MASTER'S THESIS

Botnetsimulatie in een gedistribueerd virtueel computer security lab

Jonkman, Jeroen

Award date:
2016

Afstudeeropdracht

Botnetsimulatie in een gedistribueerd virtueel computer security lab

J. H. A. Jonkman

Student: 835547840
Datum: 14 juli 2016

Open Universiteit
www.ou.nl

AFSTUDEEROPDRACHT

BOTNET SIMULATION IN A DISTRIBUTED VIRTUAL COMPUTER SECURITY LAB

J. H. A. Jonkman

Masteropleiding Software Engineering

Open Universiteit, faculteit Management, Science & Technology

Student nummer:	835547840	
Cursus code:	T75317	
Datum:	14 juli 2016	
Afstudeercommissie:	Dr. ir. H. P. E. Vranken (Voorzitter, eerste begeleider),	Open Universiteit
	Dr. ir. A. J. F. Kok (Secretaris, tweede begeleider),	Open Universiteit

Open Universiteit
www.ou.nl

DANKWOORD

Aan het eind gekomen van mijn afstudeeropdracht voor de master Software Engineering aan de Open Universiteit sluit ik een periode van twee decennia deeltijd studeren af en is het even tijd om stil te staan en even achterom te kijken. Zonder de steun van mijn ouders, vrouw en kinderen was dit niet mogelijk geweest. Dank voor jullie steun, liefde en geduld, zonder jullie was dit zeker niet gelukt. Het resultaat van deze onvoorwaardelijke steun ligt voor u, het verslag van de afstudeeropdracht 'Botnetsimulatie in een gedistribueerd virtueel computer security lab'.

Na een intensieve periode van achttien maanden is het dan zover dat ook de laatste fase kan worden afgesloten waar ik veel heb geleerd, op wetenschappelijk gebied, maar ook op persoonlijk vlak. Ik heb niet kunnen bedenken dat je met een leerstoornis als dyslexie nog zo ver zou kunnen komen.

Ik wil even in het bijzonder stil staan bij mijn twee studie begeleiders, Harald Vranken en Arjan Kok en hen bedanken voor de fijne samenwerking en de kans die ik heb gekregen om bij hen te mogen afstuderen.

Daarnaast wil ik speciaal de medewerkers van het studiecentrum in Emmen bedanken voor hun luisterend oor, inzet en positieve sturing zonder hen was ik zeker niet zover gekomen.

Lieve allemaal, heel erg bedankt!

Jeroen Jonkman

Emmen, 14 juli 2016.

INHOUDSOPGAVE

LIJST VAN FIGUREN

LIJST VAN TABELLEN

LISTINGS

SAMENVATTING

Sinds de geboorte in 1983 van het Open Internet, kunnen computersystemen over de gehele wereld vrijelijk met elkaar communiceren. Door het open karakter van het Internet is beveiliging van deze computersystemen een steeds grotere rol gaan spelen, mede omdat deze systemen ook konden worden misbruikt.

Zo rond het jaar 2000 ontstonden de eerste botnets, waarmee ook de grotere informatiesystemen aangevallen konden worden door gebruik te maken van met vele bots besmette computersystemen. Bots ofwel softwarerobots zijn ongewenste softwaretoepassingen die op afstand beheerd worden door een botherder, vaak zonder dat de eigenaar van het besmette computersysteem dit weet. De botherder kan deze bots opdrachten geven en daarmee schade toe brengen aan gebruikers van het Internet. Dit doet de botherder middels het C&C Systeem van het botnet.

Het simuleren van grote botnet systemen in een afgesloten computer laboratorium is een arbeidsintensief proces waar veel kosten mee gemoeid zijn. In dit onderzoek wordt een eerste verkenning gedaan naar de mogelijkheden om een virtueel computer laboratorium hiervoor te kunnen gebruiken. Binnen de 'Open Universiteit' (OU) wordt al enige tijd gebruik gemaakt van het 'Virtuele Computer Security Lab' (VCSL) waarmee studenten computernetwerken kunnen simuleren en allerlei experimenten kunnen uitvoeren. Inmiddels is er ook een gedistribueerde versie van het VCSL het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) beschikbaar.

De vraag die in dit onderzoek centraal staat: **zijn botnetsimulaties mogelijk in het DVCSL?**

Om deze vraag te kunnen beantwoorden is voor dit wetenschappelijk empirisch onderzoek een praktijkgericht onderzoeksmodel gebruikt. Na het bestuderen van recent wetenschappelijk onderzoek op de drie onderzoeksdeelgebieden, botnets, het DVCSL en netwerkvirtualisatie, is er een onderzoeksmodel gemaakt. Vervolgens is er een conceptueel model gemaakt waarin de relatie tussen botnets en het DVCSL is weergegeven met de rol die netwerkvirtualisatie hierin vervult. Op basis van de hoofd onderzoeksvraag en deze twee modellen, zijn er drie onderzoeksvragen geformuleerd.

Welke ontwerp-principes en ontwerp-condities kunnen worden vastgesteld voor de virtualisatie van botnets? Eerst zijn de kenmerken die van belang zijn voor de virtualisatie van botnets vastgesteld. Deze hebben betrekking op de architectuur, topologie en de levenscyclus van botnets. Door het beschrijven van deze attributen en kenmerken en vervolgens vast te leggen in een structuur van nodes en verbindingen ontstaat er een virtuele netwerktopologie. Hiermee zijn de ontwerp-principes en ontwerp-condities van botnets geanalyseerd middels de techniek van netwerkvirtualisatie.

Welke ontwerp-principes en ontwerp-condities kunnen worden vastgesteld voor netwerkvirtualisatie in het DVCSL? De kenmerken die van belang zijn voor de netwerkvirtualisatie hebben betrekking op de architectuur van het VCSL en het DVCSL en de topologie van het DVCSL. Op basis hiervan is er een belangrijk ontwerp-principe vastgesteld, dat het DVCSL zowel verticaal als horizontaal schaalbaar is. Dit betekend dat de systeemcapaciteit van het DVCSL kan worden uitgebreid door snellere systemen te gebruiken, maar ook door extra systemen aan het DVCSL toe te voegen. Een belangrijke ontwerp-conditie van de virtuele nodes en verbindingen is dat deze te implementeren moeten zijn in een Linux omgeving.

Kunnen op basis van de kenmerken van botnets en de kenmerken van het DVCSL botnetsimulaties worden uitgevoerd middels netwerkvirtualisatie? Door beide virtuele netwerktopologieën met elkaar te vergelijken, kan er een uitspraak worden gedaan of het mogelijk is om de kenmerken van botnets te virtualiseren in het DVCSL. Hiermee kan op een wetenschappelijke wijze worden vastgesteld of botnetsimulaties mogelijk zijn in het DVCSL. Het belangrijkste resultaat van deze analyse is dat een botnet moet kunnen draaien in een Linux omgeving. Hiermee is vastgesteld dat botnets kunnen worden gesimuleerd in het DVCSL.

Het vinden van een geschikte botnet die in het DVCSL gesimuleerd kon worden was de grootste uitdaging van dit onderzoek. Uiteindelijk is er een Open Source botnet gevonden die voldoet aan de kenmerken van een kwaadaardige botnet en daarnaast relatief veilig kan worden gebruikt in een afgesloten virtuele omgeving. Met deze botnet zijn er een aantal experimenten uitgevoerd in het DVCSL om vast te kunnen stellen of het DVCSL geschikt is om hierin botnetsimulaties te kunnen uitvoeren.

Tijdens deze botnetsimulatie is er een beperking van het DVCSL naar voren gekomen. Als meerdere DVCSL nodes gebruik maken van het zelfde fysieke netwerkverbinding, dan wordt de virtuele bandbreedte beperkt. Hiervoor is ook een oplossing gevonden door voor elke DVCSL node een router op te nemen in de virtuele netwerktopologie. In de botnetsimulatie casus is deze oplossing toegepast.

Als laatste is er een praktijk voorbeeld uitgewerkt waarin er een botnetsimulatie is uitgevoerd zoals deze ook op het 'echte' Internet zich voordoen. Hiermee is op een wetenschappelijk empirische wijze het antwoord gegeven op de hoofd onderzoeksvraag.

Zijn botnetsimulaties mogelijk in het DVCSL? In dit empirisch deel van het onderzoek is aangetoond dat het mogelijk is om een botnet omgeving te kunnen simuleren in het DVCSL. Door op een beperkte schaal experimenten uit te voeren is zowel in theorie als in de praktijk bewezen dat botnetsimulaties mogelijk zijn in het DVCSL. Hierdoor is het mogelijk met beperkte middelen, herhaalbaar en controleerbaar wetenschappelijk empirisch onderzoek te kunnen doen naar botnets door gebruikt te maken van het DVCSL van de OU.

Trefwoord: botnet, virtual computer security lab, simulatie, netwerkvirtualisatie

SUMMARY

Since the birth of the Open Internet in 1983 computer systems in the whole world can freely communicate with each other. Due to the open nature of the Internet, security of the computer systems began to take an increasingly important role, partly because these computer systems could also be abused.

Around the year 2000 we saw the first botnet arise, this meant that the larger information system could be attacked by the use of multiple with bots infected computer systems. Bots or software robots are unwanted software applications that are remotely managed by a botmaster, often unknowingly by the owner of the infected computer system. The botmaster can give these bots commands and thereby cause harm to users on the Internet. The botmaster accomplices this via the C&C Systems of the botnet.

Simulating large botnet systems in a closed computer lab is a labor intensive process that involves allot of resources. This research, made a first exploration of the possibilities to use a virtual computer lab for this instead. For quite a while students within the 'Open Universiteit' (OU) can perform networking and security exercises inside an encapsulated common networking environment so called 'Virtual Computer Security Lab' (VCSL). Also there is a distributed version of the VCSL available called the 'Distributed Virtual Computer Security Lab' (DVCSL).

The central question of this research is: **are botnet simulations possible in the DVCSL?**

In order to properly answer this question, scientific empiric research and a practice oriented research model are used. A research model has been created after studying recent scientific research on the three research areas. These research areas are botnets, the DVCSL, and network virtualization. In addition, a conceptual model is created which shows the relationship between botnets and the DVCSL, and also the role that network virtualization fulfilled herein. Three research questions are made based on the main research question and these two models.

Which design principles and design conditions can be found for the virtualization of botnets? First there are important characteristics established for the virtualization of botnets. These relate to the architecture, topology, and the life cycle of botnets. By describing and recording the attributes and characteristics we can create a virtual network topology with a structure of nodes and connections. With this, the design principles and design conditions of botnets were analyzed using the technique of network virtualization.

Which design principles and design conditions can be found for network virtualization in DVCSL? The features that are of interest to the network virtualization relate to the architecture of the VCSL and the DVCSL and the topology of the DVCSL. On this basis, there is a further important design principle that says that DVCSL need to be both horizontally and vertically scalable. This means that the system capacity of the DVCSL can be extended by using more advanced systems or by adding more systems to the DVCSL. An important design condition of the virtual nodes and connections is that they must be able to be implemented within a Linux environment.

Is it possible that based on the characteristics of botnets and characteristics of the DVCSL to perform botnet simulations using network virtualization? By comparing both network topologies it's possible to come to a conclusion whether it is possible to virtualize characteristics of botnets in the DVCSL. This way we can determine in a scientific way whether botnet simulations are possible in the DVCSL. The most prominent result is that a botnet should be able to run within a Linux environment. With this it has been proven that botnets can be simulated in the DVCSL.

Finding a suitable botnet that could be simulated in the DVCSL was the biggest challenge of this research. Ultimately, there is an Open Source botnet found that meets the characteristics of a malicious botnet. A botnet that in addition of the aforementioned characteristics can also be used relatively safe in a closed virtual environment. There where a number experiments carried out in the DVCSL with this botnet to be able to determine whether or not the DVCSL is suitable to be able to carry out botnet simulations.

During this botnet simulation a limitation of the DVCSL emerged. If the multiple DVCSL nodes use the same physical network connection, the virtual bandwidth of the DVCSL is restricted. The solution found for this is including a router for each DVCSL node in the virtual network topology. The botnet simulation case applied this solution.

Finally, there are botnet simulations developed and implemented as if they would also occur on the 'real' Internet. With this the answer to the main research question is given.

Are botnet simulations possible in the DVCSL? In this empirical part of the research is shown that it is possible to simulate a botnet environment in the DVCSL. By carrying out experiments on a limited scale its proven theoretically and practically that botnet simulations are possible in the DVCSL. This makes it possible to perform repeatable and verifiable scientific empirical research with limited resources within the DVCSL of the OU.

Keywords: botnets, virtual computer security lab, simulation, network virtualization

1

INLEIDING

Sinds de geboorte in 1983 van het Open Internet zoals wij deze nu kennen, kunnen computersystemen over de gehele wereld met elkaar communiceren. Door het open karakter van het Internet is veiligheid, en de daaraan gekoppelde beveiliging, een steeds grotere rol gaan spelen. Het Internet heeft vandaag de dag zo'n grote rol in onze samenleving dat het niet meer weg te denken is. Het is een voorziening geworden die misschien wel net zo belangrijk is als gas, water en elektra.

Het Internet kan ook worden gebruikt om schade aan te brengen aan de gebruikers van het Internet of aan de diensten die aan het Internet gekoppeld zijn. Een van deze ongewenste toepassingen zijn botnets, deze bestaande uit softwarerobots of ook wel bots benoemd, die zich kunnen verspreiden via onbeschermde computersystemen [CJM05]. Vaak zijn deze botnets verantwoordelijk voor onderandere het onbeschikbaar maken van publieke dienstverlening van zowel commerciële bedrijven als overheden.

1.1. ACHTERGROND

Om grip te krijgen op deze ongewenste software toepassingen is er veel onderzoek nodig op het terrein van detectie van botnets, maar ook naar de werking van deze botnets. In dit wetenschappelijk onderzoek wordt gekeken naar de mogelijkheden om de werking van deze botnets te kunnen simuleren in een gedistribueerd virtueel computer security lab voor afstandsonderwijs, het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) van de 'Open Universiteit' (OU).

Het doel van dit wetenschappelijk onderzoek is te achterhalen wat de mogelijkheden zijn van het DVCSL voor het simuleren van botnets. Dit onderzoek kent twee primaire deelgebieden, namelijk botnets en het DVCSL. Naast deze twee primaire deelgebieden wordt er gebruik gemaakt van de technieken van netwerkvirtualisatie om te kunnen bepalen of het mogelijk is botnets te kunnen simuleren in het DVCSL.

1.1.1. BOTNETS

Botnets bestaan vaak uit een groot aantal softwarerobots die beheerd worden door een beheerder ook wel botherder of botmaster genoemd. De eerste botnets zijn rond het jaar 2000 voor het eerst ontdekt en het aantal botnets blijft groeien [Liu+11]. Door deze groei

en de steeds nieuwere technieken die deze botnets gebruiken wordt de schade die deze botnets aanrichten steeds groter [Bai+09].

Het doel van dit onderzoek richt zich niet op het ontdekken en onschadelijk maken van deze botnets. Het richt zich op de werking van deze botnets en hoe deze gesimuleerd kunnen worden in het DVCSL.

Tijdens de literatuurstudie zijn een aantal kenmerken van botnets vastgelegd waar dit onderzoek rekening mee moet houden, zoals topologie, architectuur en levenscyclus. Op basis van deze kenmerken is onderzocht of het mogelijk is om een botnet te kunnen virtualiseren.

1.1.2. NETWERKVIRTUALISATIE

Om de kenmerken van botnets te kunnen analyseren en virtualiseren is er een bepaalde abstracte tussenvorm nodig. Hiervoor zijn de technieken van netwerkvirtualisatie gebruikt [CB08]. Hiermee kan er een algemene vertaling worden gemaakt zodat er inzicht verkregen kan worden aan welke ontwerp-principes en ontwerp-condities de virtualisatie van botnets moeten voldoen.

1.1.3. DVCSL

Binnen de OU wordt al een enige tijd gebruik gemaakt van het 'Virtuele Computer Security Lab' (VCSL) bij de cursus 'Security and IT' [VK09]. Inmiddels is hiervan een gedistribueerde variant gemaakt, het DVCSL waarbij meerder VCSL omgevingen met elkaar verbonden kunnen worden tot één grote virtueel computer security laboratorium.

Om te kunnen bepalen of de ontwerp-principes en ontwerp-condities het DVCSL voldoet voor netwerkvirtualisatie. Zijn tijdens de literatuurstudie een aantal kenmerken vastgelegd waar dit onderzoek rekening mee dient te houden. Deze kenmerken van het DVCSL hebben betrekking op de topologie, architectuur van het VCSL en architectuur van het DVCSL.

1.2. PROJECTKADER

Het projectkader waarin het wetenschappelijk onderzoek is uitgevoerd is onderverdeeld in een aantal paragrafen. In de eerste paragraaf wordt stil gestaan bij de achtergronden van dit onderzoek. Vervolgens wordt het probleemgebied uitgelegd en afgesloten met de hoofd onderzoeksvraag.

1.2.1. PROBLEEMGEBIED

Om deze vorm van kwaadaardige software toepassingen beter te kunnen ontdekken en te bestrijden is meer kennis nodig van deze toepassingen. Zoals hoe deze botnets zich gedragen en hoe deze zich ontwikkelen [TA11]. Dit is niet of nauwelijks mogelijk om dit op het 'echte' Internet te kunnen uitvoeren mede omdat het zo groot en complex is geworden [Liu+11].

Een van de oplossingen hiervoor is het virtualiseren van het Internet in een afgesloten geschaalde omgeving middels netwerkvirtualisatie. Op dit terrein is al behoorlijk veel onderzoek gedaan [BB07; Cal+10; EG11]. Het opzetten en herbouwen van een dergelijk afgesloten geschaalde Internet omgeving kost veel tijd en resources. Wellicht zijn er nieuwe

manieren en mogelijkheden voor het opzetten van een afgesloten geschaalde Internet omgeving en kunnen deze bijdragen aan het onderzoek naar botnets.

Binnen een aantal universiteiten wordt er gebruik gemaakt van een VCSL om netwerken te simuleren voor onder andere beveiligingscursussen [VK09]. Met deze VCSL kunnen kleine netwerkomgevingen worden gebouwd en getest. Wellicht is het mogelijk om hiermee een geschaalde versie van het Internet te kunnen simuleren.

Met de gedistribueerde versie van deze VCSL kunnen grotere gevirtualisuerde netwerken worden gemaakt [Vra+11; HV12; Haa+14]. Wellicht is het mogelijk dat in het DVCSL botnets kunnen worden gesimuleerd waardoor onderzoek naar botnets weer een stap verder gebracht kan worden.

Het grote voordeel van deze gevirtualiseerde netwerken in het DVCSL is dat deze snel en gemakkelijk worden gebouwd omdat deze middels software kunnen worden gerealiseerd. Hierdoor kan een simulatie gemakkelijk worden gestart en herstart door de mogelijkheid om iedere keer met een van te voren gedefinieerde beginsituatie te kunnen starten.

1.2.2. ONDERZOEKSVRAAG

Op basis van de hierboven benoemde terreinen van botnets, netwerkvirtualisatie en de werking van het DVCSL kan de hoofd onderzoeksvraag worden geformuleerd, namelijk:

Zijn botnetsimulaties mogelijk in het DVCSL?

1.3. LEESWIJZER

In Hoofdstuk 2 'Methode' wordt de gebruikte onderzoeksmethode beschreven waarin het conceptueel ontwerp van dit wetenschappelijk onderzoek is uitgewerkt. Hierin is de hoofd onderzoeksvraag opgedeeld in een aantal onderzoeksvragen.

In de volgende twee hoofdstukken worden de twee onderzoeksdeelgebieden behandeld waarin recentelijk wetenschappelijk onderzoek is bestudeerd. Ook is hierin een eerst analyse gedaan van deze onderzoeksdeelgebieden. In Hoofdstuk 3 'Botnets' wordt de theorie van botnets geanalyseerd en in Hoofdstuk 4 'DVCSL' wordt de werking van het DVCSL geanalyseerd.

