# Inleiding

Ik vroeg mij altijd al af hoe een programma werkte met een web server. Toen ik van het eindproject hoorde, dacht ik na een tijdje, waarom niet mijn eigen programma schrijven die werkt met een web server? Daaruit was ik verder gaan werken. Het einddoel wat ik in gedachten had was een programma met een mooie layout die communiceert met een web server die altijd draait en waarover ik de controle heb. Dit wilde ik in C++ voor elkaar krijgen, aangezien ik C++ al redelijk onder de knie heb en ik het wel een leuke uitdaging vond om het in C++ te doen en niet een andere taal, zoals Python. Ook vond ik het zonde van mijn tijd om een hele nieuwe taal te leren speciaal voor dit project. Ik wil een nieuwe programmeertaal leren als ik er ook echt blijvend nut in kan vinden, zodat ik ermee verder ga en het niet alleen voor dit project ga gebruiken en voor de rest nooit meer.

Tijdens het maken van dit programma, kreeg ik een bericht van een “internet-vriend” die had gehoord van mijn programma, zijn vraag was of ik er nog een DLL-injection methode van kon maken, wat het programma eigenlijk een soort van “loader” maakte, een programma die een dll in een programma injecteert om zo functies te onderscheppen en te veranderen. Dit programma is dus ook een beetje iets geworden om mijn tijd in te steken en mijn kennis van C++ en web servers te verbreden.

Om het kort samen te vatten wat het “ultieme” einddoel is:  
Een programma die een login systeem heeft en een bestand downloadt van een web server, alleen als de gebruiker is ingelogd met een online systeem (MySQL). Dit bestand (een DLL) injecteert hij dan automatisch na het downloaden en verwijdert het bestand meteen erna.

Vanwege de tijd, zal ik echter dingen moeten gaan schrappen of vereenvoudigen, zoals het login systeem en de layout van het programma.

# Vooronderzoek

## Bronnen

## <https://curl.haxx.se/libcurl/>

Hier staat de documentatie van een library gemaakt in C. Zoals ze het zelf beschrijven is het een client-side URL transfer library, wat voor mij handig zou kunnen zijn voor een online login verification.

<https://www.geeksforgeeks.org/socket-programming-cc/>

Hier staat uitgelegd hoe sockets werken en het idee erachter. Dit kan handig zijn als ik een bi-directional communicatie wil opstarten tussen programma en web server om zo de hele tijd data te blijven verzenden.

<https://www.w3schools.com/php/php_mysql_intro.asp>

Uitleg over hoe je een MySQL database moet integreren in Php, wat nuttig kan zijn voor mijn online login systeem, wil ik hem gaan maken.

<https://cloud.google.com>

De manier om een web server te hosten.

<https://msdn.microsoft.com/en-us/library/bb384843.aspx>

Hier staat uitgelegd hoe je een windows programma kan maken in c++

<https://github.com/ocornut/imgui>

De documentatie van een library voor de opmaak van een programma.

<https://en.wikipedia.org/wiki/DLL_injection>

Uitgelegd wat DLL-Injection is.

<https://www.codeproject.com/Articles/4610/Three-Ways-to-Inject-Your-Code-into-Another-Proces>

De manieren om een DLL in een programma te injecteren

## Werkafspraken en planning

Ik had niet echt werkafspraken en een planning, aangezien ik de vrijheid had om dit allemaal achter mij te laten, aangezien ik in mijn eentje was.

## Resources

Ik zal een server nodig hebben om mijn web sever vanaf te draaien. Dit is te doen d.m.v. USBWebServer, maar valt ook voor elkaar te regelen om een cloud service aan te schaffen (Amazon Web Service of Google Cloud bijvoorbeeld).

Om mijn C++ programma te schrijven zal ik een IDE en een compiler nodig hebben om mijn code te schrijven en compilen om het van code naar een .exe om te zetten. Dit zal Visual Studio 2017 worden, aangezien dit al is geïnstalleerd op mijn computer en ik dit heel erg fijn vind werken.

# Hoofd- en deelvragen

## Hoofdvraag

Hoofdvraag

**Hoe maak je een programma maken die een bestand van een web server injecteert in een process?**

## Deelvragen

Deelvraag 1: Hoe maak je een web server?

Je hebt een web server nodig om een programma te laten communiceren met een web server. Dit moet echter zelf gedaan worden, maar hoe?

Deelvraag 2: Hoe maak je een login systeem?

Om ervoor te zorgen dat niet iedereen in je programma kan komen heb je een login systeem nodig. Welke vormen zijn er en hoe is dit voor elkaar te krijgen?

Deelvraag 3: Hoe injecteer je een bestand in een process?

Om een bestand in een process te injecteren, moet je eerst weten hoe alles werkt.

