaarheden in de hardware van PC platformen voldoende aanleiding om niet meer volledig op de software-scheiding van operating systemen te kunnen vertrouwen. Traditionele software VPN-oplossingen bieden niet voldoende weerstand tegen hoge dreigingen. Dit geeft voldoende aanleiding om hardware geïsoleerde VPN-oplossingen te creëren.

De afstudeer opdracht is het ontwikkelen van deze oplossing. Hierbij moet er demonstratieproduct ontwikkeld worden waarbij er een scheiding is aangebracht tussen het pc-platform en het internet. Dit wordt ontwikkeld op een USB Armory mk ii. De eindgebruiker zou alleen de USB Armory hoevn in te pluggen. Hierna zal de USB Armory eventuele authenticatie afhandelen, en een secure VPN tunnel opzetten.

2. USB Armory MK 2

In dit project wordt het product ontwikkeld op een USB Armory MK 2.



Afbeelding 1. USB Armory mk II (bron: https://nl.mouser.com/new/f-secure/crowd-supply-usb-Armorymkii/)

De usb Armory is een open source computer platform ter grote van een usb stick. Hij is ontwikkeld met Information Security Applications in mind. Het bijzondere van de USB Armory is dat hij een groot assortiment aan features heeft die ervoor zorgen dat het ideaal is voor het ontwikkelen van secure solutions. Enkele voorbeelden van deze features is Ethernet, USB storage en keyboard/mouse emulation, ARM trustzone, en een externe cryptographic co-processor.

12.1 2.1. USB Armory spec's

De usb Armory heeft de volgende hardware specificaties

Naam	Spec
CPU	NXP i.MX6UL 528MHz /i.MX6ULZ ARM® Cortex™-A7 900MHz
Werkgeheugen	512MB/1GB DDR3 RAM (afhankelijk van versie)
Videokaart	Geen
Opslag	Internal 16 GB eMMC + External SD kaart
aansluitingen	DRP (Dual Role Power) receptacle + UFP (Upstream Facing Port) plug, USB 2.0 only (*no video support*), via externe debug card UART, GPIO, SPI, I²C, CAN breakout
Afmeting	65 x 19 x 6 mm
besturingssysteem	Ondersteuning voor linux en android.
Connectiviteit	u-blox ANNA-B112 Bluetooth module

12.2 2.1.1 Verschil MX6UL en MX6ULZ

De Armory heeft afhankelijk van de variant een MX6UL of een MX6ULZ als CPU. De MX6UL heeft een snelheid van 538 MHz waar de MX6ULZ een hogere snelheid van 900 MHz heeft. De trade-off waardoor de MX6ULZ variant sneller is is dat deze niet beschikt over On the fly Bus Encryption Encryption (OTF-BEE) wat zorgt voor OTF DRAM encryptie. Tevens beschikt de MX6UL niet over de Data Co-processor. In plaats hiervan heeft hij een Cryptographic accelerator and assurance module (CAAM). Deze biedt een hashing en encryption API aan.

De MX6ULZ maakt gebruik van een data co-processor (DCP) voor zijn cryptografische functies.

In eerste instantie is er geen groot verschil tussen CAAM en DCP. DCP ondersteund enkel geen OTF-BEE. Tevens wordt een verwijzing gemaakt dat de DCP beschikte over een "subset aan CAAM functionaliteiten". Online waren er geen verwijzingen te vinden wat deze functionaliteiten waren. Hiervoor zou ik de API van de Armory moeten aanroepen. Echter op moment van schrijven is deze nog niet beschikbaar.

samenvatting:

Zowel de MX6UL als de MX6ULZ beschikken over functionaliteit voor hashing en crypto generation. Echter de MX6UL beschikt over OTF DRAM encryptie waar de MX6ULZ dit niet kan. Tevens maakt de MX6UL gebruik van pseudo number generation waar de MX6ULZ true number generation heeft. De clocksnelheid van de MX6ULZ 900MHz waar de MX6UL een respectabele 528 MHz heeft.

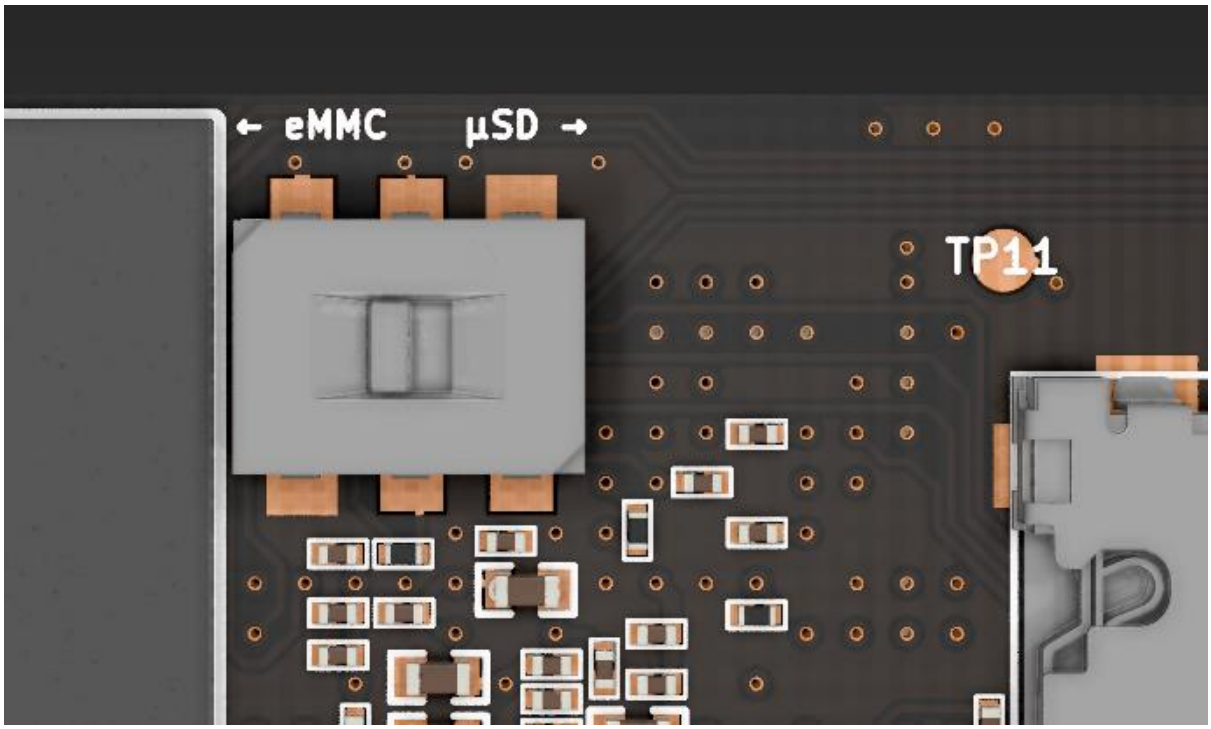
Functie	MX6UL	MX6ULZ
Snelheid	528MHz	900 MHz
Random number generator	Nee	Ja
OTF DRAM encryptie	Ja	Nee
Crypto module	CAAM	Data Co-processor

12.3 2.1.2 Werkgeheugen

De Armory komt in 512 MB en 1GB varianten. In de datasheet wordt vermeld dat de MX6UL een 512MB en 1GB variant heeft en de MX6ULZ standaard 1GB is. Echter worden er op verscheiden webstores op internet de MX6ULZ vermeld met 512MB aan RAM. Of dit een fout is of dat er na het publiceren van de datasheet een nieuwe versie is uitgekomen is nog onbekend. Op moment van schrijven zijn is er geïnformeerd bij de webshops naar deze verschillen, maar tot heden nog geen antwoord op gekregen.

12.4 2.1.3 Boot storage

Standaard beschikt de Armory over een interne eMMC storage van 16 GB. Tevens kan er via een externe SD kaart of via Serial Download Protocol (SDP) geboot worden. Door middel van een switch aan de zijkant van de Armory kan er geselecteerd worden of hij op het interne geheugen of vanaf het externe geheugen moet booten.



Afbeelding 2 – eMMC/SD card boot switch – (Bron: https://github.com/f-secure-foundry/usbArmory/wiki/Boot-Modes-(Mk-II))

Als de Armory niet in staat is om een bootable image te vinden doet hij het volgende :

- eMMC boot mode:
    i. Try eMMC
    ii. Fall back to microSD
    iii. Fall back to USB SDP mode
- microSD boot mode:
    i. Try microSD
    ii. Fall back to USB SDP mode

In eerste instantie zou dit gebruikt kunnen worden om een malicieus operating system te booten om een man in the middle (MitM) uit te voeren. Echter heeft de Armory de mogelijk om 4 signed pubic CA keys in de Armory fuse box te “branden”. Deze kunnen in combinatie met de security features van de Armory gebruikt worden om een enkel geverifieerde images te booten.

Samenvatting:

De Armory kan via het interne geheugen, via een externe SD kaart, of via SDP booten. Als hij geen bot image kan gebruiken maakt die gebruik van en fallback. Een MiTM attack kan voorkomen worden door de boot image te signen met Secure boot.

12.5 2.1.4 Aansluitingen

De Armory beschikt over DRP (Dual Role Power) female USB c poort, en een UFP (Upstream Facing Port) male USB c poort. De male USB poort wordt gebruikt als stroombron voor de Armory. Hoewel het beide USB c poorten zijn maken ze gebruik van USB 2 technologie. Dit houdt in dat ze beide een maximum doorvoersnelheid van 480 Mbits/s. Tevens ondersteunen deze poorten geen video output.

Het bijzondere van beide devices is dat ze verschillende devices kunnen emuleren. Ze kunnen gebruikt worden om het volgende te emuleren.

- Ethernet
- HID device (bijv. keyboard en mouse)
- Flashdrive.

Tevens kan er doormiddel via een extern breakout board (Sparkfun DEV 16368 Debug board) verbonden worden met GPIO, SPI, I^2C, en CAN breakout. Tevens stelt dit device Seriële (UART) verbinding over USB op te zetten met de Armory. Echter verlies je, doordat je het debug board moet in pluggen, wel de enige USB c poort.

12.6 2.1.4 besturingssystemen

Volgens de makers zou de de Armory MK 2 zou de meeste linux kernels moeten kunnen ondersteunen. Op het moment van schrijven zijn er twee pre-compiled Linux images beschikbaar, **Debian 10 (Buster),** en **Arch linux**. Tevens is er voor de debian image builtroot files beschikbaar om een eigen customized image te maken.

12.7 2.1.5 Connectiviteit.

Standaard beschikt de USB Armory enkel over een Bluetooth module voor draadloze communicatie. Om te verbinden met het internet moet je Armory gebruik maken van de usb emulatie functionaliteit van de host computer.

Standaard hebben de Debian en Archlinux image de Ethernet emulation functie enabled, en draaien hier een DHCP server achter. Als de Armory in een computer geplugd wordt zal deze via een network bridge voorzien kunnen worden van internet.