Vervolgens wordt in Hoofdstuk 5 'Botnetsimulatie' onderzocht of het mogelijk is op basis van deze twee analyses een botnetsimulatie uitgevoerd kan worden in het DVCSL. In Hoofdstuk 6 'DVCSL botnetsimulatie casus' worden de botnetsimulatie mogelijkheden van het DVCSL beschreven om hiermee de hoofd onderzoeksvraag te kunnen beantwoorden.

In het afsluitende Hoofdstuk 7 'Conclusie' wordt de conclusie van dit onderzoek gegeven en wordt er kort stil gestaan bij de resultaten en de gevonden beperkingen van dit wetenschappelijk onderzoek. Dit hoofdstuk wordt afgesloten met mogelijk toekomstig onderzoekswerk en een zelf reflectie.

2

METHODE

In dit hoofdstuk is beschreven hoe het wetenschappelijk onderzoek is uitgevoerd en welke methodiek hiervoor is gebruikt. Als eerste is er een literatuurstudie uitgevoerd naar recentelijk verricht wetenschappelijk onderzoek op het terrein van botnets, netwerkvirtualisatie en de werking van het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL). Op basis van de bestudeerde literatuur is er een conceptueel ontwerp gemaakt om de onderzoeksvraag te kunnen beantwoorden.

2.1. CONCEPTUEEL ONTWERP

In het conceptueel ontwerp van een wetenschappelijk onderzoek wordt beschreven wat de doelstelling van het onderzoek is en welke onderzoeksvragen hiervoor beantwoord moeten worden en tevens welke onderzoeksmethodes worden toegepast. Hiervoor is er eerst na de bestudeerde literatuur van de drie onderzoeksdeelgebieden een onderzoeksmodel gemaakt. Hierin is aangegeven hoe het onderzoek uitgevoerd dient te worden en welke onderzoeksmethode hiervoor wordt gebruikt. Om de relaties tussen de drie onderzoeksdeelgebieden aan te geven is er ook een conceptueel model gemaakt waarin deze relaties worden beschreven.

2.1.1. DOELSTELLING

De doelstelling van dit wetenschappelijke onderzoek is te achterhalen of het mogelijk is om botnetsimulaties te kunnen uitvoeren in het DVCSL.

Zoals in de inleiding staat aangegeven kunnen botnets veel schade aanrichten aan systemen die zijn gekoppeld aan het Internet. Om inzicht te krijgen in de werking van botnets is er al veel onderzoek verricht naar de werking van deze botnets [CJM05; Raj+06; BB07; Bai+09; Liu+11; TA11; Zha+12; RMG13; Sil+13]. Maar ook botnets worden steeds verbeterd zodat ze minder opvallen in het reguliere Internet verkeer. Het doel van het al uitgevoerde onderzoek is vaak gericht op het herkennen van deze botnets zodat deze onschadelijk gemaakt kunnen worden.

Door de enorme toename van deze botnets en het gebruik van steeds nieuwe technieken door deze botnets is er nog meer wetenschappelijk onderzoek nodig. Bij het onderzoek naar de werking en detectie van botnets zijn nieuwe technieken gebruikt, zoals virtualisatie

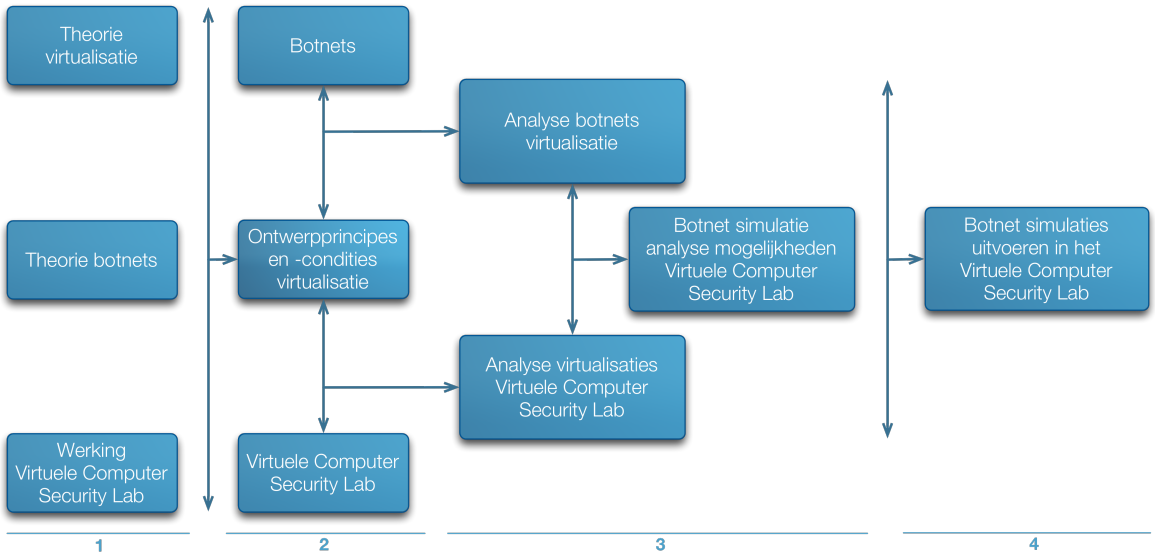
en simulatie [Cal+10]. Vaak is hiervoor een grote infrastructuur nodig, waardoor er veel kosten worden gemaakt voor het opzetten en onderhouden van deze fysieke infrastructuur.

Het doel is om te onderzoeken of het mogelijk is om een botnet te kunnen simuleren in een virtuele netwerkinfrastructuur. Met het DVCSL van de 'Open Universiteit' (OU) kan zo'n virtuele netwerkinfrastructuur worden gebouwd [VK09; Haa+11; Vra+11; HV12; Haa+14], maar de vraag is of het DVCSL ook geschikt is om botnetsimulaties te kunnen uitvoeren.

Om deze vraag te kunnen beantwoorden is er een gestructureerd wetenschappelijk onderzoek nodig en hiervoor is er eerst een onderzoeksmodel gemaakt.

2.1.2. ONDERZOEKSMODEL

Het onderzoeksmodel is de basis van het uit te voeren wetenschappelijk onderzoek. Het onderzoeksmodel voor dit wetenschappelijk onderzoek is weergegeven in figuur 2.1 en is op te delen in een aantal fasen.



Figuur 2.1: Onderzoeksmodel

In de eerste fase van dit onderzoek is er een literatuur studie gedaan naar de theorie van botnets, netwerkvirtualisatie en de werking van het DVCSL.

Op basis van opgedane kennis in de eerste fase en de gestelde onderzoeksvraag is gekozen voor een wetenschappelijk empirisch onderzoek op een praktijk-gericht onderzoeksmodel [VD15]. Dit houdt in dat er proefondervindelijk wordt vastgesteld of het mogelijk is om botnetsimulaties uit te voeren in het DVCSL. Dit is de laatste fase van dit onderzoek. Daarvoor moet op wetenschappelijke wijze worden bepaald of dit mogelijk is en wat hiervoor de randvoorwaarden zijn. Het doel van deze fase is om voldoende wetenschappelijke kennis op te doen over de drie onderzoeksgebieden en een conceptueel ontwerp voor dit onderzoek op te leveren.

In de tweede fase van dit wetenschappelijk onderzoek worden algemene ontwerp-principes en ontwerp-condities die van belang zijn voor virtualisatie geanalyseerd. Dit is gedaan door

recentelijk wetenschappelijk onderzoek te analyseren die is opgedaan in de eerste fase van dit onderzoek. Wat deze ontwerp-principes en ontwerp-condities precies zijn wordt in het onderzoek verder uitgewerkt. Een voorbeeld van een ontwerp principe is dat het DVCSL op bijna alle besturingssystemen kan draaien. Eerst wordt er een analyse gedaan van de virtualisatie van botnets en welke kenmerken hierbij een rol spelen. Ook wordt er een analyse gedaan naar de virtualisatie mogelijkheden van het DVCSL en welke kenmerken hierbij een rol spelen. Een kenmerk die hierbij een rol kan spelen zou bijvoorbeeld de architectuur van het DVCSL kunnen zijn. Voor deze kenmerken gelden ook dat deze verder worden uitgewerkt in dit onderzoek. Na het afronden van de tweede fase zijn de ontwerp-principes en ontwerp-condities voor de virtualisatie van botnets in het DVCSL bekend.

Op basis van deze twee analyses wordt in de derde fase van dit onderzoek vastgesteld welke kenmerken een rol spelen bij de virtualisatie van botnets in het DVCSL. Hierbij kan worden geanalyseerd welke kenmerken er wel en welke kenmerken er niet gevirtualiseerd kunnen worden. Na deze fase kan op een wetenschappelijke wijze worden bepaald of het mogelijke is botnetsimulaties te kunnen uitvoeren in het DVCSL.

Met de bevindingen van de derde fase kan in de laatste fase van dit wetenschappelijk onderzoek worden vastgesteld of het ook praktisch mogelijk is botnetsimulaties uit te kunnen voeren in het DVCSL. Dit is het empirisch deel van dit wetenschappelijk onderzoek. Het resultaat van deze fase is een uitgevoerde botnetsimulatie in het DVCSL.

2.1.3. CONCEPTUEEL MODEL

Het conceptueel model van dit wetenschappelijk empirisch onderzoek is weergegeven in figuur 2.2 en bestaat uit de drie onderzoeksdeelgebieden. Dit model beschrijft de relatie tussen botnets en het DVCSL.

Figuur 2.2: Conceptueel model

Zoals in dit figuur is aangegeven is er geen directe relatie tussen botnets en het DVCSL, zoals deze er wel is tussen bijvoorbeeld botnets en het Internet.

Om botnets te kunnen simuleren in het DVCSL moeten deze worden gevirtualiseerd en hiervoor is de theorie van netwerkvirtualisatie gebruikt. De basis van netwerkvirtualisatie is dat entiteiten kunnen worden ondergebracht in een structuur van nodes en verbindingen tussen deze nodes. Door aan deze nodes en verbindingen kenmerken en eigenschappen toe te kennen is het mogelijk om verschillende structuren met elkaar te kunnen vergelijken en hierover uitspraken te doen. Deze structuur van nodes en verbindingen wordt ook wel een netwerktopologie genoemd [CB08].

Door deze abstracte tussenvorm kunnen de kenmerken van botnets en de kenmerken van het DVCSL met elkaar worden vergeleken en op basis van deze analyse kunnen onderbouwde uitspraken worden gedaan over deze kenmerken. De kenmerken die hierbij een rol kunnen spelen zijn bijvoorbeeld architectuur, topologie en gedrag. De architectuur beschrijft waaruit een te onderzoeken object bestaat. De topologie beschrijft de relaties tussen de verschillende entiteiten van een architectuur. En met het gedrag kan de interactie tussen deze entiteiten worden beschreven.

Door de kenmerken van botnets weer te geven in een structuur van nodes en verbindingen in een netwerktopologie en door de kenmerken van het DVCSL ook weer te geven in een structuur van nodes en verbindingen.

Daarna kunnen de kenmerken van botnets en het DVCSL worden geanalyseerd door gebruik te maken van de technieken van netwerkvirtualisatie. Op basis van deze analyse kunnen er onderbouwde uitspraken worden gedaan over de te combineren netwerktopologie.

2.1.4. ONDERZOEKSVRAGEN

Om de hoofd onderzoeksvraag van dit wetenschappelijk empirisch onderzoek te kunnen beantwoorden kan op basis van het onderzoeksmodel zoals deze in figuur 2.1 is weergegeven de volgende onderzoeksvragen worden geformuleerd:

- **Zijn botnetsimulaties mogelijk in het DVCSL?**

Op basis van het onderzoeksmodel kunnen voor de hoofd onderzoeksvraag de volgende onderzoeksvragen worden geformuleerd:

- **Welke ontwerp-principes en ontwerp-condities kunnen worden vastgesteld voor de virtualisatie van botnets?**
- **Welke ontwerp-principes en ontwerp-condities kunnen worden vastgesteld voor netwerkvirtualisatie in het DVCSL?**
- **Kunnen op basis van de kenmerken van botnets en de kenmerken van het DVCSL botnetsimulaties worden uitgevoerd middels netwerkvirtualisatie?**

Als de bovenstaande onderzoeksvragen beantwoord kunnen worden, dan kan ook het antwoord worden gegeven op de hoofd onderzoeksvraag en kan worden vastgesteld of het mogelijk is botnets te simuleren in het DVCSL van de OU.

Deze onderzoeksvragen zijn nog te abstract en kunnen niet direct worden beantwoord. Daarvoor is op basis van het conceptueel model voor elke onderzoeksvraag een aantal deelvragen geformuleerd. Hieronder zijn de onderzoeksvragen met de daarbij geformuleerde deelvragen weergegeven:

- **Welke ontwerp-principes en ontwerp-condities kunnen worden vastgesteld voor de virtualisatie van botnets?**
  - Welke kenmerken van botnets zijn van belang voor de virtualisatie van botnets?
  - Kunnen de kenmerken van botnets worden gevirtualiseerd middels netwerkvirtualisatie?

- **Welke ontwerp-principes en ontwerp-condities kunnen worden vastgesteld voor netwerkvirtualisatie in het DVCSL?**
  - Welke kenmerken van het DVCSL zijn van belang voor de netwerkvirtualisatie?
  - Kunnen de kenmerken van de netwerkvirtualisatie van botnets worden gevirtualiseerd in het DVCSL?
- **Kunnen op basis van de kenmerken van botnets en de kenmerken van het DVCSL botnetsimulaties worden uitgevoerd middels netwerkvirtualisatie?**
  - Welke kenmerken van botnets kunnen worden gesimuleerd in het DVCSL?
  - Welke kenmerken van botnets kunnen niet worden gesimuleerd in het DVCSL?
  - Zijn er aanpassingen nodig aan het DVCSL om botnetsimulaties te kunnen uitvoeren?

In de volgende hoofdstukken worden eerst deze deelvragen beantwoord. Door op een gestructureerde manier de antwoorden op de gestelde onderzoeksvragen te formuleren, kan er een uitspraak worden gedaan over de hoofd onderzoeksvraag. Het ultieme bewijs zou zijn dat er een praktijkvoorbeeld van een botnetsimulatie gegeven kan worden.

2.2. AFBAKENING

De nadruk van dit wetenschappelijk empirisch onderzoek ligt op het onderzoeken van de mogelijkheden van botnetsimulatiesin het DVCSL. Hiermee is wel een algemeen onderzoekskader aangegeven, maar het onderzoekswerkveld is nog te groot en kan verder worden afgebakend.

De keuzes die zijn gemaakt tijdens dit onderzoek op de drie onderzoeksterreinen komen in de volgende hoofdstukken terug en worden daar verder toegelicht. Daarnaast kunnen er een aantal algemene kaders worden geformuleerd voor dit onderzoek om de beschikbare onderzoekstijd zoveel mogelijk toe te spitsen op de doelstelling van dit onderzoek. Deze zijn hieronder kort toegelicht.

ONDERZOEKSTIJD

Het doel van dit onderzoek is het beantwoorden van de onderzoeksvragen en het uitvoeren van een botnetsimulatie. Hierdoor is het niet mogelijk een uitgebreid onderzoek te doen naar botnets, netwerkvirtualisatie en de werking van het DVCSL.

Hiervoor moet de beschikbare onderzoekstijd zo efficiënt mogelijk worden gebruikt. Daar waar keuzes zijn gemaakt om bepaalde dingen wel of niet verder te onderzoeken worden deze keuzes toegelicht waarom deze zo zijn gemaakt.

GEBRUIK VAN HULPMIDDELEN

Er moet zo veel mogelijk gebruik gemaakt worden van bestaande hulpmiddelen en software componenten. Het uitbreiden van bestaande hulpmiddelen en software componenten is geen onderdeel van dit onderzoek.

GESCHREVEN SOFTWARE

Software die geschreven wordt tijdens dit onderzoek moet werkend worden opgeleverd met voldoende documentatie. Er hoeft geen compleet functioneel of technische ontwerp van

gemaakt te worden. Het doel van dit onderzoek is niet het schrijven van software, maar verschillende software componenten werkend te krijgen.

ANALYSE VAN DE GEGENEREERDE GEGEVENS

Het analyseren van de gegevens die door de botnetsimulaties gegenereerd kunnen worden is geen onderdeel van dit onderzoek. Daar waar het mogelijk is moeten deze gegenereerde gegevens beschikbaar gemaakt kunnen worden, zodat deze in een vervolg onderzoek geanalyseerd kunnen worden.

3

BOTNETS

In dit hoofdstuk is de analyse gedaan van botnets, zoals deze is aangegeven in het onderzoeksmodel van paragraaf 2.1.2. Als eerste wordt het gerelateerd wetenschappelijk onderzoek op het gebied van botnets behandeld. Daarna worden de kenmerken van botnets geanalyseerd om vervolgens de kenmerken van een geschikte botnet te kunnen bepalen.

Vervolgens wordt het gerelateerd wetenschappelijk onderzoek van netwerkvirtualisatie behandeld die een relatie heeft met de virtualisatie van botnets. Daarna worden de kenmerken van netwerkvirtualisatie gebruikt om de kenmerken van de botnets topologie te beschrijven.

Als laatste worden de onderzoeksvragen beantwoord.

3.1. ONDERZOEK NAAR BOTNETS

Er is veel wetenschappelijk onderzoek gedaan naar alle facetten van botnets, maar voor dit onderzoek is specifiek gezocht naar recentelijk wetenschappelijk onderzoek dat op een of andere manier te maken heeft met het simuleren en het virtualiseren van botnets [CJM05; Raj+06; BB07; Bai+09; Liu+11; TA11; Zha+12; RMG13; Sil+13].

Een botnet bestaat vaak uit maar enkele componenten. In de bestudeerde wetenschappelijke literatuur worden deze componenten wel beschreven alleen verschilt de definitie hiervan per onderzoek en type botnet. Er is wel een algemene architectuur te beschrijven waarin deze componenten beschreven zijn [Bai+09; RMG13].

Er is een entiteit die het botnet beheert, ook wel botherder of botmaster genoemd. Het botnet zelf bestaat uit een structuur van softwarerobots en één of meerdere commando systemen. Deze commando systemen worden vaak C&C Systeem genoemd. De softwarerobots ofwel bots maken gebruik van verschillende technieken om onbeschermde systemen te infecteren om hiermee controle te krijgen over deze besmette systemen. De botmaster kan een opdracht geven aan het C&C Systeem die dan uitgevoerd wordt door deze bots op de besmette systemen. Deze architectuur is de basis van een botnet en is een belangrijk kenmerk waarmee rekening gehouden moet worden in dit onderzoek.

Daarnaast is er ook veel studie gedaan naar de verschillende soorten en types botnets en de daarbij behorende specifieke kenmerken [TA11; RMG13]. De individuele botnets zijn hierbij onderverdeel in groepen. Deze onderverdeling is met name gedaan op basis van de

topologie van botnets. De botnets die onderzocht zijn hebben een bepaalde topologie en daarbij een aantal specifieke kenmerken, zoals bijvoorbeeld de gebruikte communicatie protocollen. Dit geheel wordt ook wel de botnet taxonomie genoemd [RMG13]. Al deze specifieke communicatieprotocollen zijn van minder belang voor dit onderzoek.