Deelvraag 4: Hoe maak je een programma in Windows?

Om een programma te schrijven die iets doet, moet je wel eerst een programma met een layout (dus geen console) kunnen maken.

Deelvraag 5: Hoe beveilig je je programma en web server?

Om hackers te voorkomen, hoe beveilig je login systeem, programma en web server?

# Achtergrondinformatie

In dit hoofdstuk geef je duidelijk alle informatie weer die gebruikt wordt bij het beantwoorden van de deelvragen.

## Deelvraag 1: Hoe maak je een web server?

Je hebt 2 mogelijkheden, zelf een web server hosten vanaf je eigen computer, d.m.v. USBWebserver, een programma die we in de 4e hadden gebruikt om onze Informatica website te hosten vanaf onze computer. De andere manier is om het vanaf een echte server te laten hosten die de hele tijd draait. De 2e optie leek mij de beste, aangezien dit toch ietwat beter overkomt. Ik ging dus kijken naar een hosting service. Voor het (gratis) hosten van een project kwam ik uit bij AWS (Amazon Web Service) of Google Cloud. Voor de keuze tussen deze 2 werd een4 website gebruikt, die alle verschillen goed uitlegde tussen de twee.

Hieruit kwam dat Google Cloud de beste optie was. Na dit aanschaffen kwam ik erachter dat er veel meer zat achter een paar bestanden uploaden en dat je klaar bent.

Ik moest uiteindelijk de keuze maken om een App Engine of een Compute Engine te gebruiken om mijn web service te hosten. Ik snapte het verschil niet, totdat ik een5 pagina had gelezen waarin alle verschillen goed waren uitgelegd.

Hieruit kwam dat de App Engine beter voor mij was, aangezien dit automatisch het aantal “instances” bepaalt voor de “load” die je web service ondervindt. Bij een Compute engine moet je helemaal je eigen VM (Virtual Machine) configureren en opzetten, terwijl dit bij een App engine al een stuk makkelijker gaat.

Hoe krijg je vervolgens je bestanden naar de web server die je hebt gecreërd? Dit wordt een “app deployen” genoemd. Je uploadt je app naar de App Engine waarvan het gaat draaien. Gelukkig biedt google een uitleg6 over hoe je dit voor elkaar moet krijgen.

Na het creëren van een app.yaml kan je je bestanden uploaden naar de App Engine waar je draait. Nu heb je een web service gemaakt!

De volgende vraag was, hoe krijg je het voor elkaar dat mensen dit kunnen bereiken? Elke web service heeft zijn eigen “IP” adress, een aantal cijfers waar je eigenlijk naartoe gaat wanneer je een website intypt. Wanneer je een website intypt, bijvoorbeeld [www.google.com](http://www.google.com) die je doorstuurt naar een “IP”, waar je uiteindelijk een verbinding mee maakt en je de bestanden doorstuurt die je browser interpreteert naar een “echte” website. Dit wilde ik ook voor elkaar krijgen, maar hoe? Dit werkt met een domeinnaam. Ik had vervolgens de domeinnaam commendme.xyz opgekocht. Na wat instellingen veranderen werkt vervolgens alles. Dit leidde tot een web service die te bereiken is door commendme.xyz in te voeren in je browser, die je vervolgens naar mijn web service leidt.

## Deelvraag 2: Hoe maak je een login systeem?

Je hebt een offline en een online login systeem. Een offline login systeem is een login systeem waar de inloggegevens op de computer van de eigenaar staan. Meestal zijn ze in de code van het programma “vastgezet”, maar het kan ook dat het programma de login gegevens vergelijkt met een (geheim) bestand op de computer van de gebruiker.

Een “online” login systeem is een login systeem waar de correcte inloggegevens zich niet op de computer van de gebruiker bevinden maar ergens in de cloud. De gebruiker voert zijn inloggegevens in, die het programma opstuurt naar een web server. De web server controleert of de opgestuurde gegevens kloppen met de inloggegevens in de database van de web server. Vervolgens stuurt de web server een bericht terug naar het programma die de gebruiker toegang geeft of weigert, afhankelijk van de inloggegevens. Dit is voor elkaar te krijgen d.m.v. een HTTP request die je programma uitzendt naar een web server en die een response stuurt, afhankelijk van de login info. Dit valt ook voor elkaar te krijgen door een socket te creëren tussen programma en web server8.

## Deelvraag 3: Hoe injecteer je een DLL in een process?

De naam DLL (Dynamic Link Library) spreekt al redelijk voor zich. Het is een library die een link legt met een applicatie en “dynamic” is, omdat het door meerdere applicaties gebruikt kan worden en geladen wanneer nodig. Een website7 had heel mooi uitgelegd wat voor verschillend methodes er waren.