12.8 2.2 features

De Armory werd toen die uit kwam gepromoot als een “opensource USB computer ontworpen met security applicaties in gedachten”. Hierdoor heeft die ook een assortiment van security based features waar gebruik van gemaakt kan worden. In dit deel van het hoofdstuk vat ik deze samen en licht de bijzonder of relevante toe. De Armory beschikt over de volgende security features:

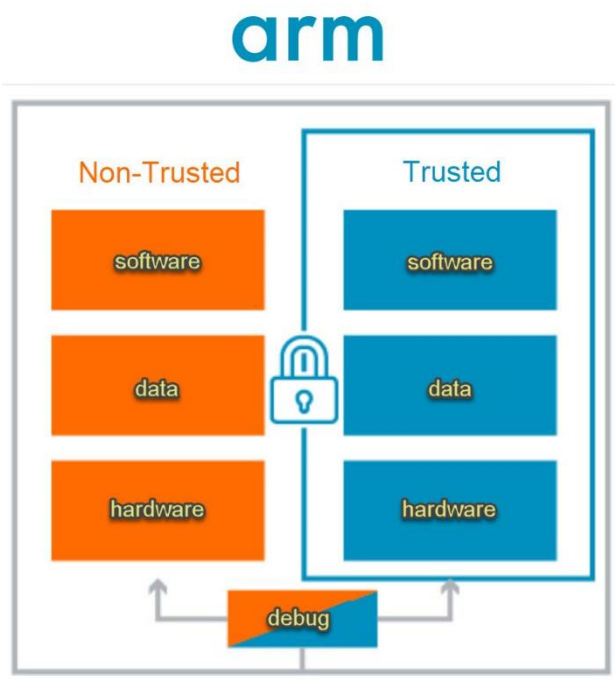
Naam	omschrijving
ARM trustzone	Scheiding tussen “veilige” en “onveilige” software resources maken
HABv4 Secure boot	On chip internal Boot rom authenticatie.
USB emulation	Emulatie mogelijkheden voor USB storage, HID devices (keyboard, mouse, joystick, etc) en Ethernet.
Bluetooth	V5.0 (Bluetooth low energy), theoretische snelheid van 1,4 MBits/s
Customizabe led	Blauwe en witte led.
Secure non volitile storage	een random 256 bit OTPMK key is fused voor encryptie/decryptie van userdata, alleen leesbaar voor de Data Co-processor (of CAAM, in geval van MX6UL model)
On the fly Ram encryption	On the fly external RAM encryptie, enkel voor MX6UL model
External Cryptographic Co-processor (MX6ULZ model) / crypto accelerator and assurance model (MX6UL model)	Levert functies voor hasing en encryptie voor security functionaliteiten via een externe chip.
eMMC Replay protected Memory blocks	Bescherming flash memory dmv een shared secret tussen de host en eMMC storage.

De twee features waar we iets dieper op in gaan (vanwege hun relevantie voor het project) zijn ARM trustzone en de DCP/CAAM.

12.9 2.3 Arm Trustzone

Security is een grote zorg voor embedded systemen, vooral voor systemen die verbonden zijn met het internet. Een gehackte embedded device biedt een grote hoeveelheid mogelijkheden voor hackers om misbruik te maken. Wat kan variëren van Distributed Denial of Service (DDoS) -aanvallen tot ongeautoriseerde toegang tot interne netwerken. Hierdoor is het dan ook belangrijk om bij het ontwikkelen voor embedded systemen het aanvals oppervlakte klein mogelijk te maken.

Arm trustzone is een techniek om een zogeheten “vertrouwde” en niet “vertrouwde” zone te creëren. Dit wordt uitgevoerd door een secure operating systemen en een normaal OS op 1 core te draaien.



De programmeur kan zelf bepalen wat er in Trustzone draait. Het kan een software libary omvatten of een heel operating system draaien in de trusted area. Non secure software kan niet bij de software in de trusted area en resources die zich hier bevinden.

Trustzone is gebaseerd op het principe van zo weinig mogelijk rechten. Dit houdt in dat system modules zoals driver geen toegang hebben tot een resource tenzij dit absoluut noodzakelijk is.

Figuur 3, Scheiding tussen trusted en non trusted zone (bron: https://www.microcontrollertips.com/embedded-security-brief-arm-trustzone-explained/#_edn1)

Als er communicatie plaats moet vinden tussen de trusted area, en non trusted area dan kan dit doormidden van de ARM software suite.

Conclusie.

Als Trustzone goed toepast wordt kan het aanvals oppervlakte van het embedded systeem drastisch gereduceerd worden. Echter is het dan wel kritisch dat de programmeur dit ook goed toepast.

Voor dit project is het belangrijk dat de VPN client veilig draait. Andere linux services zouden hier geen invloed op moeten hebben en zouden daarom prima in de non-trusted zone kunnen draaien.

Het enige nadeel van ARM trustzone is dat er preformance problemen kunnen ontstaan. Er draaien 2 operating systemen op 1 device, wat ervoor zorgt dat voor zware applicaties minder resources beschikbaar.

12.10 2.4 External cryptographic functionaliteit.

Zowel de Armory MX6UL als de Armory MX6ULZ beschikken over secure cryptoprocesor. De MX6UL beschikt over een Cryptographic accelerator and assurance module (CAAM) en de MX6ULZ beschikt over een Data Co-processor (DCP).

Op linux systemen maken programma's gebruik van de Linux kernel Crypto API. Dit is simpel gezegd een set aan cryptogratische functies die aangeroepen kunnen worden voor hashing en encryptie/decryptie functionaliteiten. Echter draait deze API in de kernel. In een normale situatie kan een programma niet zomaar bij de kernel (omdat deze in het operatingsystem draait). Echter als een systeem geïnfecteerd is met een virus kunnen de keys die gebruikt worden door de Crypto API aangepast worden wat heel de encryptie waardeloos maakt.

Om dit tegen te gaan maken sommige embedded systemen gebruik van secure crypto processors. Dit zijn losse gededicеerde chips die crypto grafische functies kunnen afhandelen. Deze co-processors hebben meerdere fysieke beveiliging lagen die geknoei voorkomen. De cryptoprocessor onthult geen sleutels of uitvoerbare instructies op een bus, behalve in gecodeerde vorm.

Voor dit project is het relevant om alle encryptie en hashing operaties op de cryptoprocessor te laten uitvoeren. Niet alleen zorgt dit voor een extra beveiligingen laag, maar tevens is het een stuk lastiger voor virussen om private keys te achterhalen op deze manier.

3. OpenVPN-NL

In dit project wordt gebruik gemaakt van een aangepaste versie van openVPN genaamd openVPN-NL. OpenVPN-NL is een versie van OpenVPN die is aangepast om zoveel mogelijk beveiligingsmaatregelen te bevatten die nodig zijn om in een geclassificeerde omgeving te werken.

Deze versie van OpenVPN is ontwikkeld door Fox-it, in samenwerken met de AIVD, voor Nederlandse ministeries die openVPN wilde gebruiken. Openvpn-NL is opensource en gratis te downloaden van de website van Fox-it

3.1 Verschillen tussen openvpn en openvpn-NL

Openvpn-nl en openvpn zijn vrijwel identiek. Het grootste verschil tussen de versie is dat openvpn-nl een aantal “onveilige” features uitgeschakeld of verwijderd zijn. Tevens is er gebruik gemaakt van een andere cryptografische libary. Dit heeft geen effect op de functionaliteit van openvpn. Configuratiefiles van beide versies zijn uitwisselbaar in gebruik.

	OpenVPN	OpenVPN-nl
onderhouder	OpenVPN community	Fox-it
Certificering	Geen	NLNCSA criteria Level 2, "Departementaal VERTROUWELIJK" (Dutch) if deployed in compliance with deployment advisory ("inzetadvies")
Funtionaliteit	Volledig	Onveilige opties verwijderd, of hardened, anders ongewijzigd
Cryptografische Libary	OpenSSL	Mbed TLS
Default encryptie	BF-CBC / AES-256-GCM, SHA1	AES-256-CBC / AES-256-GCM, SHA256 (andere opties niet toegestaan)
Toegestanne groepen voor (EC)DH	DH: 1024-8192 bits, ECDH: any supported curve	DH: 2048-4096 bits, ECDH: P-256, P-384
Acceptabele size RSA	1024-8192 bits	1024-4096 bits (using less than 2048 bits is advised against, and requires explicitly setting "--tls-profile legacy")

(Bron: https://openvpn.fox-it.com/about.html)

4. Conclusie analyse

Uit de analyse komt dat de USB Armory een goed assortiment heeft aan security features waar gebruik gemaakt van zouden kunnen worden voor het project. Zowel op hardware niveau als op software niveau zijn er verscheiden features die de gebruikt kunnen worden om het eind product veiliger of meer functionaliteit te geven.

Het grootste nadeel aan de Armory is dat hij wel een Bluetooth module heeft geen WiFi/ethernet module. Hiervoor zou er gebruik gemaakt moeten worden van een externe Wi-Fi dongel om de Armory van internet te voorzien.

Bij het onderzoek naar OpenVPN-NL bleek dat deze in vrijwel alle aspecten gelijk is aan een normale instantie van OpenVPN. Een openVPN client zou prima met een OpenVPN-nl server moeten kunnen verbinden en vis versa mits ze dezelfde security protocollen gebruiken.

Bijlage C: Applicatie documentatie

Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie bij Technolution BV. Gouda

Bart de Langen
15077071
HBO-ICT Network & System Engineering
De Haagse Hogeschool Delft
Versie 0.3
26 maar 2021

13 Inleiding

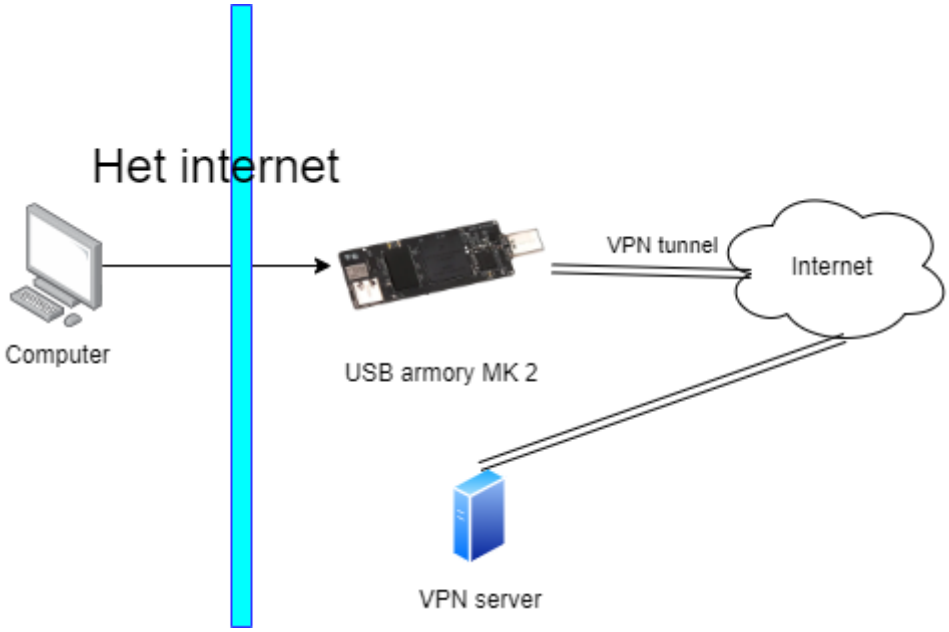
Dit document in gewijdt aan het de overdracht van het prototype voor het afstudeerproject van Bart de Langen. In dit afstudeerproject werd er een proof of concept ontwikkeld. Het doel van POC was om een hardmatige USB-stick te ontwikkelen die een gebruiker in zijn computer kon steken. Hierna zetten de USB-stick een veilige VPN tunnel op en deelde deze met de computer. Er werd hierbij gebruik gemaakt van een USB armory als platform. Het idee hierachter was dat de computer een veilige internetverbinding kon krijgen zonder ooit verbonden te zijn geweest het internet.

In het eerste deel van het document wordt er dieper ingegaan op de opdracht en welke eisen hieraan verbonden waren. Het tweede deel is gededicеerd aan het bouwen van een geupdate image voor de USB armory (er is een image beschikbaar voor easy deployment). In het laatste deel wordt de code van de Proof of concept uitgelegd, en worden de gemaakte keuzes beargumenteerd.

14 De opdracht en platform.

Het doel van dit project is het maken van een demonstratieproduct waarbij er een scheiding is aangebracht tussen de computer en het internet. Dit wordt bereikt doormiddel van een USB armory die functioneert als VPN-module. De Usb armory vormt een soort muur tussen de computer en het internet.

Zodra de gebruiker de USB armory in de computer steekt maakt deze verbinding met internet en probeert vervolgens een veilige VPN-tunnel op te zetten. Als de VPN-tunnel opgezet is deelt hij deze verbinding met de computer door middel van Ethernet Emulation. Op deze manier heeft de computer een veilige internetverbinding zonder dat hij verbonden was met het “onveilige internet.”



14.1 Platform

Voor het platform wordt er gebruik gemaakt van een USB armory MK 2(900 MHz CPU variant). De USB armory heeft standaard geen Wi-Fi of ethernet mogelijkheden. Via een Ralink RT5370 Wi-fi USB adapter kan hij verbinden met een Wi-Fi netwerk. Voor het werkend krijgen van deze USB adapter moet de kernel wel re-compiled worden (dit wordt in een later hoofdstuk besproken).