Ook is er onderzoek gedaan naar het gedrag van botnets [Raj+06; BB07; Cal+10; Zha+12]. Het gedrag van botnets kan 'in het wild' worden geanalyseerd, maar ook in een afgesloten netwerkomgeving. Het grote verschil is dat het gedrag van de gehele botnet omgeving te analyseren is in een afgesloten netwerkomgeving. Bij botnets die 'in het wild' voorkomen kan dat meestal niet. En kan maar een beperkt deel van dit gedrag worden geanalyseerd omdat niet alle botnet entiteiten bekent zijn, of wel dat er onvoldoende informatie van verzameld kan worden. Bij het analyseren van het gedrag van botnets in een afgesloten netwerkomgeving is dat vaak wel mogelijk, omdat alle botnet entiteiten zich in dat afgesloten netwerk bevinden. Toen deze techniek voor het eerst is ingezet bij het analyseren van botnets bestonden deze afgesloten netwerken uit fysieke systemen die compleet gescheiden waren van het echte Internet [Raj+06; BB07; Cal+10]. Een ander belangrijk voordeel van gescheiden netwerken is dat er geen besmetting kan plaats vinden van het echte Internet of vanuit het echt Internet. De kosten van het opzetten en onderhouden van deze gescheiden fysieke netwerkomgevingen zijn vaak hoog.

Er zijn ook beperkingen aan het kunnen emuleren of simuleren van botnets. Het verschil is dat bij het emuleren niet de echte programma code wordt gebruikt maar een programmacode dat hetzelfde gedrag vertoont en met het simuleren wordt wel de echte programmacode gebruikt. Bij het emuleren van bijvoorbeeld een communicatie protocol dat door botnets gebruikt kan worden is er een technisch limiet aan het te aantal te emuleren entiteiten [EG11]. Bij het simuleren van botnets in een virtuele omgeving zijn er ook beperkingen zoals de grootte van het botnet en de gebruikte technieken van een botnet [Liu+11]. Een ander facet is dat botnets ook steeds nieuwe technieken gebruiken en hebben hierbij ook de weg naar de Cloud gevonden. Hierdoor zijn er nog veel meer doelwit systemen ter beschikking gekomen die relatief beperkter zijn qua resources dan de traditionele doelwit systemen [Zha+12].

3.2. KENMERKEN BOTNETS

In de bestudeerde wetenschappelijke literatuur van botnets zijn er een aantal kenmerken naar voren gekomen. De kenmerken die een belangrijke rol spelen bij de virtualisatie van botnets, zoals deze hierboven zijn beschreven hebben, te maken met:

- Architectuur
- Topologie
- Levenscyclus

In de volgende drie paragrafen worden deze kenmerken verder toegelicht en waarom deze belangrijk zijn voor de virtualisatie van botnets. Ook zijn de gemaakte keuzes voor de afbakening van dit onderzoek voor deze kenmerken hierin opgenomen.

3.2.1. ARCHITECTUUR

Er zijn vele verschillende botnets met hun eigen architectuur [RMG13], waarbij een aantal elementen vrijwel altijd voorkomen en andere elementen die specifiek zijn voor een speci-

fieke botnet. De belangrijkste architectuur elementen zijn [Sil+13]:

- Een C&C Systeem waaruit het botnet wordt aangestuurd en onderhouden.
- De besmette software en een onbeschermde dienst om het botnet mee op te zetten en uit te breiden.
- De met een bot besmette bot systeem waarmee botnet aanvallen uitgevoerd kunnen worden.
- Één of meerder doelwit systemen of diensten die aangevallen kunnen worden door het botnet.

De definitie van een doelwit systeem in de literatuur verschilt per onderzoek. Voor dit onderzoek is dit een systeem die aangevallen kan worden door één of meerdere bots. Een mogelijke botnet aanval kan zijn om het doelwit systeem te besmetten met een bot.

Daarnaast is er nog een botherder ook wel botmaster genoemd, die het botnet beheert en aanstuurt. Deze is geen onderdeel van de technische architectuur, maar van de functionele architectuur.

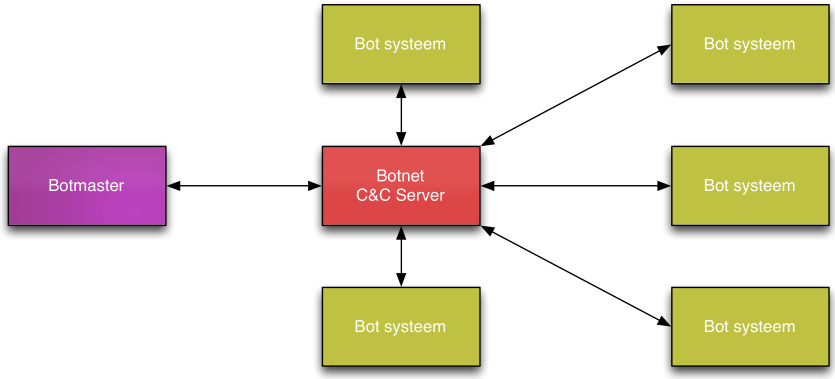
3.2.2. TOPOLOGIE

Met de topologie van botnets beschrijven we de communicatie structuur van een botnet. Hierbij gaat het met name om de communicatie tussen de botmaster en het C&C Systeem en de bots. Maar ook de communicatie tussen de C&C Systeem en de bots is hierbij van belang.

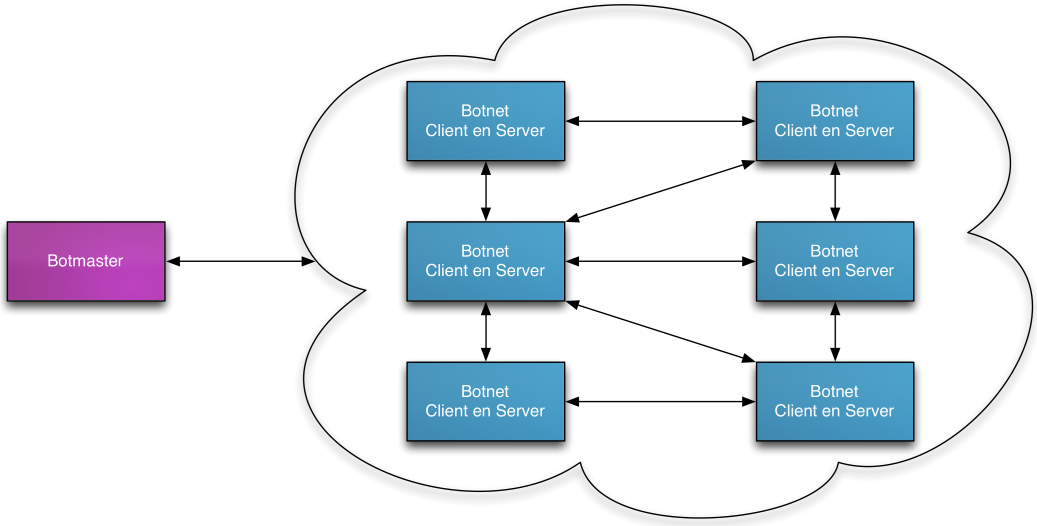
Botnets hebben zich door de jaren heen steeds door ontwikkeld waarbij de aansturing van de botnets steeds geavanceerde is geworden. Eerst was de communicatie tussen het C&C Systeem en de bots op een gecentraliseerde manier georganiseerd en later is dit via een gedistribueerde manier georganiseerd. Tegenwoordig communiceren het C&C Systeem en de bots voornamelijk op een hybride manier waarbij er geen vaste communicatie structuur is [CJM05; Bai+09; RMG13]. Dit is een van de kenmerken van botnets waarmee rekening gehouden moet worden en kan worden gekenmerkt als de topologie van botnets. De botnet topologie is schematisch weergegeven in figuur 3.1.

De topologie van botnets is onder te verdelen in de drie verschillende netwerkstructuren waar een botnet uit kan bestaan [Bai+09]:

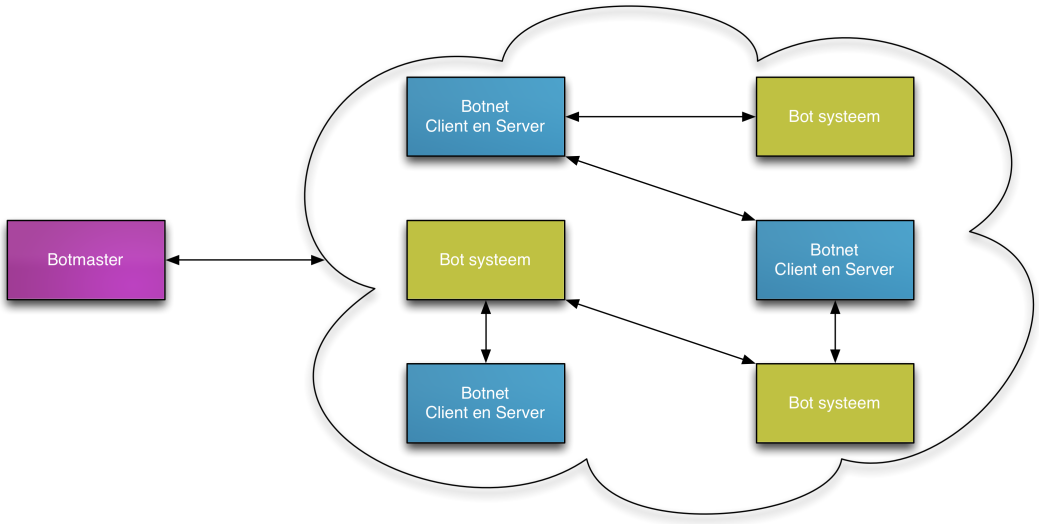
- Gecentraliseerde topologie
  De botmaster stuurt rechtstreeks het centrale C&C Systeem van het botnet aan en alle bots hebben een verbinding met het centrale C&C Systeem. Dit is de meest simpele vorm van communicatie en is te vergelijken met een client-server architectuur.
- Gedistribueerde topologie
  De botmaster stuurt een bot in het botnet aan en het botnet zorgt er voor dat deze informatie naar alle bots wordt doorgegeven. Hierbij zijn de bots ook zelf een soort C&C Systeem. Een belangrijk punt is dat de informatie van het botnet wordt gedeeld met de meeste botnet entiteiten.



(a) Gecentraliseerde topologie



(b) Gedistribueerde topologie



(c) Hybride topologie

Figuur 3.1: Botnet Topologie

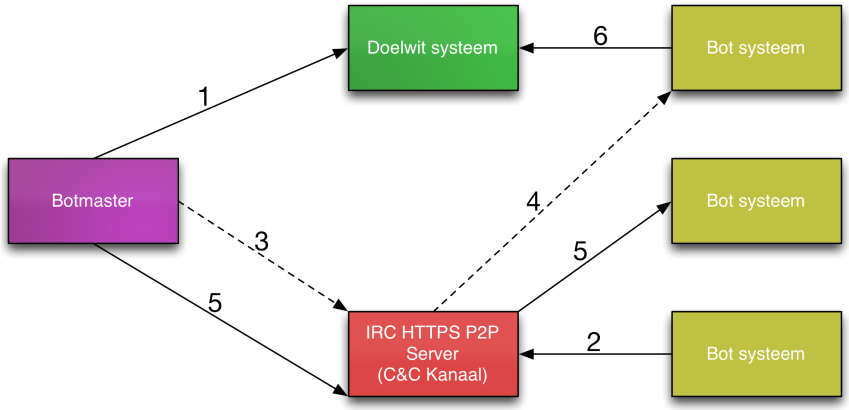
- Hybride topologie
  De botmaster stuurt of een bot of een C&C Systeem aan, waarbij de informatie via de ene bot naar een andere bot of C&C Systeem gaat. Het concept achter deze structuur is dat een bot of C&C Systeem maar een zeer beperkt aantal andere bots of C&C Systeem kent, maar dat alles via-via gekoppeld is. Er is geen duidelijk communicatie structuur te herkennen, daarom wordt dit ook wel een ongestructureerde netwerkarchitectuur genoemd.

Als afbakening voor dit onderzoek heeft een botnet topologie met de centrale netwerkarchitectuur de voorkeur omdat deze het gemakkelijkst te simuleren is [Cal+10].

3.2.3. LEVENSCYCLUS

De levenscyclus van een botnet beschrijft de interactie tussen de botmaster, de bots en het C&C Systeem. De levenscyclus varieert per type en individuele botnet. Er zijn een aantal globale fasen te herkennen bij alle botnets en de daarbij behorende kenmerken [TA11; RMG13; Bai+09].

De levenscyclus van botnets bestaat uit een aantal fases, zoals een infectie fase waarbij een doelwit systeem wordt geïnfecteerd met een bot. Een fase waarbij een actieve bot zich bekent maakt bij het C&C Systeem. En de aanvalsfase waarbij de botmaster een opdracht geeft aan het C&C Systeem, die op zijn beurt deze door geeft aan de aangesloten bots, die deze dan vervolgens uitvoerd [Raj+06]. Daarnaast is er nog een fase waarbij het C&C Systeem de bots van updates voorziet. De levenscyclus is schematisch weergegeven in figuur 3.2.



1. De botmaster infecteert een doelwit systeem met een bot.
2. Een bot maakt zich bekend bij een C&C Systeem.
3. De botmaster geeft opdrachten door via het C&C Systeem.
4. Het C&C Systeem geeft deze opdrachten door aan het bots.
5. De bots worden voorzien van updates via het C&C Systeem.
6. De bots vallen een doelwit systeem aan.

Figuur 3.2: Botnet Levenscyclus

Welke fasen van de levenscyclus van een botnet geschikt zijn om te simuleren in het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) hangt af van de gekozen botnet. Voor dit onderzoek is niet nodig om alle fasen van een botnet te simuleren. Voor het beantwoorden van de onderzoeksvragen is het al voldoende dat één of twee van deze fases gesimuleerd kunnen worden. Welke fases hangt mede af van de mogelijkheden van het DVCSL en de gekozen botnet.

3.3. GESCHIKTE BOTNET

Voordat er op zoek kan worden gegaan naar een geschikte botnet voor dit wetenschappelijk onderzoek moet eerst worden onderzocht aan welke kenmerken een botnet moet voldoen waarmee de botnetsimulaties uitgevoerd kunnen worden in het DVCSL. Daarnaast zijn er een aantal selectiecriteria's waar rekening mee kan worden gehouden bij het beoordelen van de geschiktheid van een botnet voor dit onderzoek.

3.3.1. KENMERKEN VAN EEN GESCHIKTE BOTNET

Door de beperkte onderzoekstijd kunnen er keuzes worden gemaakt om deze zo optimaal mogelijk te kunnen benutten. Daarvoor zijn er een aantal kenmerken van botnets vast gesteld waaraan een geschikte botnet minimaal zou moeten voldoen:

- Topologie.
  Een botnet is het meest geschikt als deze een gecentraliseerde topologie heeft, omdat deze de meest simpelste vorm van communicatie gebruikt. Een botnet met een hybride topologie is het minst geschikt, omdat hiervan vaak niet bekent is hoe de communicatie tussen de verschillende botnet entiteiten verloopt.

- Levenscyclus.
  Een botnet is geschikt als hiermee minimaal één levensfase mee gesimuleerd kan worden. Als met een botnet meerdere levensfases gesimuleerd kunnen worden heeft dat een voorkeur, omdat dan een keuze gemaakt kan worden welke van de levensfases gebruikt kan worden voor de botnetsimulatie.

3.3.2. SELECTIECRITERIA VAN EEN GESCHIKTE BOTNET

De selectiecriteria van een geschikte botnet zijn onder te verdelen in verplichte eigenschappen en wenselijke eigenschappen. Een geschikte botnet moet minimaal aan de verplichte eigenschappen voldoen, anders is het niet mogelijk om daarmee botnetsimulaties uit te voeren in het DVCSL. Daarnaast zijn er nog een aantal wenselijke eigenschappen aangegeven waaraan een botnet kan voldoen. Voor deze eigenschappen geldt hoe meer er aan deze eigenschappen wordt voldaan hoe geschikter een botnet is.

VERPLICHTE EIGENSCHAP VAN EEN GESCHIKTE BOTNET

De verplichte eigenschap waaraan een geschikte botnet moet voldoen is:

- De botnet moet beschikbaar zijn voor dit onderzoek.
  Het is ook belangrijk dat een botnet gebruikt mag worden voor dit wetenschappelijk onderzoek. Met een botnet kan veel schade worden aangebracht als deze verkeerd wordt gebruikt. Een punt waar ook rekening mee gehouden moet worden is dat een botnet een actief onderdeel is van dit onderzoek en daarmee mogelijk beschikbaar wordt gesteld door dit onderzoek

WENSELIJKE EIGENSCHAPPEN VAN EEN GESCHIKTE BOTNET

Daarnaast zijn er nog een aantal eigenschappen die wenselijk zouden zijn voor een geschikte botnet voor dit wetenschappelijk onderzoek. Deze eigenschappen zijn onder andere:

- De botnet moet niet te complex zijn.
  Een geschikte botnet is niet al te complex, omdat bij een complexere botnet anders te veel onderzoekstijd verloren gaat aan het begrijpen van de werking van de botnet. Dus een geschikte botnet moet een gemakkelijk te begrijpen architectuur, topologie en levensfases hebben.

- De botnet moet gemakkelijk aan te passen zijn.
  Mocht een botnet niet helemaal voldoen aan de gestelde kenmerken, dan moet het mogelijk zijn eventueel de botnet aan te kunnen passen zodat deze hieraan wel kan voldoen. Ook als een botnet geschikt gemaakt moet worden voor het DVCSL dan is het wenselijk dat dit gemakkelijk gedaan kan worden. Een botnet die gemakkelijk aan te passen is is geschikter dan een botnet die niet gemakkelijk aan te passen is.

- Er moet voldoende informatie beschikbaar zijn over de werking en gebruik van de botnet.
  Om een botnet te kunnen gebruiken voor dit wetenschappelijk onderzoek moet er voldoende bekend zijn over de werking van deze botnet omdat deze ook moet werken in het DVCSL. Met name moet er voldoende informatie zijn over de installatie en de configuratie van deze botnet. Daarnaast moet er ook voldoende informatie beschikbaar zijn over de levenscyclus van deze botnet. Als er onvoldoende informatie beschikbaar is over de botnet dan kan de installatie extra veel onderzoekstijd kosten. Maar ook kunnen de botnetsimulaties extra veel tijd kosten. Een geschikte botnet beschikt over voldoende informatie om deze te kunnen gebruiken voor de botnetsimulaties.

- De botnet moet kunnen werken met weinig geheugen en diskruimte.
  Het idee van virtualisatie is met beperkte middelen en resources vergelijkbare acties uit te voeren in een virtuele omgeving, die normaal in een fysieke omgeving worden uitgevoerd. Daarom moet een geschikte botnet kunnen werken met een beperkte diskruimte. Daarnaast moet deze ook in een beperkte geheugenruimte kunnen draaien. Dit geldt met name voor de bots, maar in beperkte mate ook voor de 'Commando en Controle' (C&C). Een botnet die minder resources nodig heeft is daarom geschikter.

3.4. NETWERKVIRTUALISATIE

Netwerkvirtualisatie is een groot onderzoekswerkveld met vele verschillende disciplines. Voor dit wetenschappelijk onderzoek is specifiek gezocht naar recentelijk wetenschappelijk onderzoek dat te maken heeft met het virtualiseren van netwerken in een virtuele omgeving. Als selectie voorwaarde voor dit deel van de literatuur studie is gesteld dat deze een relatie heeft met de twee andere onderzoeksdeelgebieden, botnets of het DVCSL.

In deze paragraaf wordt er alleen gekeken naar de theorie van netwerkvirtualisatie waar deze een relatie heeft met het virtualiseren van botnets.

3.4.1. KENMERKEN NETWERKVIRTUALISATIE

De kenmerken van netwerkvirtualisatie die voor dit wetenschappelijk onderzoek van belang zijn, is de architectuur van de netwerkvirtualisatie zoals deze zijn beschreven in een netwerkvirtualisatie omgeving [CB08].

