Probleemstelling & Analyse

HBO-ICT: Systems & Middleware Programming

**Student 1**

Naam: Twan Terstappen  
Studentnummer: 500893880

Klas: IC102

Datum: **9-Jun-23**

Versie: 2.0

Inhoudsopgave

[1. Probleemstelling 3](#_Toc132794599)

[2. Deelvragen 4](#_Toc132794600)

[3. Analyse 5](#_Toc132794601)

[4. Programma-structuur en testing 6](#_Toc132794602)

# Probleemstelling

In Cyber Security is het van belang om op de hoogte te blijven van wat er in jouw netwerk gebeurt. Door dit te weten kan je mogelijke zwakheden of beveiligingslekken verhelpen en/of voorkomen. Een netwerk kan al snel miljoenen pakketten per dag bevatten. Het is daarom moeilijk om zelf het netwerk te controleren, daarom is het doel om zelf een programma te maken dat eventuele aanvallen of beveiligingslekken kan detecteren en zo alarm kan slaan.

In deze probleemstelling & analyse wordt er een dataset onderzocht waarin mogelijke aanvallen hebben plaatsgevonden. Er wordt onderzocht of er een TCP Hijacking en syn flood heeft plaatsgevonden. TCP Hijacking is een type aanval waarin de aanvaller een TCP-verbinding overneemt en zo een sessie met bijvoorbeeld een server heeft in plaats van een client. Het programma zal de dataset analyseren op kenmerken van een TCP Hijacking. Een aantal kenmerken van een TCP Hijacking zijn bijvoorbeeld: meerdere MAC-adressen in een TCP-verbinding, TTL (Time To Live) heeft een ander patroon, verdachte pakketten die van een onbekende bron komen, DOS of DDOS aanval op een host die een TCP-verbinding heeft.

Een SYN flood aanval is een DOS (denial-of-service) of ddos (distributed denial-of-service). Hierin worden meerdere SYN pakketten gestuurd naar de server toe om zo de server te overwelmen. Een SYN pakket is een pakket waarbij de client aangeeft dat het een TCP verbinding wil maken met de server. De server ontvangt dit pakket en stuurt een SYN ACK, hij accepteert het verzoek. Tijdens deze SYN ACK reserveert de server resources voor de eventuele TCP verbinding met de client. Normaliter is het zo dat de client een ACK terugstuurt, een acceptatie. Dit zorgt ervoor dat er een zogeheten 3-way handshake is plaatsgevonden en de TCP verbinding openstaat. Bij een SYN flood aanval zal de client deze laatste ACK niet sturen, hierdoor zal de server wachten op de ACK en heeft het onnodig resources vrijgemaakt voor een verbinding die niet gaat plaatsvinden. Bij zo’n SYN flood aanval stuurt de aanvaller zoveel mogelijk SYN verzoeken. Hierdoor wordt de server overbelast en zal deze buiten werking treden. Kenmerken van een SYN flood aanval: veel SYN verzoeken van één IP en veel SYN verzoeken in een korte tijdspan

De dataset bevat zo’n 3000 pakketten tussen een webserver en clients, waarbij elk pakket informatie bevat over de communicatie op de vijf lagen van het TCP/IP protocol stack. De dataset komt van opgevangen netwerkverkeer dat naar de webserver gaat van een universiteit.

In probleemstelling worden er specifieke vragen opgesteld die nodig zijn om de analyse uit te voeren. Door deze vragen te beantwoorden kan er gekeken worden of er kenmerken zijn van een TCP Hijacking en een SYN flood aanval. Het programma zal op deze vragen ontwikkeld worden en zal na een analyse van de dataset met een rapportage komen en inzicht kunnen geven of er kenmerken zijn van een TCP Hijacking en SYN flood aanval.