14.2 Besturingsysteem

Voor het proof of concept product maakt de Armory gebruik van **Debian** 10 (buster) .

In theorie zou iedere distro hierop moeten kunnen draaien, maar zou deze wel eerst uitgerust moeten worden met de juiste drivers en programma’s.
Er zijn 2 kant en klare images beschikbaar kant klaar enkel op de Armory geïnstalleerd hoeven worden: **Arch linux**, en **Debian 10**. De Debian 10 variant is door de fabrikant van de Armory (f-secure) gemaakt en krijgt maandelijkse updates. Arch linux is door de community ooit opgezet. De laatste update die deze heeft gekregen was al bijna meer dan een jaar geleden. Doordat de Debian variant maandelijkse updates krijgt van de fabrikant is dit het meest geschikte besturing systeem op te gebruiken.

15 Image opzetten.

Er is een image beschikbaar (https://gitlab.technolution.nl/ha_incubator/primewifi-poc) die direct weggeschreven kan worden op een SD kaart. Deze image is weer een aangepaste variant van de F-secure base image (https://github.com/f-secure-foundry/usbarmory-debian-base_image) .

De image hoeft enkel op een SD kaart weggeschreven worden met een programma als bijvoorbeeld Win32 Disk Imager. Hierna het Proof of concept out of the box moeten werken. Via 10.0.0.1 kan er via SSH op de image ingelogt worden. Het default username/password is **usbarmory .** In de folder **/home/usbarmory/gui** staat de applicatie. Deze kan je starten met het commando
`sudo python3 app.py`

Navigeer hierna naar 10.0.0.1:5000 in je webbrowser om in te loggen. Het standaard account is username **admin** en als password ook **admin**

In de rest van dit hoofdstuk wordt besproken hoe er een nieuwe image (met geupdate kernel bijvoorbeeld) gemaakt kan worden.

15.1 Opzetten imaging omgeving.

In dit hoofdstuk wordt besproken hoe een omgeving opgezet kan worden om de f-secure usb armory base image te compiler. Hierbij wordt er uitgegaan van een **Ubuntu** of **Debian** omgeving waarbij de **loop** kernel module enabled/loaded moet zijn..

Zorg dat de volgende packages geïnstalleerd zijn met het volgende commando:

```
$sudo apt-get install -y bc binfmt-support bzip2 fakeroot gcc gcc-arm-linux-gnueabihf git gnupg make parted rsync qemu-user-static wget xz-utils zip debootstrap sudo dirmngr bison flex libssl-dev kmod git
```

Importeer de linux signing GPG key:

```
gpg --keyserver hkp://keys.gnupg.net --recv-keys 38DBBDC86092693E
```

import de u-boot signing GPG key:

```
gpg --keyserver hkp://keys.gnupg.net --recv-keys 147C39FF9634B72C
```

Als volgende stap downloaden we de Make file om de image te creeeren van de f-secure foundery github

```
Git clone https://github.com/f-secure-foundry/usbarmory-debian-base_image.git
```

15.2 aanpassen kernel header file.

Om de usb armory met de Ralink RT5370 Wi-fi USB adapter te laten werken moet de kernel gecompiled worden met deze driver hiervoor. Hiervoor moeten we een .config genereren, en deze vervangen voor we de image gaan compileren.

Allereerst check welke kernel gebruikt wordt in de Makefile van de git clone die we gemaakt hadden in de vorige stap. Dit zou boven aan de make file moeten staan.

336 lines (292 sloc) | 19 KB

```
1  SHELL = /bin/bash
2  JOBS=2
3
4  LINUX_VER=5.4.87
5  LINUX_VER_MAJOR=$(shell echo ${LINUX_VER} | cut -d '.' -f1,2)
6  KBUILD_BUILD_USER=usbarmory
7  KBUILD_BUILD_HOST=f-secure-foundry
8  LOCALVERSION=-0
9  UBOOT_VER=2020.10
10 ARMORYCTL_VER=1.1
```

De makkelijkste manier om de .config file te genereren en aan te passen dat die de juiste driver download is het volgende:

1. Login op Spike.
2. Download de kernelpacakage met het volgende commando (aanpassen naar jou versie natuurlijk

```
wget https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.4.87.tar.xz
```

3. Untar het met het commando, en ga de map in

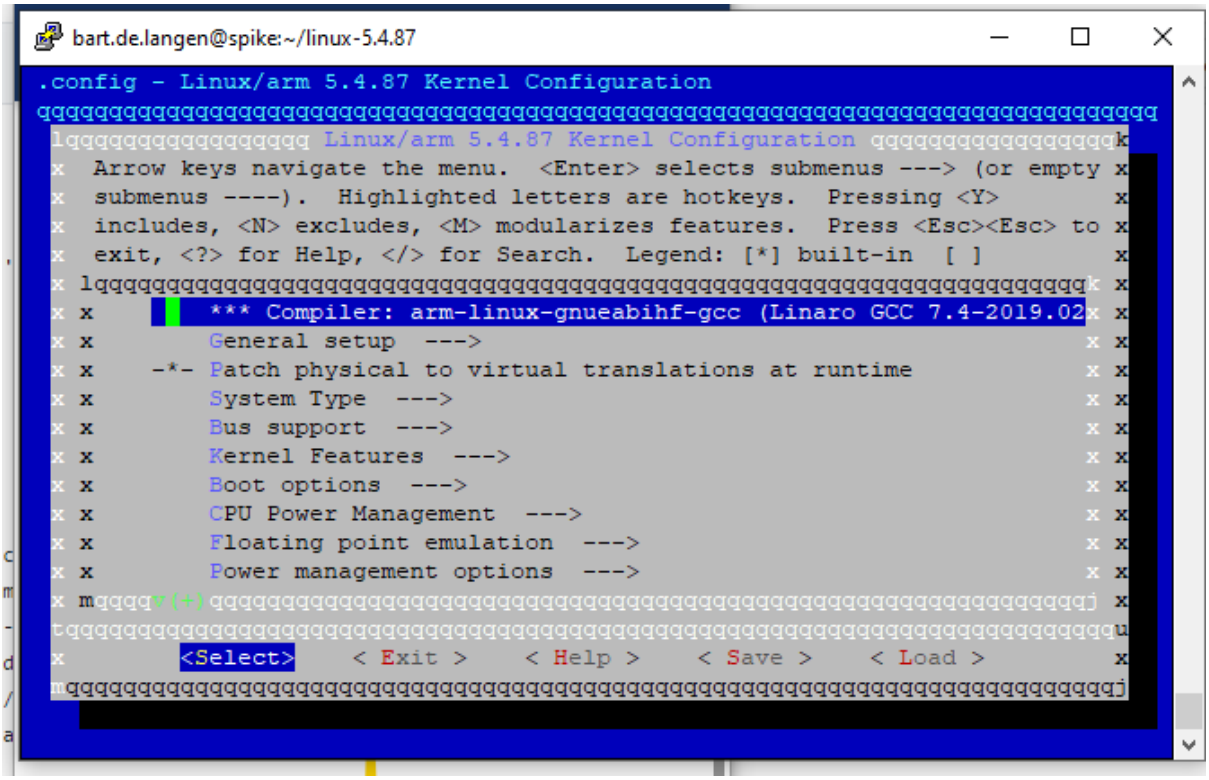
```
tar -xf linux-5.4.87.tar.xz && cd linux-5.4.87/
```

4. Laat de juiste enviroment in voor de crosscompiler, commando kan verschillen als spike is geupdated, check intranet voor de juiste enviroment..

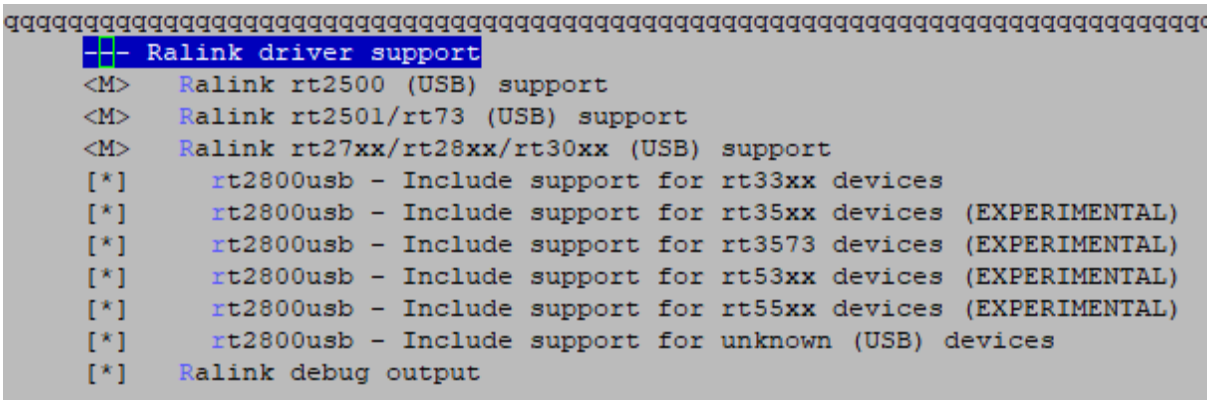
```
tl_env gcc-linaro-arm-linux/gnueabihf-7.4-2019.02
```

5. Laad menuconfig om de .config aan te passen

```
make  ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```



6. Zorg dat de ralink rt28xx usb drivers genabled zijn. Deze zouden onder device drivers -> network device support -> wireless lan -> ralink evices ->ralink driver support moeten staan .



7. Sla de .config op door op de **save** knop te drukken.
8. Download de .config file nu van spike naar je image omgeving (met bijvoorbeeld filezilla) .config files zijn hidden, dus mogelijk moet je show hidden files optie aanzetten.

Voor de volgende stap passen we de Makefile van de github clone aan zodat die onze .config file gebruikt. Zoek in de Makefile waar de .config file gedownload wordt. Comment deze line uit en vervang het voor een line die jou .config file hiernaartoe kopieerd. Bijvoorbeeld:

```
Cp /home/bart/.config linux-${LINUX_VER}/.config
```

```
        gpg --verify linux-${LINUX_VER}.tar.sign; \
        tar xfm linux-${LINUX_VER}.tar && cd linux-${LINUX_VER}; \
fi
wget ${USBARMORY_REPO}/software/kernel_conf/${V}/usbarmory_linux-${LINUX_VER_MAJOR}.config -O linux-${LINUX_VER}/.config
if test "${V}" = "mark-two"; then \
        wget ${USBARMORY_REPO}/software/kernel_conf/${V}/${IMX}-usbarmory.dts -O linux-${LINUX_VER}/arch/arm/boot/dts/${IMX}-usbarmory.dts; \
fi
cd linux-${LINUX_VER} && \
```

```
165         fi
166         #wget ${USBARMORY_REPO}/software/kernel_conf/${V}/usbarmory_linux-$
    {LINUX_VER_MAJOR}.config -O linux-${LINUX_VER}/.config
167         cp /home/bart/config linux-${LINUX_VER}/.config
168         if test "${V}" = "mark-two"; then \
169                 wget ${USBARMORY_REPO}/software/kernel_conf/${V}/${IMX}-
    usbarmory.dts -O linux-${LINUX_VER}/arch/arm/boot/dts/${IMX}-usbarmory.dts; \
```

Draai hierna het volgende commando om de image te builden

```
# For the USB armory Mk II (external microSD)
make V=mark-two IMX=imx6ulz BOOT=uSD

# For the USB armory Mk II (internal eMMC)
make V=mark-two IMX=imx6ulz BOOT=eMMC
```

Het make process duurt ongeveer 35 minuten, hierna zou er een file geproduceerd moeten zijn met de volgende naming instance:

```
usbarmory-mark-two-debian_buster-base_image-YYYYMMDD.raw
```

15.3 wegschrijven naar SD kaart:

schrijf de .raw image weg met het commando, vervang hier **sdX** met de **microSD** of **eMMC** device

```
sudo dd if=usbarmory-*-debian_buster-base_image-YYYYMMDD.raw of=/dev/sdX bs=1M
conv=fsync
```

op Windows, en andere OSen kan als alternatief Etcher hiervoor gebruikt worden

15.4 verbinden met armory.

als de image succesvol naar een SD kaart is weggeschreven zal er een wit lichtje knipperen (heartbeat) nadat deze is ingeplugt in een computer. Na enkele momenten zou je via SSH met de usb armory moeten kunnen verbinden via het ip 10.0.0.1. De gebruikersnaam is **usbarmory** en het wachtwoord is ook **usbarmory**.

15.5 aanpassen config

Debian heeft in deze versie geen standaard interface namesn (wlan0, eth0, etc). dit fixen met het volgende commando:

```
sudo ln -s /dev/null /etc/udev/rules.d/80-net-setup-link.rules
```

het tweede probleem is dat de DHCP vanuit de armory standaard geen gateway meegeeft. Dit is nodig voor de app om al het netwerk verkeerr van de computer naar de armory door te sturen.

Dit fixen we door de /etc/dhcp/dhcpd.conf aan te passen. Vervang de dhcpd.conf met de volgende tekst

```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

subnet 10.0.0.0 netmask 255.255.255.0 {
  range 10.0.0.2 10.0.0.2;
  option routers 10.0.0.1;
  option domain-name-servers 8.8.8.8 8.8.4.4
  default-lease-time 600;
  max-lease-time 7200;
}
```

De applicatie maakt gebruik van legacy ip tables. Enable deze met het volgende commando:

```
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
```

de recompiled kernel zal standard nog niet werken met de relink usb dongle. Hiervoor moet een extra package geinstalleerd worden. Allereerst edit de /etc/apt/sources.list, voeg hier achter de 1e en 2e lijn **non-free** achter.