HOOFDKENMERKEN NETWERKVIRTUALISATIE

De architectuur van de netwerkvirtualisatie bestaat uit de volgende hoofdkenmerken [CB08]:

- Fysieke topologie
  Een fysieke topologie bestaat uit een gewogen graaf met de attributen van alle fysieke verbindingen en de attributen van alle fysieke nodes. Het is een gewogen graaf omdat er gewichten aan de verschillende nodes en of verbindingen kunnen worden gegeven. Zo kunnen er bijvoorbeeld op basis van transportsnelheid routering keuzes worden gemaakt. Van een fysieke topologie kunnen er meerdere virtuele topologieën bestaan, elk met hun eigen specifieke representaties.

- Virtuele topologie
  Een virtuele topologie bestaat uit een gewogen graaf met de attributen van alle virtuele verbindingen en de attributen van alle virtuele nodes. Een virtuele topologie is een representatie van een bijbehorend fysieke topologie.

- Virtuele node
  Een virtuele node kan zowel een virtuele host zijn of een virtuele router. Een virtuele host is de bron- of het doelsysteem van netwerkverkeer. Een virtuele host komt niet binnenin in een netwerk voor, alleen maar aan de uiteinden van een netwerk. Een virtuele router komt daarin tegen alleen binnenin in een netwerk voor en acteert hetzelfde als een fysieke router. Deze is verantwoordelijk voor de routering van het netwerkverkeer van het bron- naar het doelsysteem.

- Virtuele verbinding
  Een virtuele verbinding is een verbinding tussen twee virtuele nodes in een virtueel netwerk. Deze verbinding kan een representatie zijn van één of meerdere fysieke verbindingen in de bijbehorende fysieke netwerktopologie.

Door het combineren van een fysieke netwerkinfrastructuur en gevirtualiseerde netwerken ontstaat er een nieuwe architectuur, ook wel 'Network Virtualization Environment' (NVE) genoemd [CB08]. Hierin wordt de virtuele netwerktopologie gecombineerd met de fysieke netwerktopologie en kunnen hierin de virtuele nodes en virtuele verbindingen tussen deze nodes worden weergegeven.

3.5. KENMERKEN NODES EN VERBINDINGEN

Voor de netwerkvirtualisatie van botnets zijn met name de virtuele netwerktopologie en de eigenschappen van de nodes en verbindingen van belang.

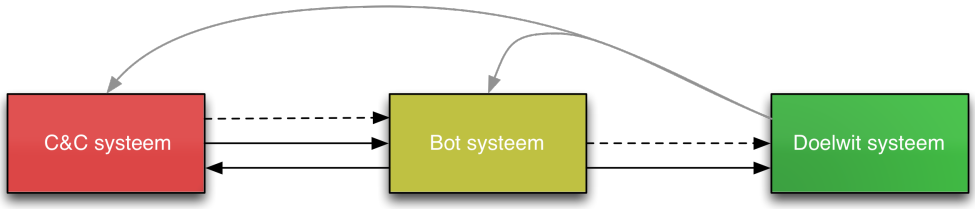
KENMERKEN BOTNETS VOOR NETWERKVIRTUALISATIE

De kenmerken van botnets voor de netwerkvirtualisatie zijn weer te geven in nodes en verbindingen met elk hun eigen specifieke eigenschappen.

De verschillende virtuele netwerktopologieën van botnet zijn terug te vinden in figuur 3.1, waarin de gecentraliseerde topologie, gedistribueerde topologie en de hybride topologie staan beschreven.

De eigenschappen van de nodes en verbindingen zijn terug te vinden in figuur 3.2, waarin vier verschillende entiteiten zijn te herkennen. De botmaster, het C&C Systeem, het bot systeem en het doelwit systeem.

In de volgende twee paragrafen worden de kenmerken van de nodes en verbindingen van een botnet graaf beschreven, zie ook figuur 3.3.



Figuur 3.3: Botnet Graaf

3.5.1. BOTNET NODES

Er zijn drie verschillende nodes te herkennen in een botnet topologie. Deze zijn hieronder kort samengevat. De botmaster is hierin niet opgenomen als een node omdat dit een persoon is.

C&C SYSTEEM

De generieke attributen van een C&C Systeem zijn dat hij opdrachten en updates kan sturen naar de doelwit systemen die door de botmaster worden aangegeven. Daarnaast kan het C&C Systeem de aangesloten bots registreren en van informatie voorzien.

BOT SYSTEEM

Dit is een systeem dat is besmet met een bot. Deze bot kan middels het bot systeem acties uitvoeren die door de botmaster zijn opgedragen via het C&C Systeem.

DOELWIT SYSTEEM

De generieke attributen van een doelwit systeem zijn dat deze geïnfecteerd kunnen worden met een bot door de botmaster, maar ook door een bot systeem aangevallen kan worden. Op het moment dat een doelwit systeem is geïnfecteerd met een bot dan noemen we het een bot systeem.

3.5.2. BOTNET VERBINDINGEN

Er zijn vier verschillende verbindingen te herkennen in een botnet topologie. Deze zijn hieronder kort samengevat.

C&C SYSTEEM NAAR BOT SYSTEEM

Hiermee geeft het C&C Systeem opdrachten aan een bot systeem en daarnaast voorziet het C&C Systeem een bot systeem met updates.

BOT SYSTEEM NAAR C&C SYSTEEM

Hiermee maakt de bot, die zich op het bot systeem bevind, bekent bij het C&C Systeem. Daarnaast kan de bot informatie delen met het C&C Systeem.

BOT SYSTEEM NAAR DOELWIT SYSTEEM

Hiermee valt de bot op het bot systeem een doelwit systeem aan die door de botmaster middels het C&C Systeem is aangegeven. Dit kan in de vorm zijn van een opdracht, bijvoorbeeld in de vorm van een spam-bericht, met daarin de vraag om een bepaalde actie te ondernemen. Of in de vorm van kwaadwillende toepassingen, bijvoorbeeld een virus of een bot.

DOELWIT SYSTEEM NAAR HET BOTNET

Hiermee wordt het botnet van informatie voorzien van een doelwit systeem. Deze informatie kan verstuurd worden naar een bot systeem of naar het C&C Systeem. Hierbij is de gebruiker van het systeem de initiator, door bijvoorbeeld op een link te klikken in een ontvangen spam-bericht.

3.6. ONDERZOEKSVRAAG

In dit hoofdstuk zijn de antwoorden gegeven op een aantal onderzoeksdeelvragen en daarmee kan ook de eerste onderzoeksvraag worden beantwoord. Deze worden in de volgende twee paragrafen toegelicht.

3.6.1. ONDERZOEKSDEELVRAGEN

IHieronder wordt een antwoord gegeven op de eerste en tweede onderzoeksdeelvraag.

WELKE KENMERKEN VAN BOTNETS ZIJN VAN BELANG VOOR DE VIRTUALISATIE VAN BOTNETS?

De kenmerken die van belang zijn voor de virtualisatie van botnets zijn: architectuur, topologie en levenscyclus.

KUNNEN DE KENMERKEN VAN BOTNETS WORDEN GEVIRTUALISEERD MIDDELS NETWERKVIRTUALISATIE?

Door de kenmerken van botnets weer te geven in een virtuele netwerktopologie en deze kenmerken en attributen te koppelen aan de virtuele nodes en virtuele verbindingen kunnen deze worden gevirtualiseerd middels netwerkvirtualisatie.

3.6.2. ONDERZOEKSVRAAG

Op basis van de antwoorden op de voorgaande onderzoeksdeelvragen kan het antwoord op de eerst onderzoeksvraag worden gegeven.

WELKE ONTWERP-PRINCIPES EN ONTWERP-CONDITIES KUNNEN WORDEN VASTGESTELD VOOR DE VIRTUALISATIE VAN BOTNETS?

De ontwerp-principes en ontwerp-condities die vastgesteld zijn voor de virtualisatie van botnets zijn weer te geven door middel van een virtuele netwerktopologie van virtuele nodes en virtuele verbindingen tussen deze virtuele nodes.
Door het beschrijven van de attributen en de kenmerken van deze virtuele nodes en virtuele verbindingen kunnen de kenmerken van een botnet worden vastgelegd.

Op deze wijze kunnen botnets worden gevirtualiseerd middels netwerkvirtualisatie.

4

DVCSL

In dit hoofdstuk is met name de analyse gedaan van de werking van het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) zoals deze in het onderzoeksmodel van paragraaf 2.1.2 is aangegeven. Als eerste wordt het gerelateerd wetenschappelijk onderzoek op het gebied van het 'Virtuele Computer Security Lab' (VCSL) en het DVCSL behandeld. Vervolgens wordt het gerelateerd wetenschappelijk onderzoek van netwerkvirtualisatie behandeld die een relatie heeft met de werking van het DVCSL. Daarna worden de kenmerken van het DVCSL geanalyseerd waarna bepaald kan worden welke van deze kenmerken van belang zijn voor de netwerkvirtualisatie.

Door de kenmerken van de netwerkvirtualisatie van botnets te vergelijken met de kenmerken van het DVCSL kan worden vastgesteld of deze kenmerken zijn te virtualiseren in het DVCSL. Ter voorbereiding voor de botnetsimulaties zijn er twee DVCSL omgevingen geconfigureerd waarbinnen een aantal voor de hand liggende netwerkservices zijn geïmplementeerd.

Als laatste worden de onderzoeksvragen beantwoord.

4.1. Onderzoek naar DVCSL

Binnen de Open Universiteit wordt onderzoek gedaan naar de inzetbaarheid van open standaarden die ingezet kunnen worden in het afstandsonderwijs. Zo is er voor de cursus 'Security and IT' het VCSL ontwikkeld [VK09], dat gebruik maakt van een NetKit omgeving [PR08].

De architectuur van VCSL staat beschreven in verschillende wetenschappelijke artikelen en wordt op verschillende manieren gebruikt bij het afstandsonderwijs van de 'Open Universiteit' (OU) [VK09; HV12; Haa+14]. De basis van het VCSL is een NetKit omgeving waarin kleine Linux systemen kunnen worden gestart die maar enkele tientallen megabytes aan werkgeheugen in beslag nemen. Daarnaast wordt het schijfgeheugen gedeeld door alle systemen maar worden alleen de wijzigingen ten opzichte van dit gedeelde schijfgeheugen per systeem apart opgeslagen.

Het VCSL is steeds verder uitgebreid met meer functionaliteiten die de OU studenten kunnen ondersteunen bij hun studie. Één van deze uitbreidingen is het gedistribueerd maken

van de VCSL, naar voorbeeld van het generieke model van het 'Distributed Computer Security Lab' (DCSL) [Yan+04]. Hiermee kunnen er meerder VCSL omgevingen aan elkaar worden gekoppeld tot één DVCSL omgeving [Vra+11; Haa+11].

De techniek die hiervoor is gebruikt is gebaseerd op het 'Virtual Private Network' (VPN) concept [CB08]. Hierbij wordt het te virtualiseren netwerkverkeer over een bestaande fysieke netwerkverbinding gestuurd, waarbij het te virtualiseren netwerkverkeer wordt ingepakt en op de plaats van bestemming weer wordt uitgepakt. Op deze wijze kan het te virtualiseren netwerkverkeer compleet gescheiden blijven van het fysieke netwerkverkeer. Deze techniek wordt ook gebruikt in 'Peer-to-Peer' (P2P) verbindingen, waar sommige typen botnets ook gebruik van maken.

Door meerdere VCSL omgevingen aan elkaar te koppelen ontstaat er een DVCSL omgeving. De basis van het DVCSL is het handmatig opzetten van een verbinding tussen de beide VCSL omgevingen[Vra+11]. Het handmatig opzetten van deze DVCSL verbindingen is een lastig en een foutgevoelig proces.

4.2. KENMERKEN DVCSL

In de bestudeerde wetenschappelijke literatuur van het DVCSL zijn een aantal kenmerken naar voren gekomen van het VCSL en het DVCSL. De kenmerken die een belangrijke rol spelen bij de werking van het DVCSL hebben te maken met:

- Architectuur VCSL
- Architectuur DVCSL
- Topologie DVCSL

4.2.1. ARCHITECTUUR VCSL

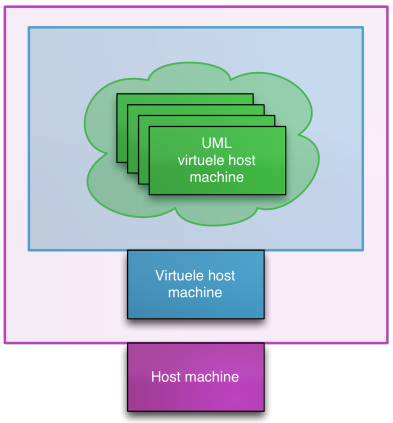
De architectuur van het VCSL bestaan uit een aantal elementen. Deze zijn hieronder kort beschreven en staan grafisch weergegeven in figuur 4.1.

- 'User Mode Linux' (UML) omgeving
  Het VCSL maakt voor de virtualisatie van de virtuele machines in een virtueel netwerk gebruik van een NetKit omgeving die gebaseerd is op UML [UML15; PR08]. Deze virtuele machines noemen we in het vervolg UML-hosts en de virtuele netwerken noemen we in het vervolg UML-netwerken. De UML-hosts en het UML-netwerk waar deze UML-hosts gebruik van kunnen maken, zijn compleet geïsoleerd en kunnen alleen maar verbindingen met elkaar maken. Voor dit onderzoek maken we gebruik van de nieuwste generatie van NetKit, NetKit-ng [Igu15].

- Virtuele host machine
  De virtuele host machine is een machine die is voorzien van een Linux of Unix besturingssysteem om de NetKit toepassing te kunnen draaien [PR08]. In dit onderzoek noemen we deze virtuele host machine in het vervolg een VCSL node.

- Host machine
  De host machine is een willekeurige machine met een besturingssysteem naar keuze waarop virtualisatie software kan worden gedraaid om de virtuele host machine te

kunnen draaien. De virtuele host machine laag is niet nodig als het besturingssysteem de NetKit toepassing direct kan draaien zoals in het geval van de meeste Linux distributies en een enkele Unix distributie.



Figuur 4.1: Architectuur VCSL

Binnen het VCSL blijft het virtueel netwerkverkeer binnen de UML omgeving. Eerder werden er andere technieken gebruikt om grote netwerken te kunnen analyseren, zoals het simuleren van netwerken. In het begin werd het te simuleren netwerk logisch gescheiden van het andere netwerkverkeer, waarbij wel gebruik gemaakt wordt van deze fysieke netwerkinfrastructuur [Whi+02]. Bij deze techniek kan er geen onderscheid worden gemaakt tussen 'veilig' en 'onveilig' netwerkverkeer, waardoor deze minder geschikt is om hierin onveilige software te draaien of beveiligingscontroles uit te voeren.

Werking van het VCSL

Hieronder is in het kort de werking van het VCSL aangegeven op basis van de bestudeerde literatuur van het VCSL [VK09].

De UML-hosts worden gestart op een VCSL node. Deze virtuele host machine kan op bijna alle huidige besturingssystemen draaien door middel van virtualisatie software. Deze VCSL architectuur is de basis van het DVCSL en is daarmee een belangrijk kenmerk van het DVCSL waarmee rekening gehouden moet worden in dit onderzoek.

Binnen een UML omgeving kunnen meerdere UML-hosts met elkaar worden verbonden in een UML-netwerk. Zo'n UML-netwerk wordt gevormd door een UML-switch die te vergelijken is met een fysieke switch.

Een UML-host die verbonden is aan meerdere UML-switches kan als router worden geconfigureerd. Door het combineren van UML-hosts met UML-switches kunnen complexe virtuele netwerken worden gebouwd in een virtueel computer laboratorium. Op deze wijze kan er dus een compleet virtueel netwerktopologie worden gebouwd bestaande uit kleine Linux systemen die kunnen fungeren als een node, maar ook als router [CB08]. Elke UML-host heeft de zelfde mogelijkheden als een fysiek Linux systeem, maar heeft veel minder geheugen en diskruimte nodig.

AUTONETKIT

Een andere discipline is het genereren van grote netwerkomgevingen in een virtuele netwerkomgeving. Het opzetten van grote netwerkomgevingen en het onderhoud hiervan is arbeidsintensief. Als deze taken geautomatiseerd kunnen worden kan hiermee veel kostbare tijd bespaard worden. Omdat de meeste van deze omgevingen niet fysiek zijn, maar in software zijn geïmplementeerd, is het mogelijk deze op een geautomatiseerde manier aan te maken. Voor onder andere NetKit is hiervoor AutoNetKit ontwikkeld [PR08; Ngu+10]. Hiermee kan op basis van een grafische representatie van een netwerktopologie, automatisch de configuratie voor de te testen netwerkinfrastructuur worden gemaakt.

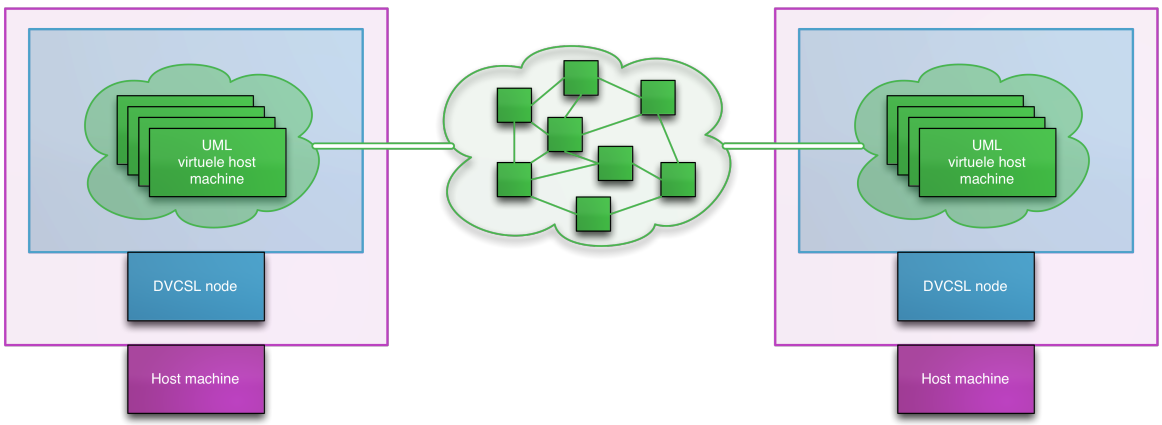
Met behulp van een grafische representatie van een netwerk, kan met behulp van AutoNetKit een NetKit omgeving worden gegenereerd [Ngu+10].

Bij een eerste analyse van dit hulpmiddel is gebleken dat dit hulpmiddel alleen geschikt is voor het maken van grote virtuele netwerken in één VCSL omgeving. Om deze geschikt te maken moeten de router nodes worden voorzien van extra functionaliteit om gebruikt te kunnen worden in het DVCSL. Hierbij is de keuze gemaakt om dit hulpmiddel hiervoor niet uit te breiden omdat het geen toegevoegde waarde heeft voor dit onderzoek.

4.2.2. ARCHITECTUUR DVCSL

De architectuur van het DVCSL voegt aan de architectuur van het VCSL toe dat meerdere VCSL omgevingen met elkaar gekoppeld kunnen worden. Dan wel rechtstreeks of middels een centrale autoriteit [Vra+11; Haa+11]. Een DVCSL omgeving bestaat uit minimaal twee gekoppelde VCSL omgevingen [Vra+11]. Op het moment dat een VCSL node is uitgebreid met deze technologie noemen we het een DVCSL node.

Door meerdere VCSL omgevingen aan elkaar te koppelen ontstaat er één DVCSL omgeving. Dit wordt gedaan door twee UML-switches van twee verschillende VCSL omgevingen met elkaar te verbinden. Voor een algemene representatie van de DVCSL architectuur, zie figuur 4.2.