Om kort uit te leggen wat hier staat: Je moet plaats maken in het programma om je DLL in te laden. Wanneer de DLL is geladen, zorg je ervoor dat het programma waar je DLL in is ge-injecteerd de begincode van je DLL laadt. Vervolgens creëert je DLL een “thread” in je programma waar je code vanaf gaat draaien en voilà.

## Deelvraag 4: Hoe maak je een programma in Windows?

Elk Windows programma heeft een “WinMain” functie nodig, een functie die de start van je programma is.

Vervolgens heeft je programma een functie nodig om alle Windows “events” om te zetten tot iets waar je programma op kan reageren (Een voorbeeld van een Windows event is bijvoorbeeld een rechtermuisknopklik of een ingedrukte knop op je toetsenbord).

Daarna moet je een “structure” genaamd WNDCLASSEX vullen met informatie over je window, zoals welke cursor je programma heeft, achtergrondkleur etc. Vervolgens moet je dit registreren bij Windows, zodat Windows de informatie heeft over je window en hoe Windows hiermee moet communiceren.

Hierna moet je je window creëren. Er gaat dus heel veel vooraf aan het echt creëren van je window om het goed met Windows om te laten gaan. Dit doe je met een functie genaamd CreateWindow. Deze window moet je ook nog eens zichtbaar maken voor Windows hierna.

Nu heb je je window gecreëerd en kan je beginnen met erin zetten wat je wilt, zoals knoppen, foto’s of wat dan ook.

## Deelvraag 5: Hoe beveilig je je programma en web server?

Aangezien ik mijn web server draai op de servers van google hoef ik mij niet druk te maken over de beveiliging van de web server. Waar ik mij echter wel druk over moet maken is hoe ik ervoor zorg dat mijn login systeem werkt en mijn programma. Om ervoor te zorgen dat mijn programma moeilijker wordt om te reverse engineeren kan je je code laten beveiligen met een programma genaamd VMProtect3.

Om het inloggen te beveiligen2, je kan je username naar de server sturen, de server stuurt dan een code terug. Deze code versleutel je dan met de “hash” van je wachtwoord en deze versleutelde code stuur je terug naar de server. De server checkt deze code, die hij zelf op dezelfde manier versleutelt met de hash van het wachtwoord dat bij de ingestuurde username hoort en kijkt of ze overeen komen. Daarna stuurt hij iets terug waarmee duidelijk is dat je bent ingelogd en de web server geeft dan toegang aan de gebruiker tot de bestanden op de web server.

**Overige info:**

De DLL die ik heb gemaakt is een hele simpele DLL, die een thread maakt met een geluid van 250 HZ, die 250 ms duurt met een pauze van 2 seconden ertussen.

# Conclusie

## Deelvragen

Deelvraag 1: Hoe maak je een web server?

Je zet een App Engine op met Google Cloud en upload je bestanden naar deze App Engine met een bestand genaamd app.yaml zodat de server weet wat hij precies moet draaien. Hierna zet je een domeinnaam op die gebruikers doorstuurt naar de web server.

Deelvraag 2: Hoe maak je een login systeem?

Je vraagt van de gebruiker een username en wachtwoord en stuurt deze op naar een server die nagaat of deze kloppen of controleert deze met informatie opgeslagen op de computer van de gebruiker, in het programma zelf gecodeerd of erbuiten.

Deelvraag 3: Hoe injecteer je een DLL in een process?

Je maakt plaats voor de DLL in het process, vervolgens laad je de DLL in je programma. Hierna zorg je ervoor dat het process je begincode van je DLL uitvoert, zodat die een thread creëert, waarin je al je code kan uitvoeren.

Deelvraag 4: Hoe maak je een programma in Windows?

Je maakt een WinMain functie, de beginfunctie van je Window, en een functie om alle Windows events te verwerken. Hierna vul je een WNDCLASSEX class in met informatie van je programma en registreert deze bij Windows. Hierna creëer je een window die je vervolgens zichtbaar maakt voor Windows.

## Hoofdvraag

**Hoe maak je een programma maken die een bestand van een web server injecteert in een process?**

Je creëert een Windows programma met Visual Studio 2017, die een bestand kan injecteren in een ander process en een bestand kan downloaden van een web server.

De web server creëer je door een App Engine op te starten bij Google Cloud en daar je bestanden naartoe up te loaden, waar je programma toegang tot krijgt.

# 

# Beschrijving product

## Beginontwerp

**Omschrijving**

Een programma die communiceert met een web server om een DLL te verkijgen van de web server en deze te injecteren in een programma.