```
deb http://ftp.debian.org/debian buster main non-free
deb http://ftp.debian.org/debian buster-updates main non-free
deb http://security.debian.org buster/updates main
deb https://f-secure-foundry.github.io/debian stable usbarmory
```

Hierna kan de driver geinstalleerd worden met het commando

```
Sudo apt-get update && sudo apt-get install firmware-ralink -y
```

als laatste staap heft de usb armory ook nmcli als dependencie nodig. Installer dit met de volgende commando’s

```
apt-get install network-manager
systemctl start networkmanager.service
systemctl enable networkmanager.service
```

15.6 Opzetten no vnc

Voor de captive portal vncConnector klasse zijn een paar dependencies nodig om een VNC server te kunnen runnen.

Allereerst installeer een GUI. XFCE is een redelijk light gewight gui die geinstalleerd kan worden met het volgende commando:

```
Sudo apt-get install xfce4 xfce4-goodies
```

Installeer tightvnc met het volgende commando

```
sudo apt-get install tightvncserver
```

Start de VNC server voor de initiële setup met het volgende commando. Voer vervolgens **vncserver** als password in, en geef een n bij de vraag voor een view only password.

```
$ sudo vncserver
```

Output:

```
You will require a password to access your desktops.

Password:
Warning: password truncated to the length of 8.
Verify:
Would you like to enter a view-only password (y/n)? n
xauth:  file /root/.Xauthority does not exist

New 'X' desktop is usbarmory:2

Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/usbarmory:2.log
```

Hierna installeren we noVNC en de webbrowser chromium. noVNC functioneert als web based VNC client, chromium wordt gebruikt zodat de gebruiker de captive portal via noVNC kan bezoeken

Installeer deze met het commando

```
Sudo apt-get install novnc websockify python-numpy chromium -y
```

15.7 Installeren applicatie

De applicatie maakt gebruik van python. Installeer dit met het volgende commando

```
sudo apt-get install python3 python3-pip -y
```

Als volgende stap downloaden we de applicatie

```
git clone https://gitlab.technolution.nl/ha_incubator/primewifi-poc.git
```

Navigeer naar de gekloonde git map. Als het goed is is er een txt file genaamd requirements.txt gedownload. Dit bestand bevat de benodigde modules. Deze modulen kunnen automatisch geïnstalleerd worden met het volgende commando.

```
Pip3 install -r requirements.txt
```

Mocht een package niet goed geinstallerd kunnen worden, in de map **dist-packages** zijn de packages los te vinden. Deze kan je handmatig in de python libary folder plaatsen

15.8 Starten applicatie

In de **webapp** folder bevind zich de applicatie. Navigeer naar de map. Met het volgende commando kan de applicatie gestart worden