Figuur 4.2: Architectuur DVCSL

Aangetoond is dat deze gekoppelde UML-switches zich gedragen als één logische UML-switch en dus één logische UML virtuele netwerkomgeving. Hierdoor kan een UML-host

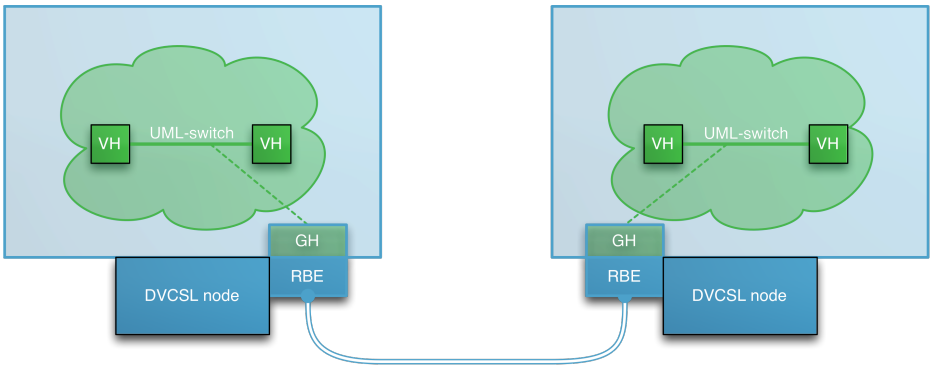
in de ene DVCSL node een andere UML-host bereiken op een andere DVCSL node. En op deze wijze kunnen virtuele netwerken worden gemaakt die over meerdere DVCSL nodes zijn verspreid. De logische UML virtuele netwerkomgeving is in figuur 4.2 in het groen aangegeven. De DVCSL nodes zijn in het blauw aan gegeven en in het paars zijn de host machines aangegeven.

De basis van het DVCSL is het handmatig opzetten van een verbinding tussen de UML-switches die draaien op de DVCSL nodes [Vra+11], of middels de centrale autoriteit die deze verbindingen automatisch opzet [Haa+11]. Het opzetten van een verbinding wordt gedaan door op de twee DVCSL nodes een 'plug' proces te starten.

Tijdens het bestuderen van de broncode van het DVCSL is naar voren gekomen dat het gebruik van een centrale autoriteit minder geschikt is voor dit onderzoek. De centrale autoriteit heeft een groot voordeel voor het opzetten van de verbindingen tussen de verschillende DVCSL nodes. Daarnaast wordt al het netwerkverkeer naar de centrale autoriteit gestuurd [Haa+11]. Voor dit onderzoek worden er maar drie VCSL omgevingen gebruikt en in een vaste configuratie. Daarmee is het voordeel van het opzetten van de verschillende verbindingen minimaal. Het geprogrammeerd starten van de communicatie processen is een goed alternatief hiervoor. Maar het gebruik van de centrale autoriteit heeft mogelijk een nadeel, omdat al het netwerkverkeer naar deze centrale autoriteit gestuurd wordt en dit mogelijk een bottleneck kan vormen in het netwerkverkeer. Hierdoor is de keuze gemaakt om voor dit onderzoek geen gebruik te maken van de centrale autoriteit en is dit een onderwerp voor vervolgonderzoek.

Het 'plug' proces maakt een verbinding met de UML-switch zoals een UML-host dat ook doet. Dit deel van het 'plug' proces wordt 'Ghost Host' (GH) genoemd. Een GH is in feite een softwarematige UML-host die verbonden is met een UML-switch. Via de GH kan het virtuele netwerkverkeer worden onderschept of worden toegevoegd aan de lokale UML-switch. Een ander onderdeel van het 'plug' proces is de 'Remote Brige Endpoint' (RBE). Hiermee wordt het virtuele netwerkverkeer ingepakt in 'User Datagram Protocol' (UDP) pakketjes en verstuurd naar de RBE op de andere DVCSL node. Op de andere DVCSL node worden deze UDP pakketjes weer uitgepakt en wordt het virtuele netwerkverkeer door de RBE en via de GH toegevoegd aan de UML-switch.

Hierdoor vormen de beide UML-switches één logische UML-switch die het virtuele netwerkverkeer koppelt op het niveau van de datalinklaag, zie figuur 4.3.



Figuur 4.3: DVCSL UML-switch

Het uitbreiden van de mogelijkheden van het DVCSL is geen onderdeel van dit onderzoek.

4.2.3. TOPOLOGIE DVCSL

In de topologie van het DVCSL mogen geen gesloten cirkels voorkomen, omdat anders het DVCSL verkeer oneindig lang rond gestuurd gaat worden [Haa+11]. In een boomstructuur komen geen cirkels voor en daarom is voor dit onderzoek gekozen om de drie VCSL omgeving aan elkaar te koppelen middels een boomstructuur. Er zijn ook andere structuren mogelijk door gebruik te maken van geavanceerde routeringstechnieken, maar dit is geen onderdeel van dit onderzoek en een onderwerp voor een vervolgonderzoek.

Met drie DVCSL nodes is het mogelijk om de UML omgeving dynamisch te laten verkleinen door een DVCSL node te stoppen en daarna weer dynamisch te laten groeien door deze DVCSL node weer te starten. Hiermee kan de dynamiek die op het echte Internet bestaat goed worden gesimuleerd en kan worden gebruikt voor het analyseren van bepaalde levensfasen van botnets.

Binnen dit onderzoek wordt maar beperkt gebruik gemaakt van de schaalbaarheid van het DVCSL. Zo is een VCSL en daarmee ook het DVCSL verticaal schaalbaar door de capaciteit van de fysieke machine waarop een DVCSL node draait te vergroten. Hierdoor kunnen er meer UML-hosts worden gedraaid op één DVCSL node. Maar het DVCSL is ook horizontaal schaalbaar omdat er dynamische extra DVCSL nodes kunnen worden toegevoegd. Door meer DVCSL nodes toe te voegen aan de DVCSL omgeving kunnen er ook meer fysieke machines worden toegevoegd aan de DVCSL omgeving waardoor de machine capaciteit wordt vergroot.

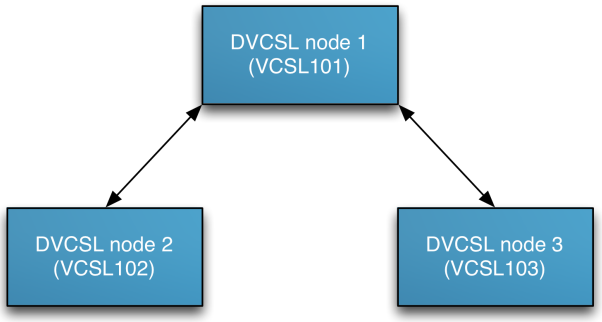
Voor dit onderzoek zijn hiervoor drie Fedora Linux systemen geïnstalleerd en geconfigureerd met NetKit-ng, VCSL101, VCSL102 en VCSL103 [Igu15]. De keuze voor Fedora Linux is gebaseerd op de argumenten dat het een vrij toegankelijke Linux distributie is en dat hieraan nog steeds actief aan wordt ontwikkeld door een grote Linux distributeur.

Hieronder een korte omschrijving van de drie DVCSL nodes :

- DVCSL-node-1 (VCLS101)
  Dit is het VCSL systeem dat dient als root node van de DVCSL topologie. Hierop draaien twee communicatieprocessen. Één die een verbinding maakt naar de VCSL102 en één die een verbinding maakt naar de VCSL103.

- DVCSL-node-2 (VCLS102) en DVCSL-node-3 (VCLS103)
  Dit zijn VCSL systemen die gekoppeld zijn aan de root node van de DVCSL topologie. Hierop draait één communicatieproces per node die een DVCSL verbinding maakt naar de VCSL101.

De implementatie van deze topologie is weergegeven in een virtueel netwerktopologie, zie figuur 4.4. Ook deze opstelling heeft zijn mogelijke beperkingen omdat het netwerkverkeer dat van DVCSL-node-2 naar DVCSL-node-3 via DVCSL-node-1 moet worden verstuurd en visa versa.



Figuur 4.4: DVCSL Netwerktopologie

4.3. DVCSL OMGEVINGEN

Met de DVCSL omgevingen worden aangegeven op welke wijze de verschillende DVCSL nodes met elkaar verbonden zijn. Hieronder zijn er twee voorbeeld omgevingen weergegeven die tijdens dit onderzoek zijn gebruikt.

Zoals in figuur 4.4 is aangegeven bestaat de DVCSL omgeving binnen dit onderzoek uit drie DVCSL nodes. Deze drie DVCSL nodes vormen samen één UML virtuele netwerkomgeving, deze is in het groen aangegeven in figuur 4.2. Alle UML-hosts zijn verbonden met dezelfde logische virtuele UML-switch. Hierdoor kunnen ze vrijelijk met elkaar communiceren op het datalink niveau omdat ze zijn aangesloten op dezelfde virtuele netwerk switch. Elke UML-host heeft een vast IP-adres waarmee deze bereikbaar is voor de andere UML-hosts. Alle virtuele UML-host nodes zijn aan elkaar gelijk en hebben geen specifieke taken en er zijn geen specifieke services geactiveerd.

De verschillende DVCSL nodes binnen een DVCSL omgeving moeten elkaar kunnen bereiken op hun IP-adressen. Het lokale virtuele netwerkverkeer van de RBE wordt gestuurd naar de andere RBE middels het IP-adres van de andere DVCSL node en visa versa. Dit noemen we het DVCSL verkeer. Het DVCSL verkeer maakt gebruik van het UDP netwerkprotocol, zoals in de vorige paragraaf is beschreven. Naast dat de DVCSL nodes elkaar moeten kunnen bereiken op hun IP-adressen, moeten de gebruikte UDP poorten ook bereikbaar zijn.

Op de meeste Linux systemen zijn deze poorten afgesloten en wordt het netwerkverkeer naar deze poorten niet toegelaten. Deze vorm van softwarematige toegangsbeperking op netwerkniveau wordt een firewall genoemd. De firewall moet hiervoor worden aangepast zodat deze het DVCSL verkeer wel door laat.

Als het DVCSL verkeer niet mogelijk is, of doordat de DVCSL nodes elkaar niet kunnen bereiken, of doordat het DVCSL verkeer niet wordt toegelaten, dan kan het virtuele netwerkverkeer ook niet worden uitgewisseld tussen de betreffende DVCSL nodes. Hierdoor kan er geen volledige DVCSL omgeving worden gevormd.

Voor de technische details van de onderstaande paragrafen, wordt er verwezen naar Bijlage A ‘DVCSL Configuratie’. De paragrafen van deze bijlage komen overeen met de namen van de paragrafen in dit hoofdstuk.

4.3.1. LOKALE DVCSL OMGEVING

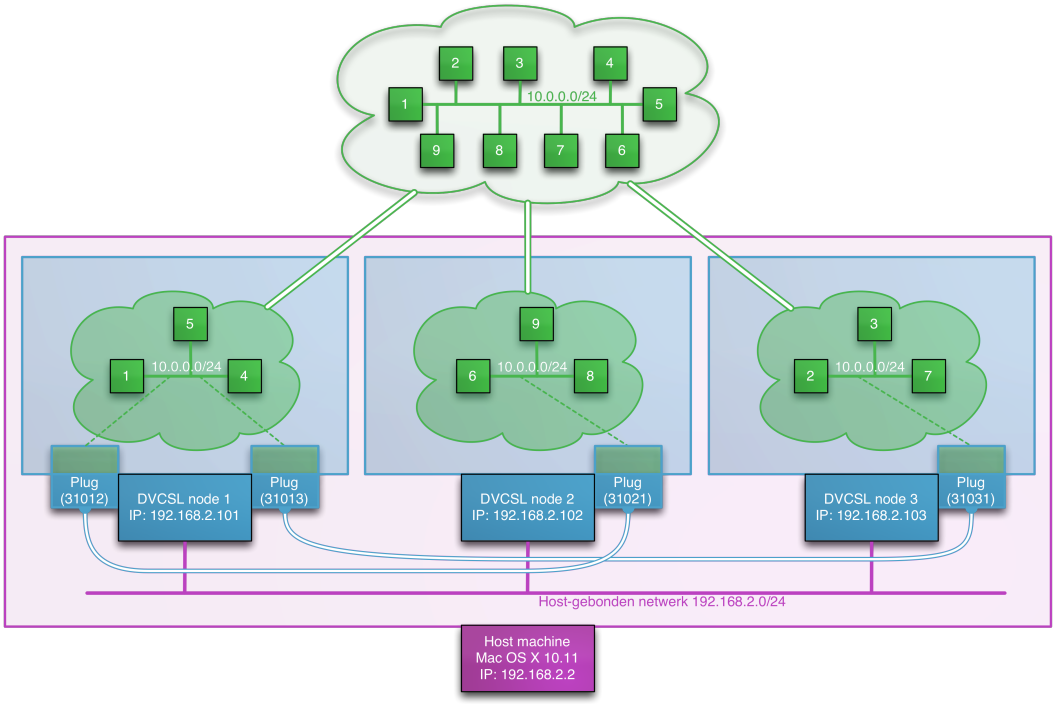
De lokale DVCSL omgeving bestaat uit drie DVCSL nodes. Deze drie DVCSL nodes zijn volledig identiek aan elkaar behalve dat ze een andere hostnaam hebben met een ander IP-adres. De drie DVCSL nodes zijn met elkaar verbonden middels een host machine gebonden netwerk, zie figuur 4.5.

Zoals in figuur 4.2 in het groen de UML virtuele netwerkomgeving is aangegeven, is deze in figuur 4.5 ook in het groen aangegeven. In het bovenste groene wolk is de logische UML virtuele netwerkomgeving weergegeven. In de drie groene wolkjes zijn de drie losse UML virtuele netwerkomgevingen weergegeven. De blauwe vierkanten zijn de drie DVCSL nodes en de paarse rechthoek is de fysieke machine waarin deze DVCSL omgeving draait.

Deze DVCSL omgeving is met name gebruikt tijdens het uitvoeren van dit onderzoek omdat alle DVCSL nodes op één fysieke machine gedraaid kunnen worden, met als voordeel dat er maar één machine beschikbaar hoeft te zijn.

Deze lokale DVCSL omgeving is te gebruiken als er geen andere fysieke machines beschikbaar zijn of als er geen connectie gemaakt kan worden met andere machines. Omdat alle DVCSL nodes op één fysieke machine draaien is deze DVCSL omgeving niet uit te breiden met extra machine capaciteit. Deze DVCSL omgeving is dus alleen maar verticaal schaalbaar.

Voor de technische details van de lokale DVCSL omgeving zie paragraaf A.3.1. Hierin wordt uitgelegd hoe deze lokale DVCSL omgeving geconfigureerd, gestart en gestopt kan worden.



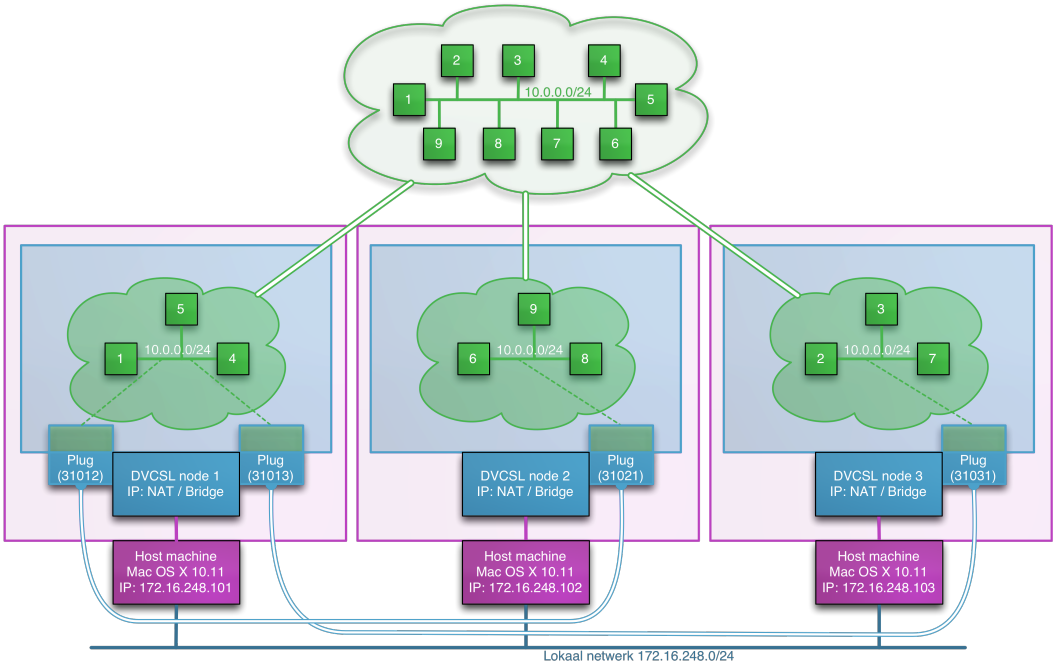
Figuur 4.5: DVCSL Basis omgeving lokaal

4.3.2. Remote DVCSL omgeving

De remote DVCSL omgeving bestaat uit drie DVCSL nodes en deze zijn volledig identiek aan elkaar behalve dat ze op drie verschillende host machines draaien, zie figuur 4.6. Deze systemen moeten elkaar wel kunnen bereiken op de gebruikte UDP poorten.

Bij deze configuratie vormen de drie DVCSL nodes samen één DVCSL omgeving, deze zijn in het paars weergegeven in de figuur. Het netwerk van de drie DVCSL nodes moet zo zijn geconfigureerd dat deze DVCSL nodes middels het extern gedeeld netwerk bereikbaar zijn, anders is er geen DVCSL verkeer mogelijk. Dit kan worden geconfigureerd via de NAT optie of de Bridge optie. Deze DVCSL omgeving is dus naast verticaal schaalbaar ook horizontaal schaalbaar. Door het toevoegen van een DVCSL node wordt extra machine capaciteit toegevoegd.

Voor de technische details van de remote DVCSL omgeving zie paragraaf A.3.2. Hierin wordt uitgelegd hoe deze remote DVCSL omgeving geconfigureerd, gestart en gestopt kan worden.



Figuur 4.6: DVCSL Basis omgeving remote

4.3.3. Netwerkconfiguratie

Binnen een DVCSL omgeving kunnen er vele services worden gebruikt en geïmplementeerd. Hierbij is de keuze gemaakt om voor dit onderzoek het aantal te gebruiken services tot een minimum te beperken.

Op basis van de bestudeerde wetenschappelijke literatuur zijn er twee services gevonden die eigenlijk altijd worden gebruikt bij de simulatie van botnets. Dit zijn 'Domain Name System' (DNS) en 'Dynamic Host Configuration Protocol' (DHCP) [Cal+10; BB07]. Deze services moeten zeker worden geïmplementeerd in het DVCSL. Ook wordt het vaak 'Simple Mail Transfer Protocol' (SMTP) genoemd, maar deze is vaak botnet specifiek en hoeft daarom niet op voorhand te worden geïmplementeerd.

Op dit moment is 'Internet Protocol versie 6' (IPv6) behoorlijk in opkomst, maar de basis is nog vaak 'Internet Protocol versie 4' (IPv4). Gezien de complexiteit van het IPv6 protocol blijft dit onderzoek zoveel mogelijk beperkt tot het gebruik van IPv4. Als een botnet alleen werkt op basis van IPv6 dan zal er een beperkte IPv6 implementatie moeten worden geconfigureerd.

De DNS netwerkservice is te vergelijken met een digitale telefoonboek van het Internet, waarin een groot aantal systeemnamen zijn opgeslagen en waarbij het IP-adres is op geslagen. Hierdoor kan op basis van het IP-adres er een hostname bij worden gezocht, maar ook op basis van de hostnaam het IP-adres worden opgezocht. Deze informatie staat niet op één grote server op het Internet maar is verdeeld op basis van de verschillende domein namen. Dit geldt niet alleen voor de hostnaam maar ook voor het IP-adres.

In tabel 4.1 DVCSL omgevingen is een voorbeeld configuratie weergegeven van de DNS namen en IP-adressen van zowel de lokale als van de remote omgeving zoals deze zijn gebruikt voor dit onderzoek. Wat hierbij van belang is, is dat de juiste IP-adressen bij de juiste DNS namen zijn geconfigureerd en dat deze op alle drie DVCSL nodes hetzelfde zijn. Dit kan middels een centrale DNS server maar ook via een lokale configuratie.