# Deelvragen

Met de deelvragen wil ik de dataset analyseren op kenmerken van een TCP Hijacking en SYN flood aanval. Door deze deelvragen kan ik antwoordt krijgen en zo kunnen constateren of er een TCP Hijacking en/of SYN flood heeft plaatsgevonden tussen de server en client. De deelvragen gaan als volgt:

## Hoeveel verschillende TCP hebben een bepaalde status

Door te weten hoeveel verschillende TCP een bepaalde status hebben kan er in kaart gebracht worden of er TCP verbinden zijn die een gevaalde handshake, succesvol zijn afgesloten of die niet zijn afgesloten. Met deze data kan er gekeken worden of er een mogelijke SYN flood aanval is geweest

## Hoeveel SYN verzoeken hebben elk IP adres

Door het weten van de SYN verzoeken voor elk IP adres kan er gekeken worden of een IP adres veel syn verzoeken heeft gestuurd. Dit kan een correlatie zijn met een SYN flood aanval. Bij een groter aantal syn verzoeken kan er geconstateerd worden dat er eventueel een SYN flood aanval is geweest van dat type IP adres.

## Welk TCP verbindingen hebben een TTL buiten de standaard afwijking van 2.5%

Door te weten of er een pakket TTL (Time To Live) buiten de standaard afwijking van 2.5 procent is kunnen we aannemen dat er een mogelijkheid is dat er een TCP Hijacking heeft plaatsgevonden. Doordat de TTL in de 2.5 procent ligt, kan het zijn dat de sessie is overgenomen en het pakket langer duurt of korter duurt voordat het bij de server aankomt. Doordat de pakket een andere route aflegt is de TTL anders en kan er een mogelijkheid zijn van een TCP Hijacking.

# Analyse

## Hoeveel verschillende TCP hebben een bepaalde status

Bij deze deelvraag is de volgende data nodig:

* Alle TCP verbinding
* Status per TCP verbidning (gefaalde-handshake, niet goed afgesloten of afgesloten)
  + SYN pakket, client naar server
  + SYN ACK pakket, server naar client
  + ACK pakket, client naar server
  + FIN pakket, van client naar server of server naar client

Eerst worden alle TCP verbindingen in kaart gebracht. Hiervan wordt er gekeken of een TCP verbinding een succesvolle handshake heeft. Hierna wordt er gekeken of deze verbinding wordt afgesloten door te kijken naar de FIN pakket. Zo niet dan wordt de TCP verbinding op niet goed afgesloten gezet.

Hierbij wordt gekeken welke status een TCP verbinding heeft. Elke TCP verbinding wordt dan gecategoriseerd op status en wordt dan bij elkaar opgeteld. Hierdoor krijg je een overzicht in hoeveel TCP verbindingen een status hebben.

## Hoeveel SYN verzoeken hebben elk IP adres

Bij deze deelvraag is de volgende data nodig:

* Alle SYN verzoeken in de dataset
* IP die bij het SYN verzoek hoort

Er wordt gekeken naar de SYN verzoeken in de dataset. Bij dit SYN verzoek wordt het IP van de SYN verzoek opgeslagen. Elke keer als er een SYN verzoek is in de dataset wordt deze bij het IP opgeteld. Hierdoor krijg je een overzicht van de aantal SYN verzoeken van elk IP

## Welk TCP verbindingen hebben een TTL buiten de standaard afwijking van 2.5%

Bij deze deelvraag is de volgende data nodig:

* Alle TCP verbindingen waarbij de verbinding niet goed of goed is afgesloten
* TTL van elk pakket van de TCP verbinding

Hierin wordt de standaard afwijking berekend op basis van alle pakket hun TTL in een verbinding. Vervolgens wordt er een gekeken of er een pakket in de verbinding buiten de drempelwaarde van 2.5% ligt. Als dit zo is kan er geconstateerd worden dat er een mogelijk TCP Hijacking heeft plaatsgevonden in de verbinding.