```
Sudo python3 app.py
```

Navigeer hierna naar 10.0.0.1:5000 in je webbrowser om in te loggen. Het standaard account is username **admin** en als password ook **admin**

.

16 Conclussie en advies

In dit project was het de bedoeling om een VPN module te maken die een beveiligde verbinding kon opzetten voor de host pc. Dit zonder dat de host pc ooit hoefde te verbinden met “onbeveiligt” internet. Voor dit project was de USB armory als platform gekozen.

Tijdens het ontwikkelen van de module zijn er verschillende problemen naar voren gekomen waardoor de USB armory geen geschikt platform is voor dit doeleinden. Voor een VPN-module is het belangrijk dat hij op een makkelijke manier een externe verbinding met het internet kan maken. Eventueel via een ethernet kabel of via een Wi-Fi verbinding. De usb armory heeft standaard geen ingebouwde Wi-Fi chip. Dit moest bereikt worden met een externe Wi-Fi dongel, wat ervoor zorgde dat de USB armory groot en onhandig was in gebruik. Ook heeft de USB armory enkel usb C poorten. Voor de meeste computer is usb B nog steeds de standaard. Dit zorgde ervoor dat er gebruik gemaakt moest worden van een usb b naar usb c converter om de usb armory te gebruiken, wat hem nog groter en onhandiger in gebruik maakte.

Het tweede argument waarom de usb armory geen geschikt platform is om een VPN-module mee te maken is dat hij hier niet voor ontwikkeld is. De USB armory is ontwikkeld voor security based applicaties. Het heeft een externe crypto grafische coprocessor, ARM trustzone, usb device emulation, en vele andere securityfeatures die geen directe nut voor het product hebben. Ze kunnen gebruikt worden om de beveiliging van het eindproduct te verbeteren, maar vaak zijn er softwarematige oplossingen die dit beter kunnen implementeren.

Het laatste argument is dat de prijs door de extra security features een stuk duurder dan nodig is. De USB armory kost 170 euro, en komt met een 900 MHz Cortex CPU, en 512 mb aan ram. Voor de zelfde prijs is een W5 pro mini stick te koop (https://www.amazon.nl/W5-Stick-Windows-ondersteuning-Bluetooth/dp/B07MB9TC33) met 4 keer zoveel computerkracht. Op deze computer kunnen zaken als disk encryptie ook gewoon geïmplementeerd worden.

Als er een gelijk soort product ontwikkeld wordt moet er ook goed over nagedacht worden hoe de openvpn processen aangestuurd worden. Standaard heeft openVPN voor de client geen manier om de configuratie live aan te passen. De openVPN service moet herstart worden om de wijzingen door te voeren. Dus als de module bijvoorbeeld meerdere vpn tunnels heeft, en 1 hiervan wordt gewijzigd moeten alle services opnieuw opstarten.

Indien het proof of concept door ontwikkeld wordt zijn er nog een aantal bugs en verbeteringen die doorgevoerd zouden kunnen worden:

- de applicatie draait momenteel onder root, om de service veiliger te maken zou er een speciale vpn module groep moeten komen die de benodigtde commando’s kan draaien.
- De webGUI is momenteel niet alleen via de host pc bereikbaar, maar ook via de wi-fi connectie. Dit zou opgelost kunnen worden door apache alleen te laten luistern naar eth0
- Een bug die momenteel erin zit is dat als een connectie van full internet, terug gaat naar een captive portal, dan crasht de vnc server
- De applicatie is ontwikkeld met Flask, de built in flask server is niet geschikt voor productie. De app zou op een WSGI container gehost moeten worden
- De VNC solution voor de captive portal is een slecht uitgewerkt idee. Hier zou gewoon iets deticateds voor gemaakt moeten voor. Voor demo doeleinden werkt het prima.

Bijlage D: Applicatie documentatie

Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie bij Technolution BV. Gouda

Bart de Langen
15077071
HBO-ICT Network & System Engineering
De Haagse Hogeschool Delft
Versie 0.3
26 maar 2021

17 Inhoudsopgave

5. Inleiding

We zien steeds meer behoefte om de data veilig tot in het end-point device te brengen. Ook wel zero-trust networking genoemd. Daarnaast groeien de dreigingen. Zo geven de recente kwetsbaarheden in de hardware van PC platformen voldoende aanleiding om niet meer volledig op de software-scheiding van operating systemen te kunnen vertrouwen. Traditionele software VPN-oplossingen bieden niet voldoende weerstand tegen hoge dreigingen. Dit geeft voldoende aanleiding om hardware geïsoleerde VPN-oplossingen te creëren.

De afstudeer opdracht is het ontwikkelen van deze oplossing. Hierbij moet er demonstratieproduct ontwikkeld worden waarbij er een scheiding is aangebracht tussen het pc-platform en het internet. Dit wordt ontwikkeld op een USB armory mk ii. De eindgebruiker zou alleen de USB armory hoevn in te pluggen. Hierna zal de USB armory eventuele authenticatie afhandelen, en een secure VPN tunnel opzetten.

6. Project

In dit hoofdstuk wordt de opdracht gedefinieerd. Hierbij wordt de probleemstelling, het doel, en de betrokken personen besproken.

6.1 Probleemstelling

Afgelopen jaren is er steeds meer behoefte om (bedrijfs)data veilig van endpoint tot endpoint te brengen. Dit is goed te zien tijdens de corona crisis waar mensen veeluit vanuit huis werken, en bedrijf data niet perse meer binnen het pand blijft. Traditionele VPN oplossingen bieden niet altijd meer voldoende weerstand tegen hoge dreigingen. Dit geeft Technolution voldoende aanleiding om te kijken naar hardware geïsoleerde VPN-oplossingen

6.2 Doelstelling

Het doel van de opdracht is om in 20 weken een hardware VPN oplossing te ontwikkelen die die op veilige wijze een VPN verbinding opzet zonder dat het pc-endpoint verbonden hoeft te zijn met het internet.

6.3 Betrokken personen/instanties

Deze opdracht wordt uitgevoerd door de afstudeerder Bart de Langen bij Technolution BV. Het afstudeerverslag zal uiteindelijk door de Haagse hogeschool beoordeeld worden of deze voldoende is om de afstudeerstage succesvol af te sluiten.

6.3.1 Betrokken instanties

- De Haagse hogeschool
  Faculteit IT & Design