Tabel 4.1: DVCSL omgevingen

	Configuratie	**Volledige hostnaam**	**IP-adres**
DVCSL-node-1	Lokaal	vcsl101.local.dvcsl.edu	192.168.2.101
(VCLS101)	Remote	vcsl101.remote.dvcsl.edu	172.16.248.101
DVCSL-node-2	Lokaal	vcsl102.local.dvcsl.edu	192.168.2.102
(VCLS102)	Remote	vcsl102.remote.dvcsl.edu	172.16.248.102
DVCSL-node-3	Lokaal	vcsl103.local.dvcsl.edu	192.168.2.103
(VCLS103)	Remote	vcsl103.remote.dvcsl.edu	172.16.248.103

De DHCP netwerkservice is te vergelijken met een telefoon provider van het Internet. Een telefoon provider geeft telefoonnummers uit en zorgt er voor dat je kan bellen. Dat doet een DHCP netwerkservice ook voor het Internet. Deze geeft namelijk IP-adressen uit waarmee kan worden gecommuniceerd met andere systemen. Daarnaast kan een DHCP service ook bepaalde configuraties automatisch beschikbaar maken, zoals de DNS netwerkservice.

Voor het configureren van de host machines is er geen gebruik gemaakt van een DHCP server, maar zijn de IP-adressen statisch geconfigureerd. Bij een standaard DHCP configuratie worden de IP-adressen dynamische uitgegeven en kunnen de gebruikte DVCSL nodes iedere keer een ander IP-adres toegewezen krijgen. Hierdoor moet de DVCSL netwerkconfiguratie bij iedere IP-adres verandering hier op aangepast worden. Binnen de UML virtuele netwerkomgeving wordt wel gebruik gemaakt van een DHCP netwerkservice, dit wordt in de volgende paragraaf verder uitgewerkt.

De DVCSL nodes die gebruikt zijn in dit onderzoek zijn voorzien van een eigen DNS service met allemaal dezelfde DNS configuraties. Voor beide omgevingen is er een aparte subdomein onder het 'dvcsl.edu' domein geconfigureerd, om zo middels de DNS de juiste DNS namen en de bijbehorende IP-adressen te kunnen opvragen.

4.4. DVCSL CONFIGURATIES

Met de DVCSL configuratie wordt aangegeven op welke wijze de NetKit omgevingen zijn geconfigureerd. De DVCSL configuratie bestaat uit een aantal losse configuratie sets. Voor elke DVCSL node is er één configuratie directory waarmee een NetKit omgeving kan worden geconfigureerd. Deze losse configuratie sets vormen samen één geheel, de DVCSL configuratie. Voor de technische details, zie paragraaf A.1.2. Ook staat hierin beschreven hoe de onderstaande DVCSL configuraties geactiveerd, gestart, gestopt en getest kunnen worden.

Zoals hierboven is beschreven zijn er voor dit onderzoek twee DVCSL omgevingen ingericht, een lokale en een remote DVCSL omgeving. Welke van deze twee omgevingen gebruikt wordt, maakt voor de DVCSL configuraties niet uit. Waar wel rekening mee gehouden moet worden is dat alle drie de DVCSL nodes deel uit maken van dezelfde DVCSL omgeving en dus elkaar moeten kunnen bereiken op de gebruikte IP-adressen en UDP poorten.

Een ander punt waar mogelijk rekening mee gehouden moet worden is de startvolgorde van de drie losse DVCSL configuratie delen. Bij de verschillende DVCSL configuraties hieronder wordt aangegeven of er een startvolgorde moet worden aangehouden voor de drie DVCSL nodes. Als basis kan worden gehanteerd om eerste de configuratie voor DVCSL-node-1 te starten en daarna de configuratie op andere DVCSL nodes te starten. De reden hiervoor is dat de DVCSL verbindingen allemaal lopen via DVCSL-node-1 en er dus geen direct contact tussen de andere DVCSL nodes mogelijk is.

4.4.1. DVCSL BASIS CONFIGURATIE

Het doel van de basis configuratie is het testen van de werking van de drie losse DVCSL nodes. Dit kan worden gedaan door bijvoorbeeld een pingtest tussen de verschillende UML-hosts. In tabel 4.2 DVCSL basis UML nodes is een overzicht gegeven van de gebruikte UML-hosts in deze DVCSL basis configuratie. Alle tabellen voor de DVCSL configuraties hebben dezelfde indeling. Zodat deze gemakkelijk met elkaar te vergelijken zijn.

Een voorbeeld van de mogelijk testen zijn een pingtest van pc101a naar pc10b en van pc102a naar pc102b en van pc103a naar pc103b. Maar ook kan hiermee een pingtest van de ene DVCSL node naar de andere DVCSL node worden getest, dus van bijvoorbeeld pc101a naar pc102a en pc103b. Maar ook kan de connectie over de root DVCSL node worden getest door een pingtest van pc103a naar pc102b.

Als al deze verbindingen mogelijk zijn dan is er een werkende DVCSL omgeving waarbij alle UML-hosts vrijelijk met elkaar kunnen communiceren.

Voor deze DVCSL basis configuratie maakt de startvolgorde niet uit omdat deze DVCSL configuratie volledig statisch is en er geen gebruik gemaakt wordt van gedeelde netwerkservices.

Deze DVCSL basis configuratie is als uitgangspunt genomen voor alle andere gebruikte DVCSL configuraties, waaraan steeds nieuwe functionaliteiten zijn toegevoegd.

Tabel 4.2: DVCSL basis UML nodes

	UML-host	IP-adres	Hostnaam (*.t75317.dvcsl.edu.)	Netwerkservices
DVCSL-node-1	pc101a	10.0.0.111	-	-
	pc101b	10.0.0.112	-	-
DVCSL-node-2	pc102a	10.0.0.121	-	-
	pc102b	10.0.0.122	-	-
DVCSL-node-3	pc103a	10.0.0.131	-	-
	pc103b	10.0.0.132	-	-

4.4.2. DVCSL SERVICES CONFIGURATIES

Binnen een fysiek netwerk zijn er vaak een aantal algemene netwerkservices beschikbaar zoals DNS, DHCP en SMTP. Bij de afbakening van dit onderzoek is hiervan aangegeven dat minimaal DNS en DHCP geïmplementeerd moeten worden.

Naast de basis DVCSL configuratie zijn er nog drie extra DVCSL services configuraties gemaakt. In de volgende drie paragrafen worden deze drie DVCSL configuraties beschreven en hoe daarin de betreffende netwerkservices zijn geïmplementeerd. Eerst zijn er twee losse netwerkservices gemaakt. Een configuratie waarin alleen de DNS netwerkservice is geïmplementeerd en een configuratie waarin alleen de DHCP netwerkservice is geïmplementeerd. Als laatste een DVCSL configuratie waarin beide netwerkservices zijn gecombineerd.

DNS SERVICE

Voor deze DVCSL configuratie is gekozen om voor de UML-hosts het 't75317.dvcsl.edu' domein te gebruiken. Dit is een fictieve domeinnaam en is op het Internet niet bekend. De reden hiervoor is tweeledig. Als eerste omdat je niet zonder toestemming iemand anders zijn domeinnaam zomaar kunt gebruiken. Ten tweede als er een connectie gemaakt wordt op welke wijze ook buiten de UML virtuele netwerkomgeving dan heeft dat geen invloed buiten deze UML omgeving.

Om gebruik te kunnen maken van dit domein is er een DNS server UML-host toegevoegd. Om niet voor alle netwerkservices een speciale UML-host op te nemen, is er een centrale UML-host toegevoegd en hebben deze router genoemd. Deze router UML-host is zo geconfigureerd dat deze zelf de gegevens beheert van het 'dvcsl.edu' domein en het '0.0.10.in-addr.arpa' domein. Het 'dvcsl.edu' domein wordt gebruikt voor het opvragen van het IP-adres van de UML hostnaam en het '0.0.10.in-addr.arpa' domein wordt gebruikt voor het opvragen van de UML hostnaam van een IP-adres.

Daarnaast is het mogelijk om IP-adressen en hostnamen op te vragen die op het Internet bestaan. Deze verzoeken worden via de DVCSL node doorgestuurd naar de host machine. Beide systemen moeten hiervoor geschikt worden gemaakt. Dit valt buiten dit onderzoek.

De aanpassing die gedaan is op de DVCSL basis configuratie van paragraaf 4.4.1 is het toevoegen van een router UML-host die een interne netwerkverbinding heeft met de DVCSL

node. Hiermee kan ook eventueel DNS bevragingen richting het echte Internet worden gedaan en verbonden is met het interne UML netwerk. Deze router UML-host biedt de DNS service aan op dit virtuele UML netwerk, waarmee de andere UML-hosts gebruik kunnen maken van de DNS netwerkservice.

Alle bevragingen van het 'dvcsl.edu' domein en het '0.0.10.in-addr.arpa' domein worden door de DNS service op de UML router gegeven. Bevragingen voor alle andere domeinen kunnen worden doorgestuurd naar de DVCSL node, waarbij de UML router fungeert als een DNS proxy. De UML omgeving blijft hiermee volledig afgesloten, alleen worden de niet te beantwoorden DNS verzoeken uitgezet richting de DVCSL node. Als deze DVCSL node niet verbonden is met het echte Internet, of niet juist is geconfigureerd, dan blijven deze bevragingen onbeantwoord. Het configureren van deze koppeling van de DVCSL node en de host machine valt buiten dit onderzoek.

Alle UML-hosts in deze DVCSL configuratie zijn zo ingesteld dat deze automatisch gebruik maken van de DNS netwerkservice die op de router UML-host draait. Een overzicht van de gebruikte UML-hosts is in de tabel 4.3 DVCSL DNS UML nodes weergegeven.

Tabel 4.3: DVCSL DNS UML nodes

	UML-host	**IP-adres**	**Hostnaam** (*.t75317.dvcsl.edu.)	**Netwerkservices**
DVCSL-node-1	router	10.0.0.1	server01	DNS server
		192.168.250.2		TAP-netwerk 192.168.250.1
	pc101a	10.0.0.111	pc111	DNS client
	pc101b	10.0.0.112	pc112	DNS client
DVCSL-node-2	pc102a	10.0.0.121	pc121	DNS client
	pc102b	10.0.0.122	pc122	DNS client
DVCSL-node-3	pc103a	10.0.0.131	pc131	DNS client
	pc103b	10.0.0.132	pc132	DNS client

DHCP SERVICE

De basis configuratie van paragraaf 4.4.1 is uitgebreid met een DHCP server UML-host, die zo is geconfigureerd dat deze voor het UML virtueel netwerk de IP-adressen uit geeft. De IP-adressen die automatisch worden uitgegeven beginnen vanaf '10.0.0.100' en gaan door tot en met '10.0.0.250'. Om dezelfde reden als hiervoor is aangegeven gebruiken we hiervoor de router UML-host.

Alle UML-hosts in deze DVCSL configuratie zijn zo ingesteld dat deze automatisch gebruik maken van de DHCP netwerkservice van de router UML-host en dus automatisch worden voorzien van een IP-adres. Het is niet van te voren te bepalen welke UML-host welk IP-adres krijgt. Dat heeft onder andere met de startvolgorde te maken. Hierbij is het wel belangrijk dat de verbinding naar de root DVCSL node is gelegd. Hiervoor is er een mechanisme gemaakt dat de UML-hosts wachten totdat deze verbinding is gelegd en daarna pas de DHCP netwerkservice gaan gebruiken.

Een nadeel van een DVCSL netwerkservice binnen de UML virtuele netwerkomgeving is dat de UML-hosts niet iedere keer hetzelfde IP-adres krijgt waarbij het moeilijker is om een individuele UML-host over meerdere herstarts van de DVCSL omgeving te volgen. Het belangrijkste voordeel is dat niet iedere UML-host een eigen configuratie hoeft te hebben maar gebruik kan maken van een soort template. Hierdoor is het gemakkelijker geworden om nieuwe UML-hosts aan een configuratie-set toe te voegen. Bijvoorbeeld als er meer machine capaciteit beschikbaar is gekomen.

Belangrijk voor een goed werkende DHCP configuratie binnen de UML omgeving is dat de root DVCSL node als eerste operationeel is, voordat de UML omgevingen op andere DVCSL nodes operationeel gemaakt worden. Een overzicht van de gebruikte UML-hosts is in de tabel 4.4 DVCSL DHCP UML nodes weergegeven.

Tabel 4.4: DVCSL DHCP UML nodes

	UML-host	**IP-adres**	**Hostnaam** (*.t75317.dvcsl.edu.)	**Netwerkservices**
DVCSL-node-1	router	10.0.0.1	-	DHCP server
	pc101a	DHCP	-	DHCP client
	pc101b	DHCP	-	DHCP client
DVCSL-node-2	pc102a	DHCP	-	DHCP client
	pc102b	DHCP	-	DHCP client
DVCSL-node-3	pc103a	DHCP	-	DHCP client
	pc103b	DHCP	-	DHCP client

DNS EN DHCP SERVICES

De uiteindelijke DVCSL service configuratie waarin zowel de DNS netwerkservice als de DHCP netwerkservice zijn samengevoegd, dient als basis voor de botnetsimulaties van Hoofdstuk 5 'Botnetsimulatie'. In feite is deze DVCSL configuratie een samenvoeging van de twee bovenstaande DVCSL configuraties, waarbij de router UML-host is voorzien van zowel de DNS als de DHCP netwerkservice.

De configuratie van de DHCP netwerkservice is uitgebreid zodat deze ook automatisch de DNS netwerkservice bekent maakt bij de UML-hosts. De DNS netwerkservice is uitgebreid zodat deze ook de juiste IP-adressen en hostnamen geeft bij de automatisch uitgegeven IP-adressen van de DHCP netwerkservice.

Alle andere UML-hosts zijn hier verder niet aangepast in vergelijking met de DVCSL DHCP configuratie. Alleen krijgen ze nu ook automatisch de juiste configuratie voor de DNS netwerkservice. Een overzicht van de gebruikte UML-hosts is in de tabel 4.5 DVCSL DHCP - DNS UML nodes weergegeven.

Tabel 4.5: DVCSL DHCP - DNS UML nodes

	UML-host	IP-adres	Hostnaam (*.t75317.dvcsl.edu.)	Netwerkservices
DVCSL-node-1	router	10.0.0.1	server01	DNS/DHCP server
		192.168.250.2		TAP-netwerk 192.168.250.1
	pc101a	DHCP	pc*	DHCP client
	pc101b	DHCP	pc*	DHCP client
DVCSL-node-2	pc102a	DHCP	pc*	DHCP client
	pc102b	DHCP	pc*	DHCP client
DVCSL-node-3	pc103a	DHCP	pc*	DHCP client
	pc103b	DHCP	pc*	DHCP client

4.5. DVCSL PROBLEMEN EN OPLOSSINGEN

Tijdens het onderzoek naar de werking van het DVCSL zijn er drie problemen gevonden:

- Een hoge processor belasting van de host machine als er een DVCSL omgeving actief is
- Netwerkpakket fragmentatie op host machine bij een flood-ping met een grotere pakket grootte
- Virtuele netwerkverkeer kan verloren raken bij zware netwerkbelasting

In de twee volgende paragrafen worden deze toegelicht en in het kort beschreven hoe deze zijn opgelost of waar rekening mee gehouden moet worden.

4.5.1. PROCESSOR BELASTING HOST MACHINE

Tijdens het configureren van de DVCSL basis configuratie van paragraaf 4.4.1 op een lokale DVCSL omgeving, was er een probleem ontstaan met de processor belasting van de host machine. Als alle drie DVCSL nodes op één host machine draaien en de basis configuratie wordt hierin gestart, is de processor van de host machine voor 100% belast, zonder dat er iets gedaan wordt in de UML omgeving.

De DVCSL-node-1 belast de host machine voor 50% en de DVCSL-node-2 en DVCSL-node-3 voor elke 25%. Hierbij is gekeken naar de processor belasting in DVCSL-node-1, hierbij is vastgesteld dat deze belasting wordt veroorzaakt door de ‘plug’ processen die voor de DVCSL verbindingen zorgen tussen de verschillende DVCSL nodes.

Dit is natuurlijk niet de bedoeling en is ten dele opgelost door een update die in een latere DVCSL implementatie was toegepast [Haa+11]. Hierbij is de processor belasting van het ‘plug’ proces met de helft terug gebracht.

Het probleem had te maken met het GH programma deel van het plug proces. Deze maakte niet goed gebruik van de wachtstatus van de UML-switch. Hierdoor wordt er continu naar

een nieuw netwerkpakket gevraagd ook als er geen nieuw netwerkpakket klaar staat. Dit probleem is opgelost door op een andere manier van deze wachtstatus gebruik te maken, waardoor er alleen naar een nieuw netwerkpakket wordt gevraagd als er ook daadwerkelijk een netwerkpakket klaar staat.

Maar hiermee is het probleem alleen voor de helft opgelost. Door het toepassen van deze update op het RBE programma deel van het plug proces, is het probleem helemaal opgelost. Ook hier werd op een verkeerde manier gewacht op een binnenkomend netwerkpakket. Door dezelfde oplossing ook toe te passen op de binnenkomende netwerkpakketten is ook dit probleem opgelost.

Na deze twee aanpassingen was de processor belasting op de host machine volledig verdwenen. Op het moment dat er heel veel virtueel netwerkverkeer in de DVCSL ontstaat, bijvoorbeeld door de flood-ping tussen pc102a en pc103a in de basis DVCSL configuratie van paragraaf 4.4.1, dan gaat de processor belasting uiteraard wel behoorlijk omhoog. Als hierbij een tweede flood-ping sessie wordt gestart kan de host processor belasting wel oplopen tot 100%. Maar dit is te verklaren en kan gezien worden als normaal gedrag van het DVCSL.

Voor de technische gegevens van dit probleem en de gebruikte oplossingen zie paragraaf A.4.1.

4.5.2. NETWERKPAKKET FRAGMENTATIE OP HOST MACHINE

Tijdens de analyse van de werking van het DVCSL tijdens een hoge UML netwerkbelasting door middel van een flood-ping met een grotere pakket grootte is gebleken dat er op het netwerkniveau van de host machine netwerkpakket fragmentatie optreed. Hierdoor wordt er niet één netwerkpakket verzonden voor één UML netwerkpakket, maar er worden twee netwerkpakketten verzonden op het niveau van de host machine.

Op zich is dit geen probleem, maar dit kan in het geval waarbij netwerkperformance belangrijk is, wel van belang zijn. Dit is geen vreemd gedrag en komt vrijwel altijd voor als er gebruik gemaakt wordt van VPN technieken. De oplossing hiervoor is het gebruik van een aangepaste 'Maximum Transmission Unit' (MTU) grootte op het 'Internet Protocol' (IP) niveau. De MTU bepaalt op het netwerkniveau hoe groot een netwerkpakket maximaal mag zijn zodat deze in zijn geheel kan worden verstuurd. Door de MTU binnen het UML virtuele netwerk iets kleiner te maken dan dat deze op het netwerkniveau van de DVCSL wordt gebruikt, dan past gevirtualiseerd netwerkpakket wel in één netwerkpakket en treedt er geen fragmentatie op het netwerkniveau van de DVCSL node.

De standaard grootte van de MTU is 1500 bytes. De maximale grootte die in de DVCSL UML netwerkomgeving gebruikt zou moeten worden voor de MTU is 1458 bytes. Dit is proefondervindelijk vastgesteld tijdens dit onderzoek, door de MTU grootte van het DVCSL netwerkverkeer aan te passen.

Voor de technische gegevens van dit probleem en de mogelijke oplossing, zie paragraaf A.4.2.