**Schetsen**

4 knoppen:

* Download – Downloadt het bestand van een web server
* Remove – Verwijdert automatisch het bestand
* Inject – Injecteert het bestand
* Auto-Inject – Downloadt het bestand en injecteert het automatisch

Bij de Inject en Auto-Inject knop ga je naar een ander menu, waar je de naam van je process in kan vullen, terug kan gaan naar het main menu en een knop wanneer je wilt injecteren.

Wanneer je het programma opstart krijg je eerst een login scherm te zien met 2 “invulbalkjes” en een knop om in te loggen.

**Verhaal & Ontwerp**

Het ontwerp moet een blauwe/paarse kleur krijgen, maar dat is ook het enige wat ik wil qua ontwerp.

**Lijst van Eisen**

* Moet automatisch kunnen communiceren met een web server
* Moet een DLL automatisch kunnen injecteren na downloaden ervan.
* Moet de DLL kunnen injecteren in een process naar keuze.

**Optionele eisen**

* Een login systeem die werkt met een online database en geen gegevens op de computer van de gebruiker zelf
* Nette layout
* Beveiliging van het bestand en login process.

## Proces

Aan het begin van het dit project begon ik met het creëren van de layout van mijn programma en Google Cloud opzetten. Ik wilde een App Engine draaiende hebben voordat ik naar Griekenland ging, ook al had ik nog niks om te uploaden qua nut voor mijn informatica project. Dit ging allemaal soepel en zonder problemen. Na de Griekenlandreis wilde ik de rest gaan doen, zoals communicatie leggen tussen het programma en de web server. Na de Griekenlandreis ging alles echter een stuk minder soepel. Ik kreeg en balngrijke libnrary niet aan het werken, die eigenlijk essentieel was om een communicatie te leggen tussen programma en web server, waardoor ik dit plan uit moest stellen, jammer genoeg tot na de deadline van dit project. Uiteindelijk is het wel gelukt via een omweg om een bestand te downloaden, maar jammer genoeg niet een online loginsysteem te maken.

## Beschrijving eindproduct

Laat hier duidelijk en uitgebreid zien hoe het eindproduct er uit ziet en wat het kan. Hier kunnen o.a. screenshots bij gebruikt worden. Dit onderdeel moet, zonder dat er naar het eindproduct gekeken wordt, een goed idee geven van het eindproduct.

# C:\Users\Sebastian\Documents\5db11925b6cb3db62998c85cd7b434c0.pngC:\Users\Sebastian\Documents\08a17cc009126f6e2435ec510801dbd6.png

# Leerpunten

Probleem: Het programma injecteert alleen maar in 32-bit processen.  
Oplossing: Compile de DLL en programma in 64-bit i.p.v. 32-bit. Als dit niet gebeurt, kan je tijdens het injecteren geen “thread” maken om vervolgens de begincode van je DLL uit te voeren.

Probleem: Een plek vinden om je gedownloade DLL tijdelijk te bewaren  
Oplossing: Krijg de path name van je .exe bestand, verwijder hier de naam van je .exe van en je krijgt de directory waar je het programma vandaan draait, waar vervolgens de DLL kan worden bewaard.

# 

# Bijlagen

## Bronvermelding

Een duidelijk genummerde lijst van bronnen1 die je gebruikt hebt. Let hierbij op zowel de aanduiding in de tekst in superscript als de manier van verwijzen in de lijst van bronnen (zie voorbeeld).

1. Bronvermelding: <http://nl.wikipedia.org/wiki/Bronvermelding>
2. Loginbeveiliging: <https://stackoverflow.com/questions/11580944/client-to-server-authentication-in-c-using-sockets>
3. VMProtect: <http://vmpsoft.com/>
4. Verschil Google Cloud en AWS: <https://medium.com/@robaboukhalil/a-tale-of-two-clouds-amazon-vs-google-4f2520516a38>
5. Verschil App Engine en Compute Engine: <https://stackoverflow.com/questions/22697049/what-is-the-difference-between-google-app-engine-and-google-compute-engine>
6. Deploy App Google Cloud: <https://cloud.google.com/appengine/docs/standard/python/getting-started/deploying-the-application>
7. DLL injection methoden: <https://www.codeproject.com/Articles/4610/Three-Ways-to-Inject-Your-Code-into-Another-Proces>
8. Uitleg Socket en Http: <https://stackoverflow.com/questions/15108139/difference-between-socket-programming-and-http-programming>

## Logboek

Sebastian Speekenbrink

|  |  |  |
| --- | --- | --- |
| **Taak (wat heb je gedaan?)** | **Datum** | **Duur** |
| Programma en App Engine maken | Voor 3 mei | 7 uur |
| Code schrijven | 19 en 20 juni | 6 uur |
| Verslag schrijven |  | 3 uur |
|  | **Totaal** | 16 uur |

## Overige bijlagen

Voeg hier overige bijlagen toe.

## 