  Rotterdamseweg 137
  2628 AL Delft
  Tel: 015-2606200

- Technolution B.V.
  Burgemeester Jamessingel 1
  2803WV gouda
  Tel: 0182 594 000

6.3.2 betrokken personen

- opdrachtgever.
  Jonathan Hofman
  Burgemeester Jamessingel 1
  2803WV gouda
  Tel: 0182 594 000

- Bedrijfsmentor
  Roy Rietveld
  Burgemeester Jamessingel 1
  2803WV gouda
  Tel: 0182 594 000

- Eerste examinator
  -
  Rotterdamseweg 137

2628 AL Delft
Tel: 015-2606200

- Tweede Examinator
  Maris Marinus
  Rotterdamseweg 137
  2628 AL Delft
  Tel: 015-2606200

- Afstudeerder
  Bart de Langen
  Brasserskade 63
  2612CA Delft

7. Tijdverdeling

In dit hoofdstuk bespreek ik de tijdverdeling van het project. Tijdens het project wordt er gebruik gemaakt van de Rapid Application development (RAD) methodiek. Dit is een incrementiele softwareontwikkelingen methodiek waar iedere increment (vanaf nu genoemd, sprint) voortbouwt op de stap ervoor. In iedere sprint wordt een stuk software ontworpen, gemaakt, en getest. In de tabel is een overzicht welke document bij welke fase horen.

Fase	activiteit	Tijd duur
Oriëntatiefase	Maken van een plan van aanpak. Maken van een analyse document (Mogelijk) literatuur onderzoek	4 weken
Incrementeel werken (sprint)	Ontwerpen van software	3 iteraties van ieder 4 weken
	Ontwikkelen software	
	Testen software	
Afronden project	Afronding documentatie, project en evt bugfixes.	2 weken.

Tabel 1 – Tijdverdeling/activiteiten project.

8. Producten en project methodiek

In dit project wordt er gebruik gemaakt van de RAD methodiek. In dit project zijn er dynamische producten die tijdens het project kunnen veranderen (omdat bijvoorbeeld de eisen aangepast worden). Deze producten horen bij het deel “incrementeel werken”. Tijdens het project worden de volgende producten opgeleverd.

- Orientatiefase
  - Plan van eisen/analyse rapport
- Incrementeel werken
  - Ontwerp rapport
  - Testrapport
  - Code hardware VPN module
- Afronding project
  - Afstudeer verslag
  - Handleiding software voor het opzetten.
  - MakeFiles om image te compilen.

9. Globale Planning

In de tabel hieronder is een globale planning ingepland voor het project. Mogelijk kan er vanaf gewijkt worden als blijkt dat het tijdbestek te veel of te weinig is. De tabel begint op 12 oktober 2020 en eindigt op 1 maart 2021. In de tabel is ervan uitgegaan van 3 incrementiele fases.

Weeknr.	42	43	44	45	46	47	48	49	50	51	52	1	2	3	4	5	6	7	8	9
Fase																				
Oriëntatiefase	■	■	■	■																
Sprint 1					■	■														
Sprint 2							■	■												
Sprint 3									■	■										
Sprint 4											■	■								
Sprint 5													■	■						
Sprint 6															■	■				
Sprint 7																	■	■		
Afronding																			■	■

10. Grenzen en randvoorwaarden.

Om de kwaliteit van het eindproduct te waarborgen zijn er een aantal grenzen en randvoorwaarden opgesproken. Deze zorgen ervoor dat het project soepel en volgens planning verloopt.

10.1 Grenzen

Voor het project gelden de volgende grenzen

- Het project wordt uitgevoerd in een werkweek van 40 weken
- De project duur is 20 weken
- Het project wordt opgezet op een USB armory MK 2
- Voor de VPN verbinding wordt er gebruik gemaakt van OpenVPN-nl
- Er wordt een USB armory MK 2 ter beschikking gesteld aan de afstudeerder
- Eventuele extra connectors en kabels kunnen geleend worden

10.2 randvoorwaarden

De volgende randvoorwaarden zijn van toepassing om het project te laten slagen

- De hardware functioneert zonder defecten
- Er wordt wekelijks een afspraak met de opdrachtgever gemaakt om de voorgang van het project te bespreken, en eventueel bij te stellen.

10.3 kwaliteitszorg

Om de kwaliteit van het eindproduct te kunnen garanderen wordt gebruikt gemaakt van het volgende:

- GitHub, voor de opslag van de code
- Object georiënteerd ontwerpen en programmeren (Waar mogelijk)
- Object georiënteerde programmeerparadigma

11. Risico Analyse

Nummer	1.
risico	Vertraging afstudeer opdracht door ziekte of corona
omschrijving	De afstudeerder wordt ziek waardoor hij langere tijd niet aan de opdracht kan werken.
kans	Gemiddeld
impact	Groot: de opdracht loopt zo veel vertraging op dat deze niet op tijd afgerond kan worden
Actie(Preventief)	Vanuit huis werken, grote groepen mensen vermeiden.
Actie(Correctief)	Kijken of opdracht aangepast kan worden, kijken of verlening mogelijk is.

Nummer	2.
risico	Onvoldoende begeleiding door uitval bedrijfsmentor
omschrijving	Door uitval van de bedrijfsmentor is er te weinig begeleiding tijdens het afstuderen.
kans	Gemiddeld.
impact	Klein: de afstudeerder krijgt minder begeleiding. Dit hoeft niet persé een groot probleem te zijn. De bedrijfsmentor is niet de bedrijfsbegeleider. Eventueel zou een van de andere collega's de taak op kunnen pakken.
Actie(Preventief)	Wekelijks contact met de bedrijfsmentor waardoor er altijd voldoende begeleiding is.
Actie(Correctief)	Contacteren bedrijfsbegeleider. Het vinden van een nieuwe bedrijfsmentor.

Nummer	3.
risico	Opdrachtgever/bedrijfsbegeleider valt weg.
omschrijving	Door ziekte of omstandigheden zou de bedrijfsmentor/opdrachtgever wegvallen.
kans	Gemiddeld.
impact	Gemiddeld - Hoog: Als dit in de eerste weken gebeurd (waar de opdracht vorm krijgt) zou dit een grote impact zijn. Als dit later in het afstudeer traject zou gebeuren is de impact minder omdat er al minder begeleiding is i.v.m. thuiswerken
Actie(Preventief)	Wekelijks contact met de bedrijfsbegeleider/opdrachtgever.
Actie(Correctief)	In overleg zou de bedrijfsmentor de taak van opdrachtgever/bedrijfsbegeleider kunnen overnemen. Wijziging doorgeven aan de HHS

Nummer	4.
risico	Hardware is niet leverbaar / functioneert naar behoren
omschrijving	De hardware waar de opdracht op uitgevoerd wordt functioneert niet naar behoren of is niet leverbaar.
kans	Klein
impact	Klein: Het platform waar op ontwikkeld wordt is een veredelde Linux omgeving. De software zou ook ontwikkeld kunnen worden op een