4.5.3. OPNIEUW VERZENDEN DVCSL VERKEER

Tijdens het analyseren van het virtuele netwerkverkeer, dat door het 'plug' proces wordt verzonden en verwerkt, komen tijdens een hoge virtuele netwerkbelasting soms verzend-

fouten voor. Op het moment dat een virtueel netwerkpakketje niet kan worden verzonden wordt hier niets mee gedaan. Dit is opgelost door het virtueel netwerkpakketje maximaal vijf keer proberen te verzenden. Bij zowel het versturen van een virtueel netwerkpakketje naar een andere DVCSL node, als het verzenden van een ontvangen virtueel netwerkpakketje naar de UML-switch kon deze fouten optreden. Na deze aanpassing zijn deze fouten niet meer ontdekt in het DVCSL verkeer.

Het zoekraken van netwerkpakketjes tijdens het verzenden en ontvangen is op zich geen groot probleem. vaak wordt dit herkent en wordt het betreffende netwerkpakketje opnieuw verzonden. Alleen dit kost extra tijd met de kans dat deze ook weer zoek raakt. Dit kan gevolgen hebben op de stabiliteit en snelheid van het virtuele netwerkverkeer. Hoe groot deze gevolgen waren is niet bekend. Onderzoek hierna kost veel tijd en is daarom niet uitgevoerd. Het oplossen hiervan was redelijk simpel en is daarom wel uitgevoerd.

Voor de technische gegevens van dit probleem en de oplossing, zie paragraaf A.4.3.

4.6. DVCSL BEPERKINGEN

Tijdens dit onderzoek is er een aantal beperkingen van het DVCSL naar voren gekomen, in de volgende paragrafen worden deze beperkingen verder toegelicht.

4.6.1. ARCHITECTUUR DVCSL

Zoals in paragraaf 4.2.1 is aangegeven, bestaat een DVCSL omgeving uit een NetKit omgeving met UML systemen. Hierdoor kunnen er alleen maar Linux systemen worden gevirtualiseerd in het DVCSL. Dit is een beperking waarmee rekening gehouden moet worden. Hiermee wordt er ook een beperking gelegd op de keuze vrijheid van een te kiezen botnet voor dit onderzoek.

Een andere beperking voor het virtualiseren van fysieke netwerken is de beperking qua geheugen en processorgebruik. Het geheugengebruik en de processorgebruik is gelimiteerd aan de fysieke beperking van de gebruikte host machines. Het voordeel van het DVCSL is wel dat vele DVCSL nodes kunnen worden gekoppeld en daarmee één grote DVCSL omgeving gevormd kan worden. Maar ook hierbij geldt dat er niet meer resources gebruikt kunnen worden dan dat er fysieke aanwezig is. Er is ook een grens aan het uitbreiden van een DVCSL omgeving met nieuwe DVCSL nodes waardoor er ook een grens is aan het geheugen en de processorgebruik van de DVCSL omgeving.

4.6.2. VIRTUEEL NETWERKVERKEER

Een andere beperking is de hoeveelheid netwerkverkeer dat over de virtuele verbinding verstuurd kan worden. Deze kan nooit meer zijn dan dat er over de onderliggende fysieke netwerkverbinding verstuurd kan worden. Dus als er gebruik gemaakt van een fysieke verbinding via het Internet via een mobile verbinding, dan wordt de virtuele netwerkverbinding altijd langzamer dan deze fysieke verbinding.

4.7. KENMERKEN NODES EN VERBINDINGEN

Voor de netwerkvirtualisatie van botnets in paragraaf 3.5 zijn er een aantal kenmerken gegeven van de virtualisatie van botnets. In deze paragraaf worden van deze kenmerken be-

paald of deze te simuleren zijn in het DVCSL en welke extra voorwaarden hieraan gesteld moeten worden.

4.7.1. BOTNET NODES

Voor dit onderzoek is het nodig dat botnet nodes kunnen draaien in een Linux omgeving omdat NetKit gebaseerd is op UML. Dit houdt in dat alle botnet nodes die gebaseerd zijn op het Windows platform niet geschikt zijn voor dit onderzoek.

Een consequentie hiervan is dat de meeste bestaande en operationele botnets die op het Internet actief zijn niet geschikt zijn voor dit onderzoek en daarmee ook niet geschikt voor virtualisatie in het DVCSL. Daarmee word het zoeken naar een geschikte botnet mogelijk extra complex.

Het probleem van het emuleren is dat hierbij niet de echte programmacode wordt gebruikt. Hierdoor kan bij het emuleren andere resultaten naar voren komen in vergelijking met de resultaten als de echte programmacode zou zijn gebruikt. Bij het simuleren van bijvoorbeeld netwerk systemen kan de echte programmacode wel gebruikt worden, maar wordt de hardware gevirtualiseerd. Dus een soort combinatie van emuleren en virtualiseren [Kni+13].

4.7.2. BOTNET VERBINDINGEN

Voor dit onderzoek zijn geen extra beperkingen gevonden voor de botnet verbindingen. Uit de bestudeerde literatuur is naar voren gekomen dat alle verbindingen die op het Internet mogelijk zijn, ook mogelijk zijn in een NetKit omgeving en daarmee ook in een DVCSL omgeving.

4.8. ONDERZOEKSVRAGEN

In dit hoofdstuk zijn de antwoorden gegeven op een aantal onderzoeksdeelvragen en daarmee kan ook de tweede onderzoeksvraag worden beantwoord. Deze worden in de volgende twee paragrafen toegelicht.

4.8.1. ONDERZOEKSDEELVRAGEN

Hieronder wordt een antwoord gegeven op de derde en vierde onderzoeksdeelvraag.

WELKE KENMERKEN VAN HET DVCSL ZIJN VAN BELANG VOOR DE NETWERKVIRTUALISATIE?

De kenmerken van het DVCSL die van belang zijn voor de netwerkvirtualisatie zijn aangegeven in paragraaf 4.2, bestaan uit de volgende drie kenmerken:

- Architectuur VCSL
- Architectuur DVCSL
- Topologie DVCSL

Voor de architectuur van het VCSL en het DVCSL is van belang dat de virtualisatie alleen kan worden gedaan met entiteiten die op een Linux omgeving kunnen draaien.

Voor de architectuur van het VCSL geldt dat de machine capaciteit alleen maar verticaal schaalbaar is.

Voor de architectuur van het DVCSL geldt dat de verschillende DVCSL nodes met elkaar moeten kunnen communiceren via speciale UDP poorten. Daarnaast is machine capaciteit van het DVCSL zowel verticaal als horizontaal schaalbaar.

Bij de topologie is van belang dat de virtuele netwerktopologie bestaat uit een boom structuur, bestaande uit een root DVCSL node met andere DVCSL nodes die hiermee op een of andere wijze gekoppeld zijn.

KUNNEN DE KENMERKEN VAN DE NETWERKVIRTUALISATIE VAN BOTNETS WORDEN GEVIRTUALISEERD IN HET DVCSL?

Als de kenmerken van netwerkvirtualisatie van botnets kunnen voldoen aan de kenmerken van het DVCSL zoals deze hierboven zijn aangegeven, dan kunnen die kenmerken van de netwerkvirtualisatie van botnets worden gevirtualiseerd in het DVCSL. De belangrijkste extra voorwaarde is dat een botnet moet kunnen draaien in een Linux omgeving.

Als de kenmerken, zoals deze in paragraaf 3.6 zijn beantwoord, kunnen voldoen aan de kenmerken van de vorige onderzoeksdeelvraag, dan kunnen deze kenmerken worden gevirtualiseerd in het DVCSL.

4.8.2. ONDERZOEKSVRAAG

Op basis van de antwoorden op de voorgaande onderzoeksdeelvragen kan het antwoord op de tweede onderzoeksvraag worden gegeven.

WELKE ONTWERP-PRINCIPES EN ONTWERP-CONDITIES KUNNEN WORDEN VASTGESTELD VOOR NETWERKVIRTUALISATIE IN HET DVCSL?

De kenmerken van netwerkvirtualisatie van botnets moeten voldoen aan die van het DVCSL zoals deze hierboven zijn aangegeven. Dan kunnen die kenmerken worden gevirtualiseerd in het DVCSL middels netwerkvirtualisatie. De belangrijkste extra voorwaarde die hierbij naar voren is gekomen is dat een botnet moet kunnen draaien in een Linux omgeving.

Een belangrijk ontwerp-principe is dat de netwerkvirtualisatie in het DVCSL zowel verticaal als horizontaal schaalbaar is, waardoor deze virtualisatie omgeving niet gelimiteerd is qua schaalbaarheid.

Een belangrijke ontwerp-conditie van de virtuele nodes en virtuele verbindingen zijn dat kenmerken hiervan te implementeren zijn in een Linux omgeving.

De kenmerken van deze virtuele nodes en virtuele verbindingen van een botnet moeten overeen komen met de kenmerken van de nodes en verbindingen van het DVCSL. Hierdoor moeten de botnets geschikt zijn om in een Linux omgeving te kunnen draaien.

Op deze wijze kunnen de kenmerken van netwerkvirtualisatie worden gevirtualiseerd in het DVCSL.

5

BOTNETSIMULATIE

In dit hoofdstuk wordt geanalyseerd of het mogelijk is om botnetsimulaties uit te kunnen voeren in het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) op basis van de vastgestelde kenmerken van botnets en de werking van het DVCSL zoals deze in de vorige twee hoofdstukken aan bod zijn gekomen. Als eerst wordt de zoektocht beschreven naar geschikte botnets die in een DVCSL omgeving kunnen draaien.

Daarna wordt er een keuze gemaakt uit de gevonden botnets, De gekozen botnet wordt verder geanalyseerd om te kunnen vaststellen of het mogelijk is met deze botnet de beoogde botnetsimulaties te kunnen uitvoeren voor dit wetenschappelijk empirisch onderzoek.

Vervolgens worden de uitgevoerde botnetsimulaties in het DVCSL beschreven en de resultaten hiervan geanalyseerd.

Als laatste worden de onderzoeksvragen beantwoord.

5.1. HET VINDEN VAN EEN GESCHIKTE BOTNETS

Het vinden van een geschikt botnet voor dit wetenschappelijk empirisch onderzoek heeft veel tijd gekost. Het wat vrij snel duidelijk dat het niet mogelijk was om een bestaande geschikte botnet te vinden die aan de gestelde ontwerp-principes en ontwerp-condities voldoet, zoals deze in Hoofdstuk 4 'DVCSL' aan bod zijn gekomen. De meeste bestaande botnet waren niet geschikt voor het Linux platform omdat deze alleen maar geschikt waren voor het Windows platform.

Daarnaast is het onmogelijk zonder de juiste connecties om een bestaande operationele botnet te krijgen voor dit onderzoek. Dit is ook heel goed te begrijpen uit beveiligingsoogpunt. Voor dit onderzoek is het ook geen absolute vereiste om een bestaande operationele botnet te moeten kunnen simuleren in het DVCSL. De botnet moet wel voldoen aan de minimale voorwaarden die gesteld zijn zoals deze in Hoofdstuk 3 'Botnets' aan bod zijn gekomen.

Als het niet mogelijk is om een bestaande botnet te kunnen vinden, dan blijven er nog twee mogelijkheden over:

- Het zoeken naar geschikte synthetische botnets.
- Het zelf maken van een "eigen" botnet.

HET ZOEKEN NAAR GESCHIKTE SYNTHETISCHE BOTNETS

De kenmerken van bestaande botnets zijn meestal wel redelijk duidelijk. Ze maken gebruik van een bepaalde topologie en hebben een eigen architectuur en hebben verschillende fases van de levenscyclus van een botnet. Een geschikte synthetische botnet moet ook aan de minimale voorwaarden voldoet om gebruikt te kunnen worden voor dit onderzoek. In de volgende paragraaf wordt dit verder toegelicht.

HET ZELF MAKEN VAN EEN "EIGEN" BOTNET

Het maken van een "eigen" botnet waarmee het wetenschappelijk onderzoek gedaan kan worden is haast niet mogelijk in de tijd die voor dit onderzoek is vastgesteld. Maar als er geen geschikte botnet gevonden kan worden, die aan de gestelde eisen voldoet, behoort dit wel tot de mogelijkheden. Daarbij komt het onderzoek wel onder druk te staan, omdat de ontwikkeltijd van een botnet af gaat van het werkelijke botnetsimulatie onderzoek.

5.2. SYNTHETISCHE BOTNETS

Synthetische botnets zijn botnets die niet op het Internet actief worden gebruikt. Deze synthetische botnets zijn vaak gemaakt voor wetenschappelijke/educatieve doeleinden. Vaak hebben deze synthetische botnets maar een beperkte functionaliteit en zijn daardoor ook niet zo schadelijk als de botnets die op het Internet actief zijn.

Om een geschikte synthetische botnet te vinden moet deze wel voldoen aan de minimale voorwaarden die zijn gesteld aan een geschikte botnet voor dit wetenschappelijk empirisch onderzoek. Deze kenmerken worden in de volgende paragraaf kort toegelicht aan de hand van de gestelde voorwaarden zoals deze naar voren zijn gekomen in de vorige twee hoofdstukken.

5.2.1. GESCHIKTE SYNTHETISCHE BOTNET

Voor een geschikte synthetische botnet geldt dat deze moet voldoen aan de gestelde kenmerken. Daarnaast zijn er een aantal selectiecriteria waar rekening mee gehouden dient te worden.

KENMERKEN VAN EEN GESCHIKTE SYNTHETISCHE BOTNET

De kenmerken waaraan een synthetische botnet dient te voldoen hebben te maken met de volgende kenmerken:

- Topologie
  Zoals bij de kenmerken van een geschikte botnet is aangegeven, is het wenselijk dat een synthetische botnet een van de simpelste topologie vormen heeft, waarbij de voorkeur uit gaat naar die met een centrale topologie, zie figuur 3.1.

- Levenscyclus
  Zoals bij de kenmerken van een geschikte botnet is aangegeven, moet er minimaal één levensfase onderzocht kunnen worden van de levenscyclus van een botnet, zie figuur 3.2.

SELECTIECRITERIA VAN EEN GESCHIKTE SYNTHETISCHE BOTNET

Daarnaast moet een synthetische botnet ook voldoen aan de verplichte eigenschappen van een botnet zoals deze zijn aangegeven in paragraaf 3.3.2. Voor een synthetische botnet geldt ook dat deze beschikbaar moet zijn voor dit onderzoek.

Ook zijn er nog een aantal wenselijke eigenschappen van een geschikte botnet aangegeven, deze zijn hieronder kort weergegeven:

- De synthetische botnet moet niet te complex zijn.
- De synthetische botnet moet gemakkelijk aan te passen zijn.
- Er moet voldoende informatie beschikbaar zijn over de werking en gebruik van de synthetische botnet.
- De synthetische botnet moet kunnen werken met weinig geheugen en diskruimte.

5.2.2. GEVONDEN SYNTHETISCHE BOTNETS

Voor dit onderzoek zijn er uiteindelijk twee synthetische botnets gevonden die mogelijk geschikt zijn om gebruikt te kunnen worden voor dit wetenschappelijk empirische onderzoek. Van deze twee synthetische botnets is er eerst een kleine analyse gemaakt om hiervan de meest geschikte botnet te kunnen bepalen. Dit is gedaan aan de hand van de gestelde criteria zoals deze hierboven zijn beschreven.

Voor deze eerst analyse is alleen de beschikbare broncode en de beschikbare documentatie die voor handen was gebruikt om de geschiktheid te bepalen. In de volgende twee paragrafen komen deze eerste analyses aan bod. Op basis van deze analyse wordt er één botnet gekozen om dit wetenschappelijk empirische onderzoek mee te starten.

De gevonden synthetische botnet zijn:

- De eerste gevonden synthetische botnet is gebaseerd op Java en PHP, deze staat bekend onder de naam BYOB, zie [Bég15].
- De tweede gevonden synthetische botnet is gebaseerd op PHP en Perl, deze staat bekend onder de naam HybridBot, zie [Cro15].

5.2.3. BYOB BOTNET

De BYOB botnet is een Open Source botnet die geschreven is in de programmeertalen Java en PHP. Op basis van de kenmerken die een synthetische botnet moet hebben zoals hierboven zijn aangegeven, is er een eerste analyse gedaan van de BYOB botnet. De resultaten van deze eerste analyse zijn in de volgende drie paragrafen uitgewerkt.

TOPOLOGIE BYOB

De BYOB botnet heeft een centrale topologie bestaande uit een botmaster en direct aangestuurde bots, zie figuur 3.1.

Hiermee voldoet de BYOB botnet aan de gevraagde kenmerken van de botnet topologie die hiervoor zijn vastgesteld.

LEVENSCYCLUS BYOB

Op basis van de beschikbare broncode en documentatie kan nog niet goed worden bepaald voor welke levensfases deze BYOB botnet zou kunnen voldoen. Wat wel duidelijk is dat er één aanvalsfase mee uitgevoerd kan worden.

Hiermee voldoet de BYOB botnet aan de gevraagde kenmerken voor de levenscyclus die voor een synthetische botnet zijn vastgesteld.

OVERIGE KENMERKEN BYOB

De BYOB botnet is voornamelijk geschreven in Java en PHP. Hierbij zijn de bots volledig geschreven in Java en is de 'Commando en Controle' (C&C) geschreven in PHP. De gebruikte gegevens worden opgeslagen in een MySQL database server. Hierdoor is de BYOB botnet volledig platform onafhankelijk en zou deze ook kunnen draaien in het DVCSL.

Deze BYOB botnet bestaat uit één C&C waarmee het botnet wordt aangestuurd. De besmette systemen kunnen handmatig worden voorzien van de BYOB bot. Zover uit de beschikbare broncode en documentatie kan worden opgemaakt, kan er alleen maar een mail spam aanval worden uitgevoerd. Hiermee zijn de aanvalsmogelijkheden van deze botnet mogelijk te beperkt en moeten er extra aanvalsmethodes worden toegevoegd.

5.2.4. HYBRIDBOT BOTNET

De HybridBot botnet is een Open Source, op PHP en Perl gebaseerde botnet. Op basis van de kenmerken die een synthetische botnet moet hebben zoals hierboven is aangegeven, is er een eerste analyse gedaan van de HybridBot botnet. De resultaten van deze eerste analyse zijn in de volgende drie paragrafen uitgewerkt.

TOPOLOGIE HYBRIDBOT

De HybridBot botnet heeft een centrale topologie bestaande uit een botmaster en direct aangestuurde bots, zie figuur 3.1.

Hiermee voldoet de HybridBot botnet aan de gevraagde kenmerken voor topologie die voor een synthetische botnet zijn vastgesteld.

LEVENSCYCLUS HYBRIDBOT

Op basis van de beschikbare broncode en documentatie kan met enige zekerheid worden vastgesteld dat er meerdere levensfasen gesimuleerd kunnen worden met deze HybridBot botnet.

Hiermee voldoet de HybridBot botnet aan de gevraagde kenmerken voor de levenscyclus die voor een synthetische botnet zijn vastgesteld.

OVERIGE KENMERKEN HYBRIDBOT

De HybridBot botnet is voornamelijk geschreven in PHP. De bots zijn volledig geschreven in Perl. De C&C is geschreven in PHP en maakt gebruik van een MySQL database. Een applicatie omgeving die gebruik maakt van Perl/PHP en gebruik maakt van een MySQL database met Apache als webserver staat bekent onder de naam 'Linux Apache MySQL Perl/PHP' (LAMP). Hierdoor is de HybridBot botnet platform onafhankelijk en zou deze kunnen draaien in het DVCSL.

Deze HybridBot botnet bestaat uit één C&C waarmee het botnet wordt aangestuurd. De besmette systemen kunnen worden voorzien van de HybridBot bot. Op basis van de beschikbare broncode en documentatie zijn er ook een aantal extra functies beschikbaar waarmee de botnet onderhouden kan worden. Dit is geen verplichte eigenschap van een te kiezen synthetische botnet maar maakt verdere wetenschappelijk onderzoek mogelijk op dit terrein.