# Programma-structuur en testing

Instance variabelen in de "TcpConnectionAnalyzer" class:

* file\_path: een string die het pad naar het datasetbestand opslaat.
* packages: een lijst die de pakketten uit het datasetbestand opslaat.
* connections: een lijst die de gevonden TCP-verbindingen opslaat.
* status\_count: een dictionary die de tellingen van verschillende statussen van de verbindingen opslaat.
* syn\_flood\_counter: een dictionary die de tellingen van SYN-verzoeken per IP opslaat.
* possible\_hijacking: een dictionary die mogelijke hijackings met afwijkende TTL-waarden opslaat.
* syn\_flood\_minimum: een integer die het minimum aantal SYN-verzoeken definieert voor een SYN-floodaanval.
* hijacking\_threshold\_percentage: een float die de drempelwaarde definieert voor de afwijking van de TTL-waarden voor mogelijke hijackings.
* syn\_flood\_warning\_color: een integer die het aantal SYN-verzoeken definieert waarbij de waarschuwingskleur rood wordt weergegeven.

Functies in de "TcpConnectionAnalyzer" class:

* load\_data(): laadt de gegevens uit het datasetbestand en retourneert deze als een lijst.
* extract\_data(): haalt de relevante gegevens uit het datasetbestand en slaat ze op in de "packages" lijst.
* find\_connection(): zoekt naar TCP-verbindingen in de "packages" lijst en roept de functie find\_ack\_fin aan.
* find\_ack\_fin(start\_index, connection): zoekt naar syn\_ack, ack en fin pakketten en slaat de datapakketten en registreert alle pakketten. Vervolgens slaat het de connection op in de lijst connections.
* count\_connection\_status(): telt het aantal verbindingen per status en slaat ze op in de "status\_count" dictionary.
* display\_count\_connection\_status(): geeft het totale aantal verbindingen en de tellingen per status weer.
* syn\_flood(): telt het aantal SYN-verzoeken per IP en slaat ze op in de "syn\_flood\_counter" dictionary.
* display\_syn\_flood(): geeft de IP-adressen weer met het aantal SYN-verzoeken, waarbij de waarschuwingskleur rood wordt gebruikt voor aantallen boven de opgegeven drempelwaarde.
* tcp\_hijacking(): zoekt naar mogelijke hijackings op basis van afwijkende TTL met een drempelwaarde van hijacking\_threshold\_percentage en slaat ze op in de "possible\_hijacking" dictionary.
* display\_tcp\_hijacking(): geeft de index van de verbindingen weer waar mogelijke hijackings zijn gedetecteerd, samen met de afwijkende TTL-waarde en de afwijkingspercentage.
* test\_criteria: Elke functie heeft zijn eigen testcriteria, maar over het algemeen zou het programma correct moeten functioneren als het de juiste resultaten produceert volgens de specificaties van elke methode en de vereisten van de argumenten.

Functie main:

main(): de hoofdfunctie van het programma die de command line interface afhandelt en de bovengenoemde methoden aanroept op basis van de opgegeven argumenten.

Command line interface:

* Het programma wordt uitgevoerd door het volgende: “python tcp\_analyzer.py ../dataset.json” gevolgd door optionele argumenten.
* Vereiste argumenten:
  + file\_name: de naam van het JSON-bestand met de dataset.
* Optionele argumenten:
  + -C, --connections: geeft de verbinding statussen in de dataset weer.
  + -S, --syn-flood: geeft alle SYN verzoeken per IP weer
  + -T, --tcp-hijacking: geeft mogelijke TCP-hijackings weer.
  + -A, --all: voert alle analyses uit (verbinding statussen, SYN-flooding, TCP-hijacking).

Testing:

* def test\_load\_data\_negative(file\_path, expected\_exception): negative test voor load\_data. Verwachte error: FileNotFoundError en json.JSONDecodeError
* def test\_extract\_data(): positieve test voor extract data. Verwachte output type is list
* def test\_count\_connection\_status\_positive(): testen van functie count\_connection\_status. Verwachte output type dictionary
* def test\_display\_count\_connection\_status\_positive(capfd): testen van functie display\_count\_connection\_status. Verwachte output:
  + Total connections: 221\n"
    - closed: 84\n"
    - no-fin: 85\n"
    - failed-handshake: 52\n"