	andere linux omgeving en vrij makkelijk overgezet kunnen worden naar andere linux omgevingen.
Actie(Preventief)	Hardware vroeg bestelen, analyse uitvoeren of de hardware geschikt is voor project
Actie(Correctief)	Opdracht uitvoeren op een ander linux based platform (bijvoorbeeld raspberry pi). In een latere fase in het project de software overzetten naar de USB armory

Nummer	5.
risico	Het is niet mogelijk Openvpn-nl te draaien op hardware
omschrijving	Het is niet mogelijk om de openvpn software te draaien op een USB armory mk 2
kans	Klein
impact	Gemiddeld: Als openVPN-nl niet zou draaien op de USB armory dan zou er gekeken moeten worden naar een ander platform, andere distributie, of een andere vpn solution
Actie(Preventief)	Onderzoeken of er een versie beschikbaar is van openvpn-nl voor de gekozen linux distributie, deze tijdens project uitvoerig testen
Actie(Correctief)	Kijken of openvpn-nl wel wilt draaien als er een andere linux distributie geinstalleerd wordt op de USB Armory, andere kijken naar een ander platform/vpn solution

Nummer	6.
risico	Hardware is niet krachtig genoeg
omschrijving	de hardware is niet krachtig genoeg om de software te kunnen draaien
kans	gemiddeld
impact	Groot: De software zou wel kunnen draaien, maar met een latency waardoor deze onbruikbaar wordt
Actie(Preventief)	Controle in Analyse rapport of de USB armory aan de minimum specs voldoen. Tijdens maken van software efficient programmeren zodat hardware niet onnodige cycles hoeft te maken
Actie(Correctief)	Kijken of software verder geoptimaliseerd kan worden. Andere programmeer taal gebruiken die minder resource intensief is.

Nummer	7.
risico	Usb armory kan niet met Wi-fi of ethernet verbinden
omschrijving	De hardware kan niet via een extern usb device verbinden met het internet
kans	Klein
impact	Gemiddeld: Er is altijd een reden dat een usb Wi-Fi dongel niet werkt, waarschijnlijk zouden driver problemen dan opgelost moeten worden.
Actie(Preventief)	Checken of de juiste drivers beschikbaar zijn op gekozen linux distributie zijn voor de te gebruiken hardware
Actie(Correctief)	Drivers handmatig installeren op Linux distributie, andere Wi-Fi dongel gebruiken, ergste geval andere Linux distributie uitkiezen waar de Wi-Fi dongel wel werkt

Nummer	8.
risico	De opdracht wordt uitgevoerd buiten de planning door inschattingsfouten/tegenslagen
omschrijving	Door inschattingsfouten of inschattingsfouten wordt de opdracht uitgevoerd buiten de verwachte planning
kans	Klein
impact	Gemiddeld: Als de opdracht niet meer volgens de tijdsplanning wordt uitgevoerd, dan kan het ervoor zorgen dat de procesvoering van het project niet meer klopt
Actie(Preventief)	Na iedere fase en sprint reflecteren. Reflectie meenemen naar de volgende fase
Actie(Correctief)	De opdracht aanpassen dat deze wel binnen de tijdscope past.

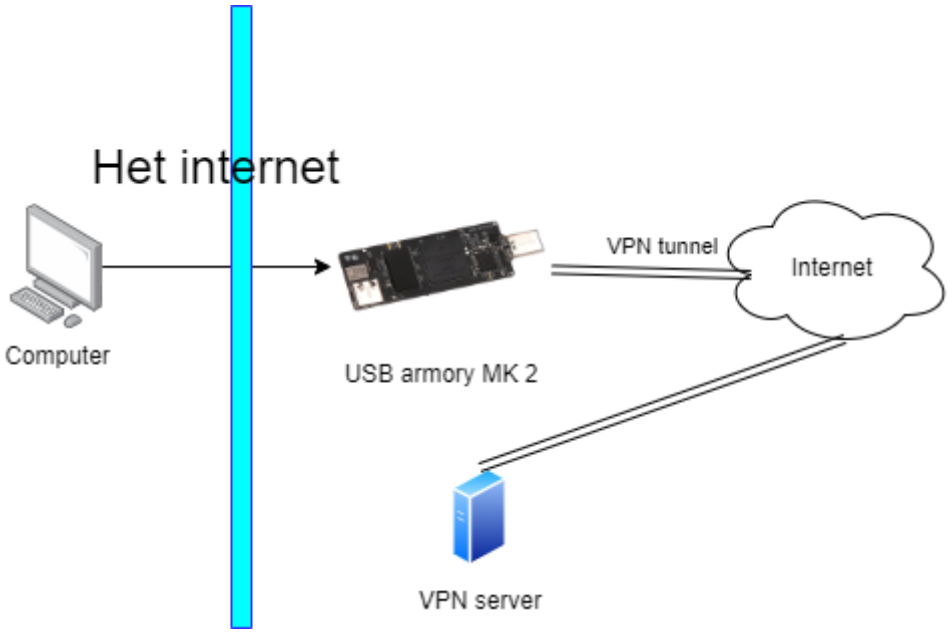
12. Programma van eisen project

In dit hoofdstuk worden de achterhaalde eisen van de opdrachtgever voor het product beschreven.

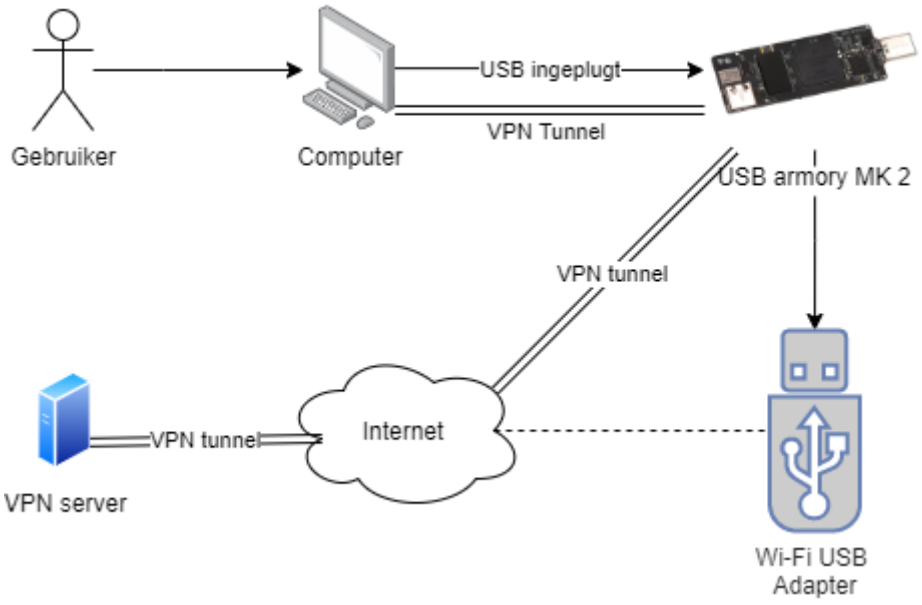
17.1 8.1. Doel Project

Het doel van dit project is het maken van een demonstratieproduct waarbij er een scheiding is aangebracht tussen de computer en het internet. Dit wordt bereikt doormiddel van een USB armory. De USB armory vormt een soort muur tussen de computer en het internet.

Zodra de gebruiker de Armory in de computer steekt maakt deze verbinding met internet en probeert vervolgens een veilige VPN tunnel op te zetten. Als de VPN tunnel opgezet is deelt hij deze verbinding met de computer doormiddel van Ethernet Emulation. Op deze manier heeft de computer een veilige internet verbinding zonder dat hij verbonden was met het “onveilige internet.”



17.2 8.2 Hardware omgeving.



Voor dit project bestaat de hardware omgeving uit een willekeurige computer, een usb armory 2, en een wifi Dongle. De USB armory zou op iedere willekeurige computer moeten werken. Zodra deze ingeplugd is gedraagt deze zich als een ethernet dongle die de computer van internet voorziet.

Standaard heeft de USB armory geen internet mogelijkheden. Om deze reden wordt er gebruik gemaakt van een Wi-Fi dongle om hem van een internet connectie te voorzien. In figuur … is een overzicht hiervan te zien.

17.3 8.3 Programmeer taal

Voor dit project zou de opdrachtgever het demonstratie product graag geprogrammeerd zien in de programmeertaal Rust. Echter is de afstudeer vrij om een andere taal te gebruiken als dit tot tijd te kort zou leiden.

17.4 8.4 Niet functionele eisen.

Uit de gesprekken met de opdrachtgever en de analyse kwamen de volgende niet functionele voort.

Niet functioneel eisen.

ID		Komt voor uit
NFE01	De Armory moet kunnen verbinden met een externe Wi-Fi Dongel voor internet.	USB armory heeft zelf geen WiFi functionaliteiten.
NFE02	De Armory is in staat via een ethernet kabel een internet verbinding op te zetten.	Gesprek met de opdrachtgever
NFE03	De USB VPN module wordt ontwikkeld op een USB Armory MK 2	Komt voort uit grenzen in het Plan van Aanpak
NFE04	De USB VPN module maakt gebruik van OpenVPN-nl client om zijn VPN verbinding op te zetten	Grenzen in het Plan van Aanpak
NFE05	Internet verbinding naar de computer vind plaats via Ethernet emulatie functionaliteit.	Komt voor uit de analyse. Anders kan de Armory geen verbinding aan de computer leveren.

17.5 8.5 functionele eisen

Na de gesprekken met de opdrachtgever zijn de volgende functionele en niet functionele eisen naar voren gekomen. Deze eisen zijn geprioriteerd op basis van de MosCoW methode

Functionele eisen

ID		prio	commentaar
FE01	De gebruiker hoeft enkel de USB VPN module in de computer te prikken waarna het programma zichzelf start. De gebruiker hoeft verder geen fysieke handelingen met de module uit te voeren.	M	
FE02	De USB VPN module probeert een secure VPN tunnel op te zetten.	M	

FE03	De computer hoeft geen directe verbinding te maken met het internet voor de USB VPN module om een VPN tunnel op te zetten	M	
FE04	De VPN module routeert het internet verkeer tussen de computer en het internet.	M	Anders kan die wel een VPN verbinding opzetten, maar dan gebeurd er niks mee
FE05	De computer kan niet direct bij de VPN module komen, configuratie moet op een indirecte manier plaatsvinden.	S	Bijvoorbeeld via bluetooth, of via een externe SSH sessie. Hiervoor iets van een GUI opzetten.
FE06	Wi-Fi hotspot selectie/verbinding vind plaats op de VPN module, de gebruiker mag wel indirect deze selecteren.	S	
FE07	De VPN module moet een vpn tunnel op kunnen zetten met een Technolution PrimeLink	C	
FE08	Voor de key generation maakt openVPN-nl gebruik van de Externe Cryptografische Chip.	C	

13. Sprint planning

In dit hoofdstuk is er een planning opgesteld welke eisen worden uitgevoerd per sprint. Deze planning kan aangepast worden aan de hand van de resultaten van een sprint. In de tabel wordt deze weergeven

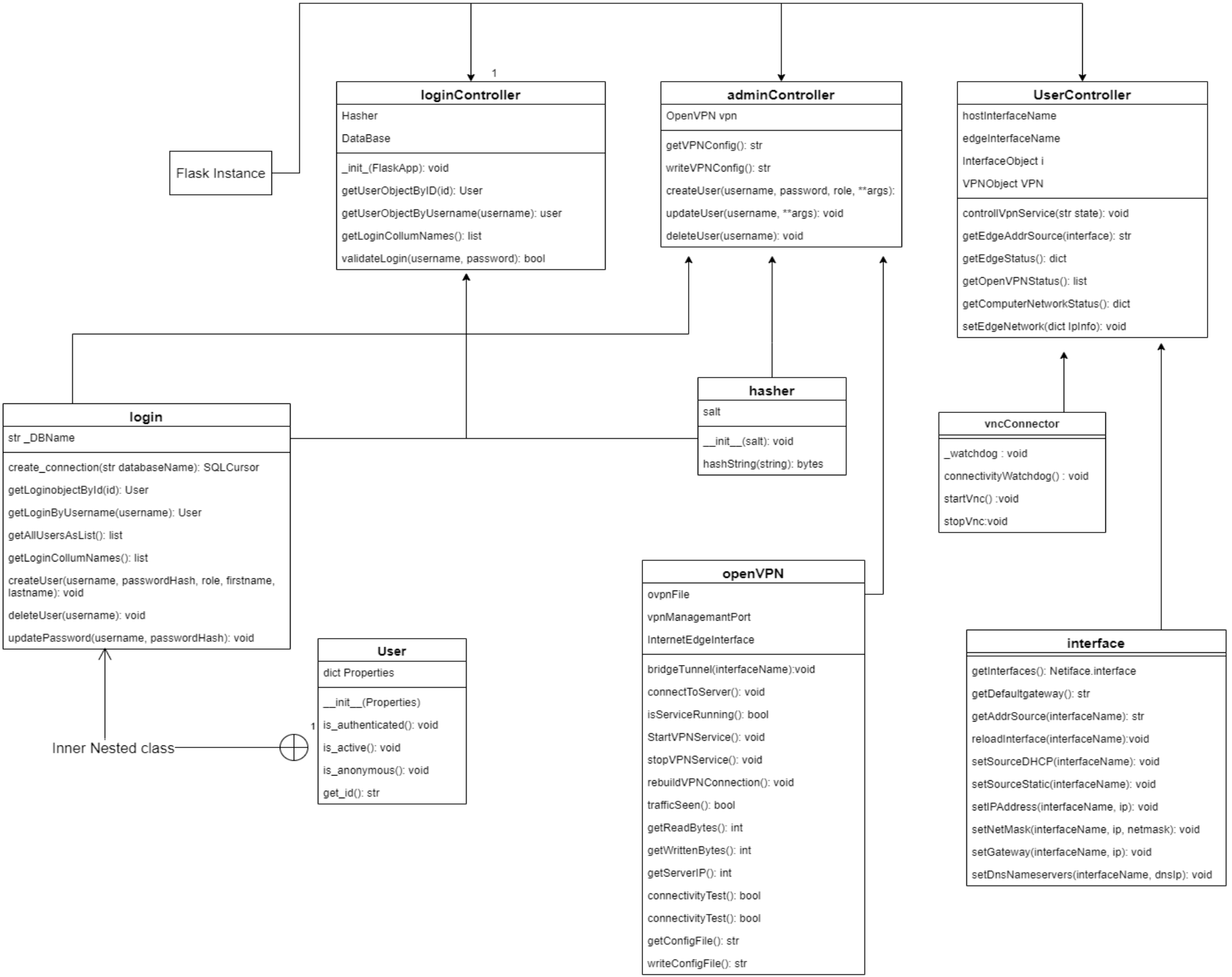
Sprint 1	Eis.	commentaar
Sprint 2	FE01	Deze eis is vrij globaal beschreven, maar deze eis is behaald als de Armory opgezet is en een programma automatisch kan starten na inpluggen.
Sprint 3	FE02/FE03	Opzetten van openVPN-nl server client
Sprint 4	FE04	Plugin voor de routering tussen de Armory Ethernet emulator en de OpenVPN tunnel maken.
Sprint 5	FE05	Maken van GUI of andere oplossing voor configuratie van Armory
Sprint 6	FE06	Wi-Fi selectie, moet plaats vinden op de Armory, maar op en of andere manier wel input nodig vanaf de computer
Sprint 7	FE07/FE08	Armory kunnen laten verbinden met een VPN PrimeLink van Technolution, key generation van openVPN gebruik laten maken van crypto chip in plaats van Kernel API.

Als de sprints eerder afgerond worden dan gepland kan er in overleg met de opdrachtgever eerder begonnen worden aan de volgende sprint. Als blijkt dat aan het einde van het project er veel tijd overblijft dan wordt deze tijd gebruikt voor de ontwikkeling van extra security features.

Bijlage E: klassendiagram

Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie bij Technolution BV. Gouda

Bart de Langen
15077071
HBO-ICT Network & System Engineering
De Haagse Hogeschool Delft
Versie 0.3
26 maar 2021


Flask Instance
1
loginController
Hasher
DataBase
init(FlaskApp): void
getUserObjectByID(id): User
getUserObjectByUsername(username): user
getLoginCollumNames(): list
validateLogin(username, password): bool
adminController
OpenVPN vpn
getVPNConfig(): str
writeVPNConfig(): str
createUser(username, password, role, **args):
updateUser(username, **args): void
deleteUser(username): void
UserController
hostInterfaceName
edgeInterfaceName
InterfaceObject i
VPNObject VPN
controllVpnService(str state): void
getEdgeAddrSource(interface): str
getEdgeStatus(): dict
getOpenVPNStatus(): list
getComputerNetworkStatus(): dict
setEdgeNetwork(dict IpInfo): void
login
str _DBName
create_connection(str databaseName): SQLCursor
getLoginobjectById(id): User
getLoginByUsername(username): User
getAllUsersAsList(): list
getLoginCollumNames(): list
createUser(username, passwordHash, role, firstname, lastname): void
deleteUser(username): void
updatePassword(username, passwordHash): void
hasher
salt
__init__(salt): void
hashString(string): bytes
vncConnector
_watchdog : void
connectivityWatchdog() : void
startVnc() :void
stopVnc:void
User
dict Properties
__init__(Properties)
is_authenticated(): void
is_active(): void
is_anonymous(): void
get_id(): str
1
Inner Nested class
openVPN
ovpnFile
vpnManagemantPort
InternetEdgeInterface
bridgeTunnel(interfaceName):void
connectToServer(): void
isServiceRunning(): bool
StartVPNService(): void
stopVPNService(): void
rebuildVPNConnection(): void
trafficSeen(): bool
getReadBytes(): int
getWrittenBytes(): int
getServerIP(): int
connectivityTest(): bool
connectivityTest(): bool
getConfigFile(): str
writeConfigFile(): str
interface
getInterfaces(): Netiface.interface
getDefaultgateway(): str
getAddrSource(interfaceName): str
reloadInterface(interfaceName):void
setSourceDHCP(interfaceName): void
setSourceStatic(interfaceName): void
setIPAddress(interfaceName, ip): void
setNetMask(interfaceName, ip, netmask): void
setGateway(interfaceName, ip): void
setDnsNameservers(interfaceName, dnsIp): void

Bijlage G: webpages views

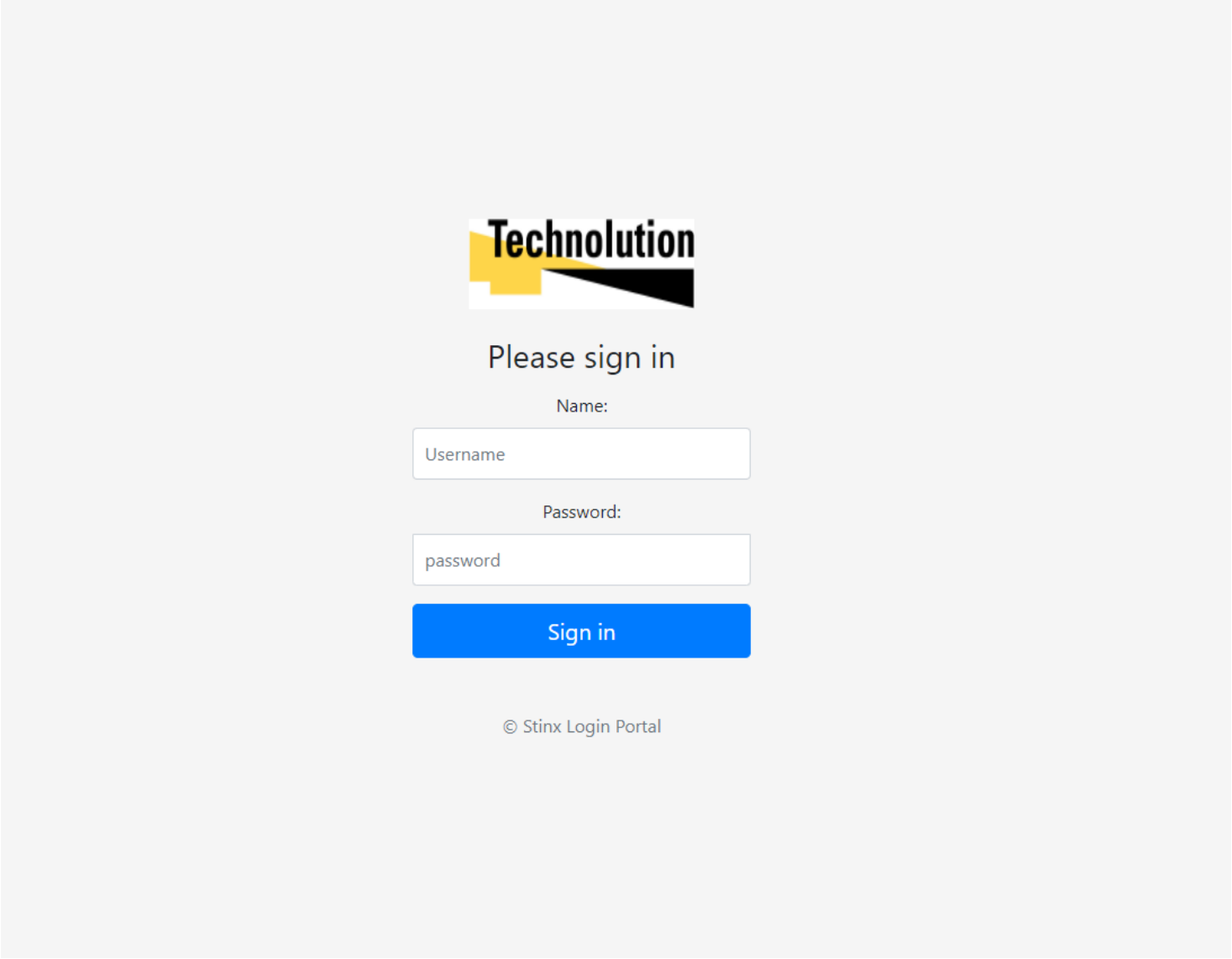
Ontwikkelen van een USB VPN module voor veilige end-to-end encryptie bij Technolution BV. Gouda

Bart de Langen
15077071
HBO-ICT Network & System Engineering
De Haagse Hogeschool Delft
Versie 0.3

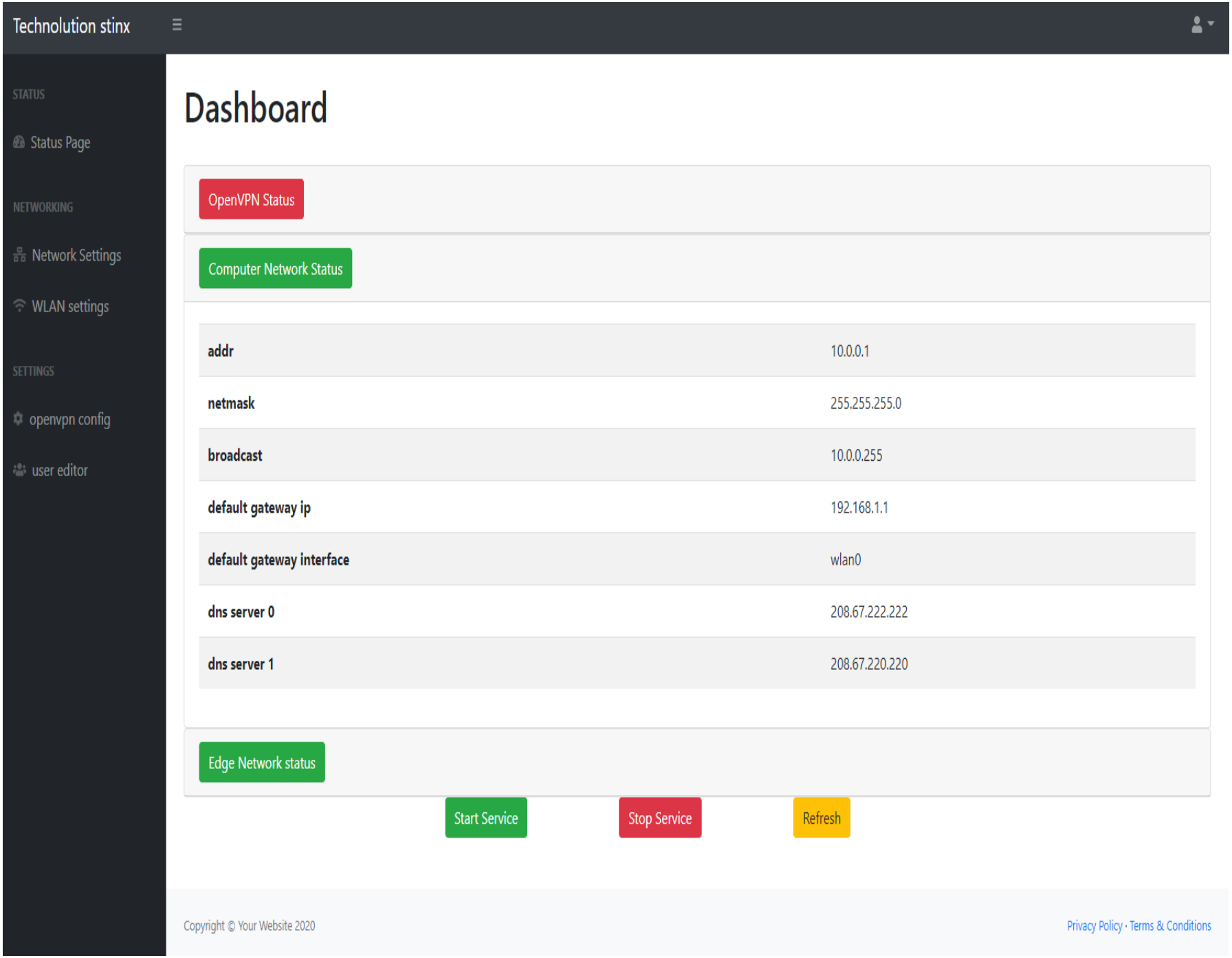
26 maar 2021

Technolution

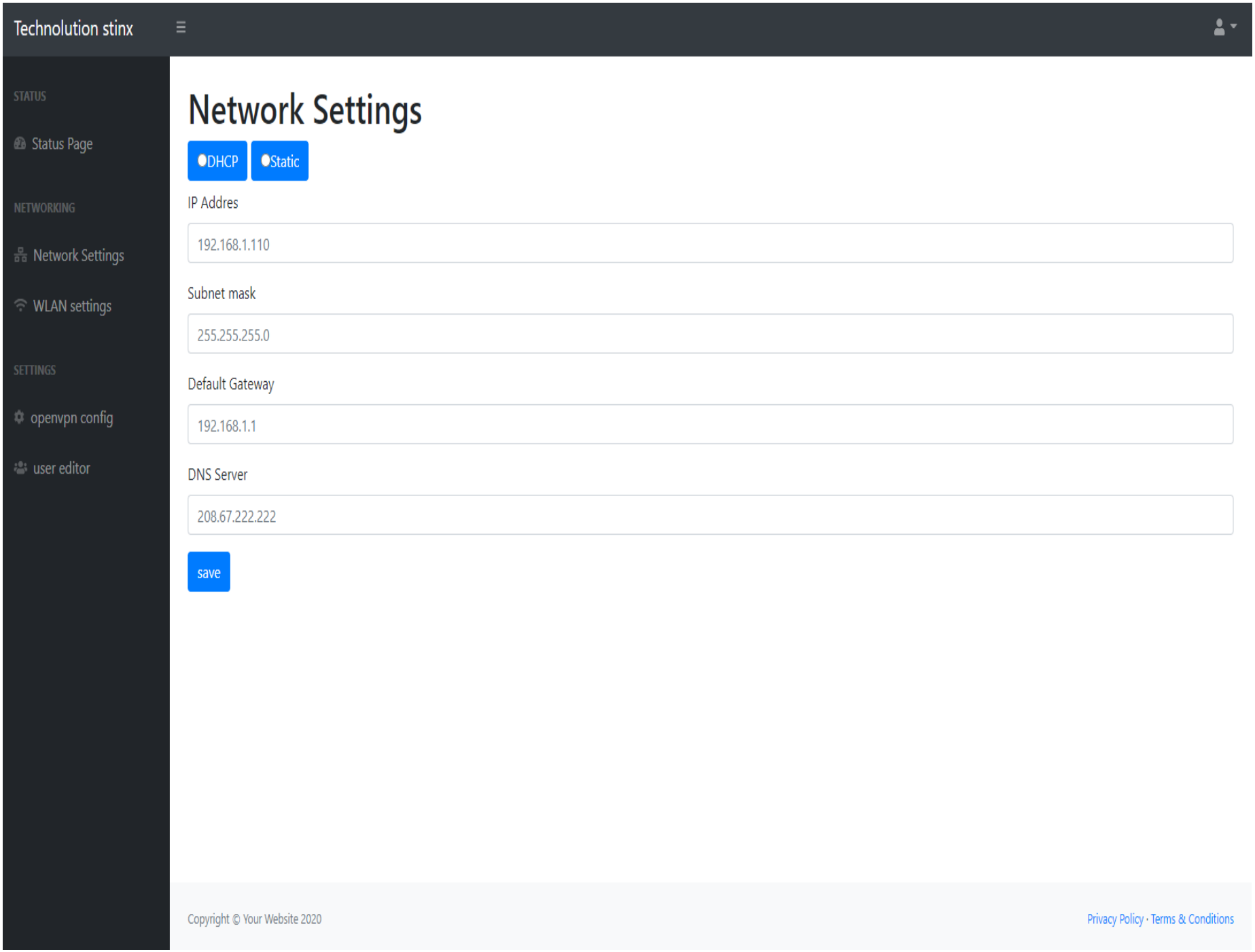
Loginpage view



Statuspagina view

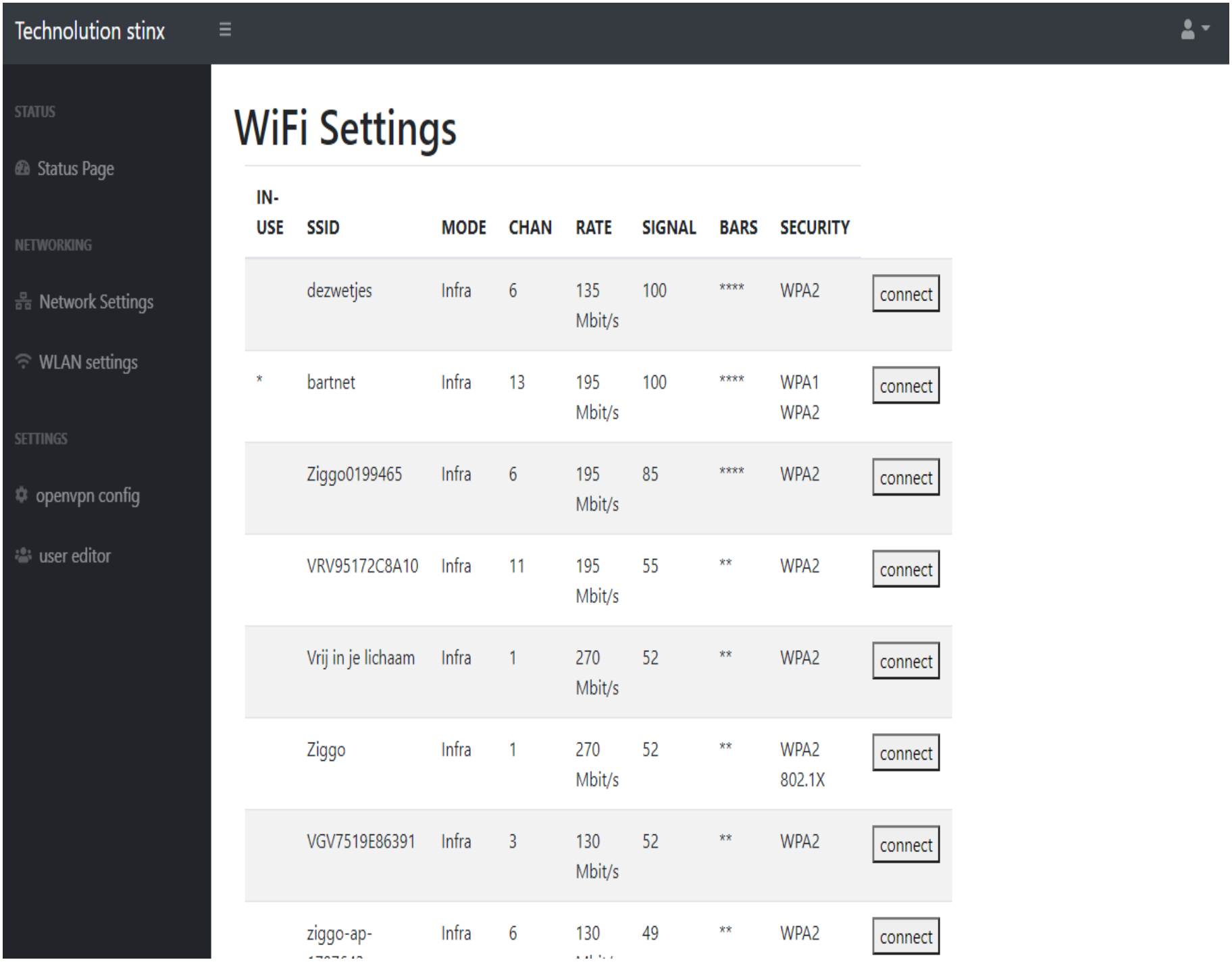


Technolution



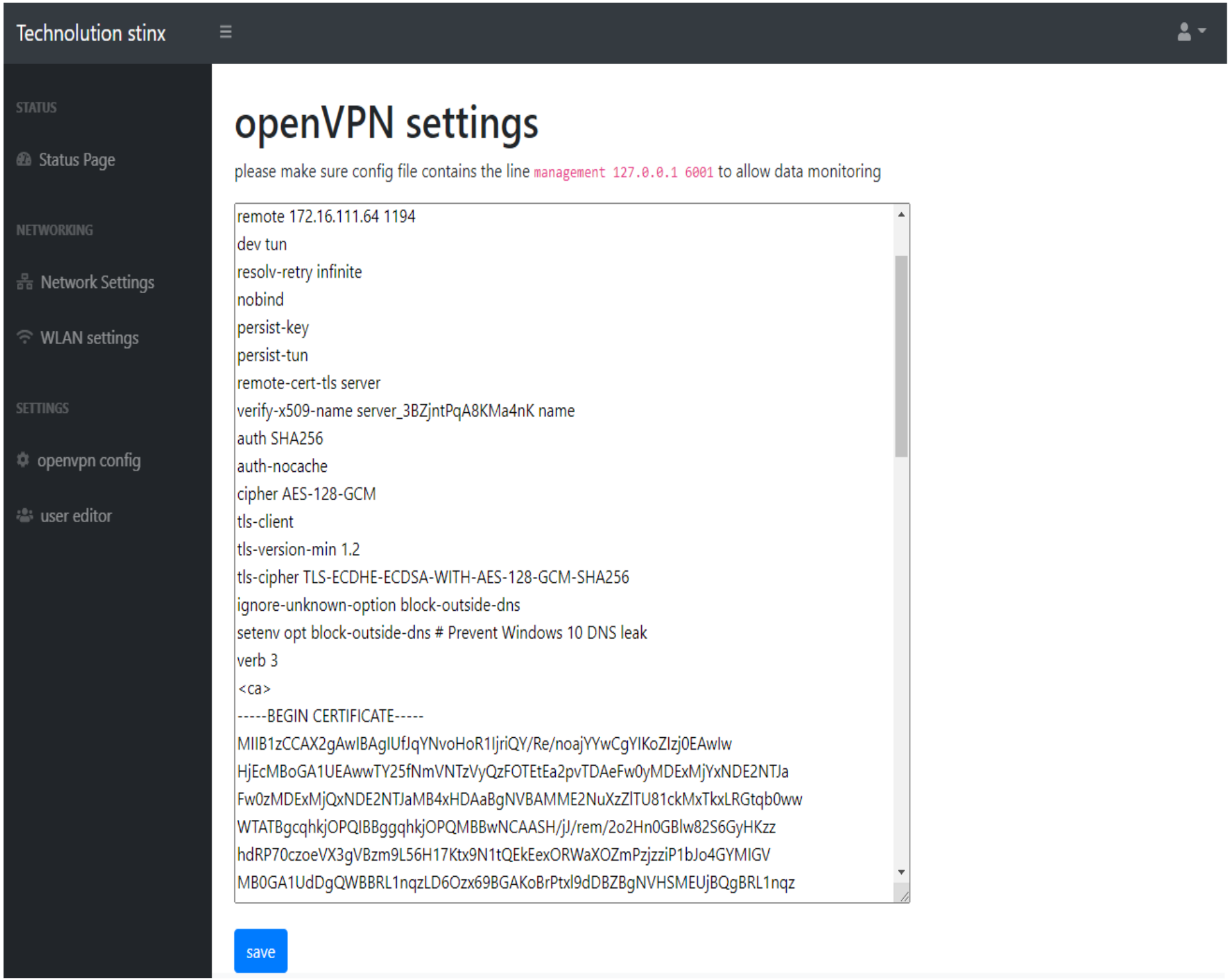
Network settings view

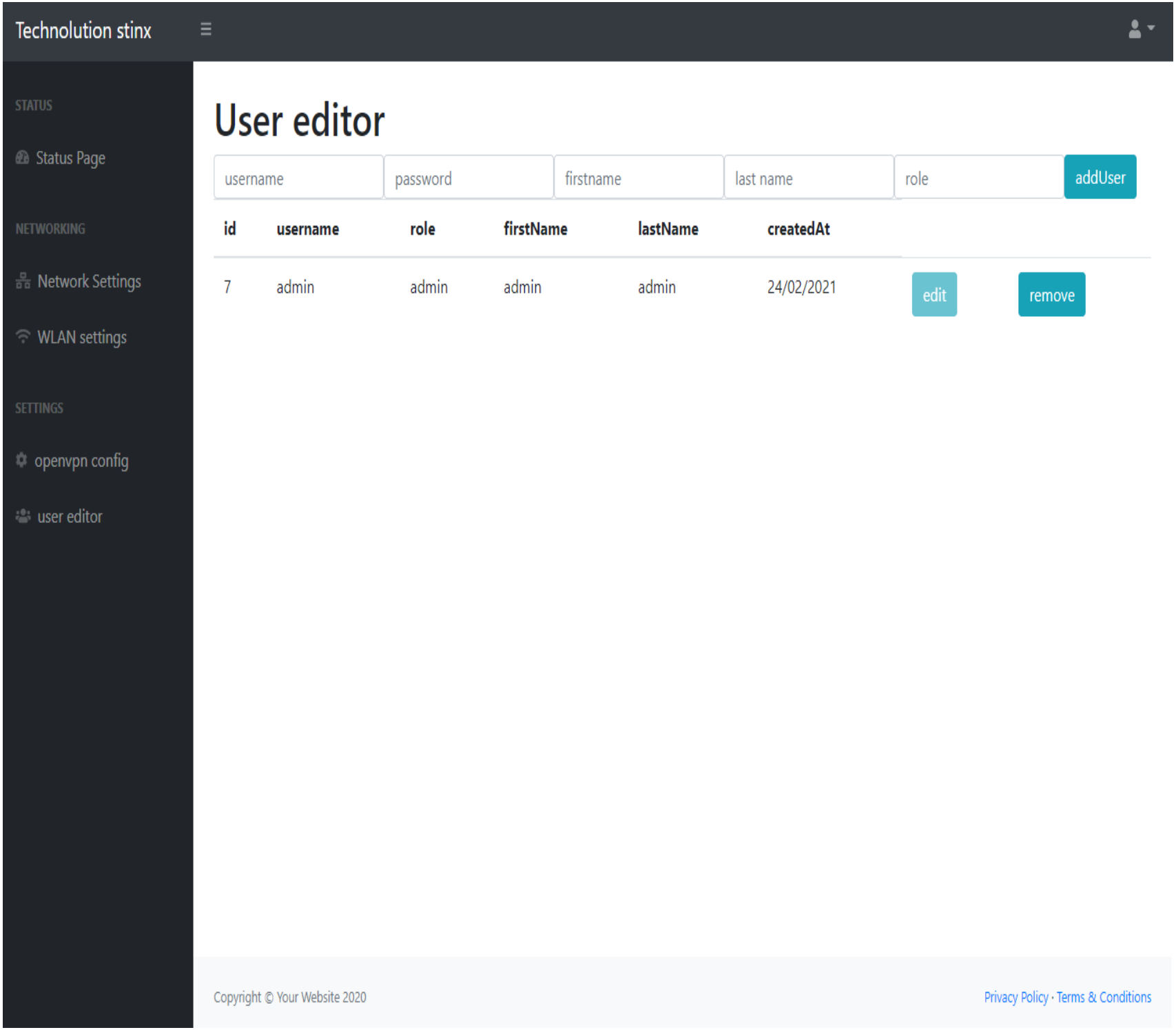
Technolution



wifi settings view

Technolution


Technolution stinx
STATUS
Status Page
NETWORKING
Network Settings
WLAN settings
SETTINGS
openvpn config
user editor
openVPN settings
please make sure config file contains the line management 127.0.0.1 6001 to allow data monitoring
remote 172.16.111.64 1194
dev tun
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
verify-x509-name server_3BZjntPqA8KMa4nK name
auth SHA256
auth-nocache
cipher AES-128-GCM
tls-client
tls-version-min 1.2
tls-cipher TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
ignore-unknown-option block-outside-dns
setenv opt block-outside-dns # Prevent Windows 10 DNS leak
verb 3
<ca>
-----BEGIN CERTIFICATE-----
MIIB1zCCAX2gAwIBAgIUfJqYNvoHoR1IjriQY/Re/noajYYwCgYIKoZIzj0EAwIw
HjEcMBoGA1UEAwwTY25fNmVNTzVyQzFOTEtEa2pvTDAeFw0yMDExMjYxNDE2NTJa
Fw0zMDExMjQxNDE2NTJaMB4xHDAaBgNVBAMME2NuXzZlTU81ckMxTkxLRGtqb0ww
WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAASH/jJ/rem/2o2Hn0GBlw82S6GyHKzz
hdRP70czoeVX3gVBzm9L56H17Ktx9N1tQEkEexORWaXOZmPzjzziP1bJo4GYMIGV
MB0GA1UdDgQWBBRL1nqzLD6Ozx69BGAKoBrPtxl9dDBZBgNVHSMEUjBQgBRL1nqz
save



user editor view