5.3. GEKOZEN BOTNET

Aan de hand van de kenmerken en de eigenschappen van de gevonden synthetische botnets wordt er in deze paragraaf een keuze gemaakt met welke synthetische botnet de botnetsimulaties voor dit wetenschappelijk empirisch onderzoek worden uitgevoerd.

5.3.1. KENMERKEN GEVONDEN SYNTHETISCHE BOTNETS

In de vorige paragraaf zijn de kenmerken van de gevonden synthetische botnets geanalyseerd. Hierbij voldoen de beide gevonden synthetische botnets aan de gestelde voorwaarden van de botnet topologie en worden beide gewaardeerd met '++'.

De HybridBot botnet voldoet aan de gestelde voorwaarden van de botnet levenscyclus en wordt gewaardeerd met '++' en de BYOB botnet wordt gewaardeerd met '+' omdat deze alleen maar aan de minimale voorwaarde voldoet.

5.3.2. EIGENSCHAPPEN GEVONDEN SYNTHETISCHE BOTNETS

Om een uitspraak te doen over de geschiktheid van de BYOB botnet en de HybridBot botnet, gaan we deze gevonden synthetische botnets vergelijken op de verplichte en wenselijke eigenschappen. Deze worden hieronder verder toegelicht.

GESCHIKTHEID VOOR DIT ONDERZOEK

Beide synthetische botnets zijn geschikt om in het DVCSL te kunnen draaien omdat beide platform onafhankelijk zijn. Op dit punt is er geen doorslaggevend verschil tussen beide synthetische botnets.

BESCHIKBAARHEID VOOR ONDERZOEK

Beide synthetische botnets zijn beschikbaar voor dit onderzoek omdat deze volledig bestaan uit OpenSource componenten. Op dit punt is er geen doorslaggevend verschil tussen beide synthetische botnets.

COMPLEXITEIT

De broncode van beide synthetische botnets is goed te begrijpen en is daarmee niet al te complex. Ook op dit punt is er geen doorslaggevend verschil tussen beide synthetische botnets.

AANPASBAARHEID

De aanpasbaarheid van de broncode is voor beide synthetische botnets redelijk gelijk. Beide maken gebruik van de meest gangbare platform onafhankelijke programmeertalen. Dus ook op dit punt is er geen doorslaggevend verschil tussen beide synthetische botnets.

BESCHIKBARE INFORMATIE

Voor de BYOB is vast gesteld dat de beschikbare informatie alleen uit de broncode van de botnet bestaat. Maar de HybridBot heeft daarnaast nog wat aanvullende plaatjes en een aantal informatieve tekstbestanden toegevoegd. Hiermee heeft de HybridBot een lichte voorkeur boven de BYOB.

BENODIGDE RESOURCES

De C&C Systemen van beide synthetische botnets zijn geschreven in PHP en maken beide gebruik van een MySQL database server en een Apache webserver. Het C&C Systeem komt in beide synthetische botnet omgevingen maar één keer voor, dus op dit punt is er ook geen doorslaggevend verschil.

Voor het bepalen van de benodigde resources van de beide synthetische bots is er een zeer beperkte analyse gedaan naar het minimaal benodigde werkgeheugen van een UML-host om daarin een bot te kunnen starten. Hierbij is vastgesteld dat een voor dit onderzoek gebruikte UML-host minimaal 55MB aan werkgeheugen moet hebben. Voor het kunnen starten van van de HybridBot bot is er 56MB aan werkgeheugen nodig en voor het starten van de BYOB bot is er 66MB aan werkgeheugen nodig. Hiermee is proefondervindelijk vast gesteld dat de HybridBot bot voldoende heeft aan 1MB werkgeheugen en dat de BYOB bot 10MB aan werkgeheugen nodig heeft.

De bots van de BYOB botnet zijn geschreven in Java waardoor er voor het draaien van een bot ook een Java virtuele omgeving gestart moet worden. Dit heeft nadelige gevolgen voor het geheugengebruik omdat er voor elke bot ook een Java virtuele machine gestart moet worden.

De bots van de HybridBot zijn geschreven in Perl waardoor deze maar relatief weinig werkgeheugen nodig heeft.

Een bot komt heel vaak voor in een botnet omgeving en als voor elke bot er een aparte Java virtuele omgeving opgestart moet worden dan maakt dit wel een groot verschil. Hiermee heeft de HybridBot botnet op dit punt de absolute voorkeur.

WAARDERING

Op basis van de bovenstaande verplichte en wenselijke eigenschappen wordt de BYOB botnet gewaardeerd met een '+' en de HybridBot botnet wordt gewaardeerd met een '++'. Beide synthetische botnets voldoen aan de verplichte eigenschappen, maar bij de wenselijke eigenschappen heeft de HybridBot botnet op twee punten een voorkeur in vergelijking met de BYOB botnet.

5.3.3. KEUZE

De waardering van de kenmerken van botnets en de waardering van verplichte en wenselijke eigenschappen is weergegeven in tabel 5.1 Beoordeling geschikte botnets. Hieruit is vast te stellen dat de HybridBot botnet het beste kan worden gebruikt bij de botnetsimulaties van dit wetenschappelijk empirisch onderzoek.

In de volgende paragraaf wordt de geschiktheid van de HybridBot botnet verder geanalyseerd, om te bepalen of met de HybridBot botnet de botnetsimulaties uitgevoerd kunnen worden in het DVCSL.

Tabel 5.1: Beoordeling geschikte botnets

	Topologie	Levenscycles	Eigenschappen	Waardering
BYOB	++	+	+	4
HybridBot	++	++	++	6

Waardering:

- − Voldoet niet de voorwaarden
- + Voldoet aan de minimale voorwaarden
- ++ Voldoet aan de gestelde voorwaarden

- De waardering is de optelling van de gegeven waardes, de grootste waardering voldoet het beste aan de gestelde kenmerken en de gevraagde verplichte en wenselijke eigenschappen.

5.4. HYBRIDBOT

Zoals hierboven is aangegeven is de HybridBot botnet een Open Source, op PHP en Perl gebaseerde botnet. In deze paragraaf wordt de HybridBot botnet verder geanalyseerd, met als doel om vast te stellen of het mogelijk is deze in het DVCSL te kunnen draaien.

Als eerste worden de kenmerken en eigenschappen van de HybridBot botnet die tijdens deze detail analyse aan het licht zijn gekomen beschreven, waarna de kenmerken van de nodes en verbindingen van paragraaf 4.7 verder worden bepaald.

Vervolgens kan de analyse worden gedaan of deze kenmerken van de nodes en verbindingen van de HybridBot botnet overeen komen met de kenmerken van de nodes en verbindingen van het DVCSL.

5.4.1. KENMERKEN HYBRIDBOT

In paragraaf 3.2 zijn de kenmerken van botnets naar voren gekomen op basis van de bestudeerde wetenschappelijke literatuur. Hierboven zijn deze kenmerken al geanalyseerd op basis van de beschikbare broncode en documentatie. In de volgende paragrafen worden deze kenmerken aangevuld door de HybridBot botnet te installeren en te testen. Deze testen zijn uitgevoerd in de dvcls-botnet-php DVCSL configuratie omgeving, zie paragraaf C.2.2.

ARCHITECTUUR HYBRIDBOT

De architectuur van de HybridBot botnet wijkt niet veel af van wat er in paragraaf 3.2.1 al beschreven is. Ook binnen de HybridBot botnet is er een C&C Systeem, bot en één of meerdere doelwit systemen. Er zijn maar weinig mogelijkheden om deze botnet automatisch uitbreiden met nieuwe bot systemen.

Er is een mogelijkheid om gebruik te maken van een bestandsuitwisseling dienst, 'File Transfer Protocol' (FTP). Maar hiervoor moet de botmaster zelf actief op zoek naar onbeschermde systemen die hiervoor zijn aangepast. Deze infectie stap kan wel handmatig

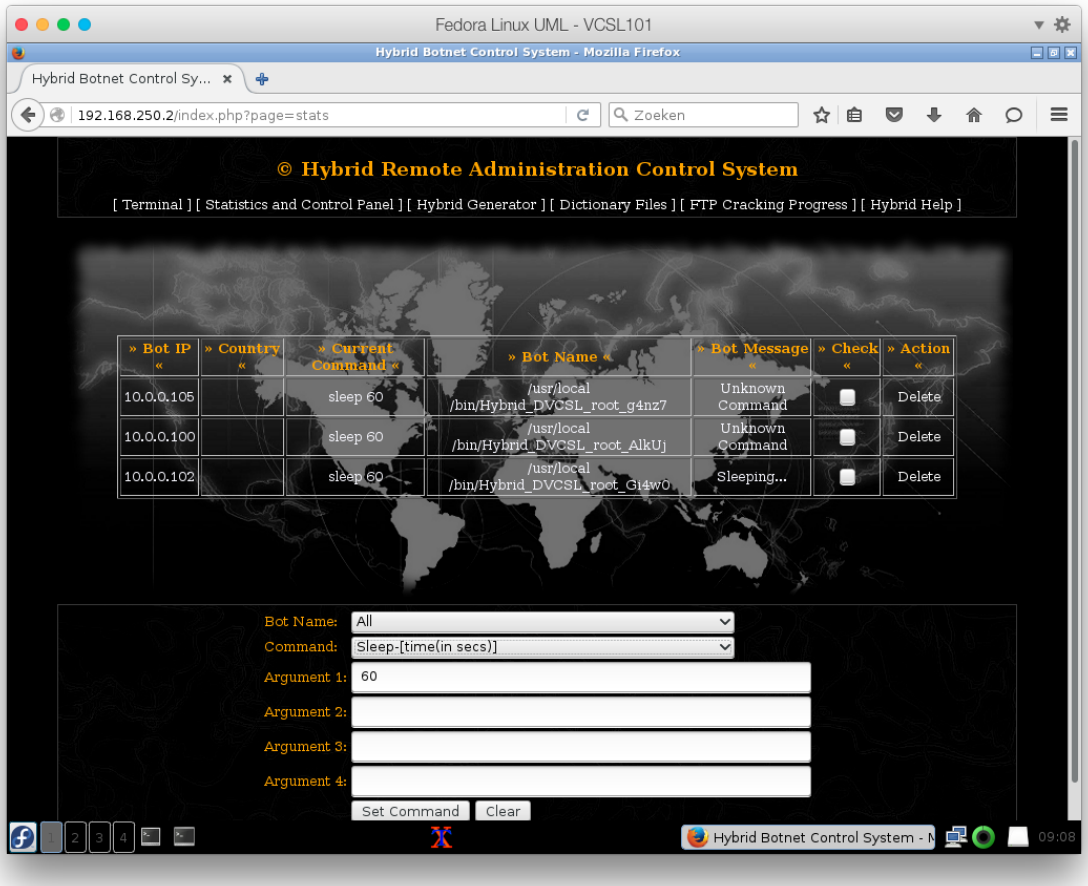
worden gedaan door de HybridBot bot op een systeem te installeren en te starten, waardoor dit architectuur element ook te simuleren is.

LEVENSCYCLUS HYBRIDBOT

Zoals hierboven al is aangegeven is de Levenscyclus van de HybridBot botnet bijna helemaal compleet. Alleen heeft de HybridBot geen geautomatiseerde infectie fase. Deze kan handmatig worden uitgevoerd door op het te besmetten systeem het bot Perl script uit te voeren, waarna de HybridBot bot op dat systeem actief wordt. Daarmee is dit systeem een HybridBot bot systeem geworden.

Zodra de HybridBot bot actief is geworden gaat deze zich registreren bij het HybridBot C&C Systeem. Het C&C Systeem zet vervolgens de bot gegevens in de MySQL database zodat deze informatie terug kan worden gegeven aan de botmaster als deze hierom vraagt.

Zowel de botmaster als de HybridBot bot maken gebruik van het C&C PHP webpagina's die worden geserveerd door een Apache webserver. Het HybridBot C&C console voor het initiëren van een botnet aanval is weergegeven infiguur 5.1.

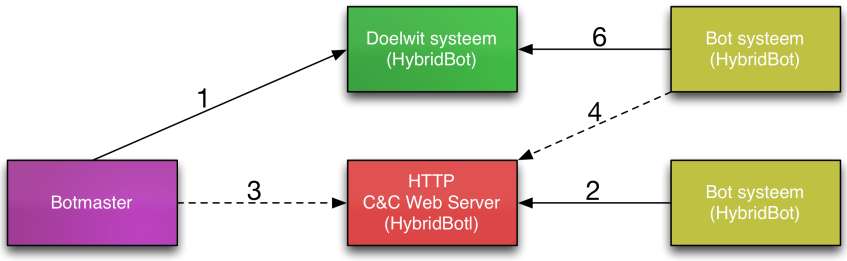


Figuur 5.1: HybridBot C&C

De botmaster kan met behulp van een standaard Internet browser de opdrachten klaar zetten in het HybridBot C&C Systeem, door naar de webpagina te gaan van het HybridBot C&C

Systeem. Hierbij kan de botmaster één geregistreerde bot kiezen, maar ook een selectie van geregistreerde bots, of alle bots.

Als de HybridBot bot zich heeft geregistreerd bij het HybridBot C&C Systeem gaat deze op gezette tijden vragen of er een actie klaar staat die uitgevoerd moet worden. Op deze wijze geeft het HybridBot C&C Systeem de opdrachten van de botmaster door aan de HybridBot bots. Dit is anders dan er in figuur 3.2 staat beschreven, waar het C&C Systeem de opdracht actief doorgeeft aan de bots. De levenscyclus van de HybridBot botnet is schematisch weergegeven in figuur 5.2.



1. De botmaster infecteert een doelwit systeem met een HybridBot bot.
2. Een HybridBot bot maakt zich bekend bij het HybridBot C&C Systeem.
3. De botmaster geeft opdrachten door via het HybridBot C&C Systeem.
4. Een HybridBot bot haalt de opdrachten op bij het HybridBot C&C Systeem.
5. ~~De HybridBot bots worden voorzien van updates via het HybridBot C&C Systeem.~~
   Niet geïmplementeerd binnen de HybridBot botnet.
6. De HybridBot bots vallen een doelwit systeem aan.

Figuur 5.2: HybridBot Levenscyclus

5.4.2. OVERIGE KENMERKEN HYBRIDBOT

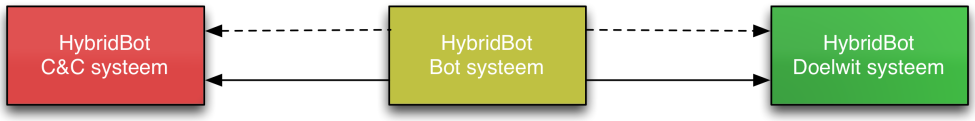
De besmette software van de HybridBot botnet bestaat uit een relatief klein Perl script die op een systeem gezet kan worden. Op het moment dat het Perl script handmatig wordt uitgevoerd, zorgt het Perl script er voor dat een kopie van het Perl script automatisch gestart wordt als het systeem gestart wordt. Dit is in feite de HybridBot bot die er voor zorgt dat het systeem een HybridBot bot systeem wordt. Daarna zorgt het Perl script er voor dat deze HybridBot bot ook al gelijk actief is en er niet hoeft worden gewacht totdat het systeem wordt herstart.

De botmaster kan een opdracht klaar zetten in het HybridBot C&C Systeem om een bepaald systeem te gaan aanvallen. Hij kan hierbij de keuze maken uit één specifieke bot, een te selecteren aantal bots of alle beschikbare bots om hiermee een te kiezen aanval uit te voeren.

In figuur 5.1 is de interface die hiervoor beschikbaar is voor de botmaster weergegeven. In Bijlage B 'Botnet Configuratie' staan de mogelijke opdrachten van de HybridBot botnet weergegeven, zie hiervoor figuur B.3.

5.5. KENMERKEN VAN DE NODES EN VERBINDINGEN

Zoals in de vorige paragraaf is aangegeven, zijn er een aantal kenmerken van de HybridBot botnet die van belang zijn voor de netwerkvirtualisatie van de HybridBot botnet. In de volgende twee paragrafen worden de kenmerken van de nodes en verbindingen van de HybridBot botnet beschreven. Zie ook de HybridBot botnet graaf van figuur 5.3.



Figuur 5.3: HybridBot Graaf

5.5.1. HYBRIDBOT NODES

Er zijn drie verschillende nodes te herkennen in de HybridBot botnet topologie. Deze zijn hiervoor al ruim aan bod gekomen. Hieronder alleen een korte opsomming hiervan.

- HybridBot C&C Systeem
- HybridBot bot systeem
- HybridBot doelwit systeem

5.5.2. HYBRIDBOT VERBINDINGEN

Er zijn twee verschillende verbindingen te herkennen in de HybridBot botnet topologie. Deze zijn hiervoor al ruim aan bod gekomen. Hieronder alleen een korte opsomming hiervan.

- HybridBot bot naar het HybridBot C&C Systeem
- HybridBot bot naar het HybridBot doelwit systeem

5.6. GESCHIKTHEID HYBRIDBOT

Op basis van de analyse hierboven is vastgesteld dat het mogelijk is om de HybridBot botnet te kunnen draaien in het DVCSL. Alle kenmerken van de netwerkvirtualisatie van de HybridBot botnet komen overeen met de kenmerken van het DVCSL zoals deze in paragraaf 4.2 zijn beschreven.

De HybridBot botnet nodes en verbindingen zoals deze hierboven zijn beschreven zijn geïmplementeerd in het DVCSL. Er zijn geen aanpassingen nodig aan de HybridBot botnet en aan het DVCSL om de beoogde botnetsimulaties te kunnen uitvoeren.

Er zijn geen specifieke doel systemen of diensten voor de HybridBot botnet ingericht. Dit kunnen willekeurige UML-systemen zijn die door de HybridBot bots benaderd kunnen worden.

De botmaster kan worden gesimuleerd door gebruik te maken van een standaard Internet browser op de DVCSL node. Dit kan worden gedaan door gebruik te maken van een

TAP-netwerk met het UML-systeem waarop het HybridBot C&C Systeem draait. Door dit virtueel netwerk kan de DVCSL node een verbinding met het UML-systeem.

In de volgende paragraaf worden de uitgevoerde botnetsimulaties beschreven die zijn uitgevoerd met de HybridBot botnet.

5.7. UITGEVOERDE BOTNETSIMULATIES

In dit empirisch deel van dit wetenschappelijk onderzoek is er een botnetsimulatie uitgevoerd met de HybridBot botnet. Door een levensfase van deze botnet te simuleren in het DVCSL kan proefondervindelijk worden vastgesteld of met het DVCSL botnetsimulaties mogelijk zijn.

Voor deze proef is er een aparte DVCSL configuratie gemaakt, waarin de verschillende botnetsimulaties zijn uitgevoerd.

Voor de technische details van de onderstaande paragraven, zie Bijlage B 'Botnet Configuratie', paragraaf B.2.

5.7.1. DVCSL BOTNET TEST CONFIGURATIE

De DVCSL botnet test configuratie bestaat uit drie DVCSL nodes, waarbij de botmaster en de HybridBot C&C Systeem gebruik maken van DVCSL-node-1. Als basis van deze DVCSL configuratie is de DVCSL configuratie van de gecombineerde 'Domain Name System' (DNS) en 'Dynamic Host Configuration Protocol' (DHCP) netwerkservices gebruikt.

Voor de botmaster is er een TAP-netwerk beschikbaar gemaakt en voor de HybridBot C&C Systeem is de 'router' UML-host ingericht met extra werkgeheugen. Dit extra geheugen is nodig om de Apache webserver en de MySQL database te kunnen draaien in deze UML-host en voor de HybridBot bots en bot systemen is gebruik gemaakt van de standaard UML-hosts. De virtuele netwerkconfiguratie is weergegeven in tabel 5.2 DVCSL botnet test UML nodes.

Tabel 5.2: DVCSL botnet test UML nodes

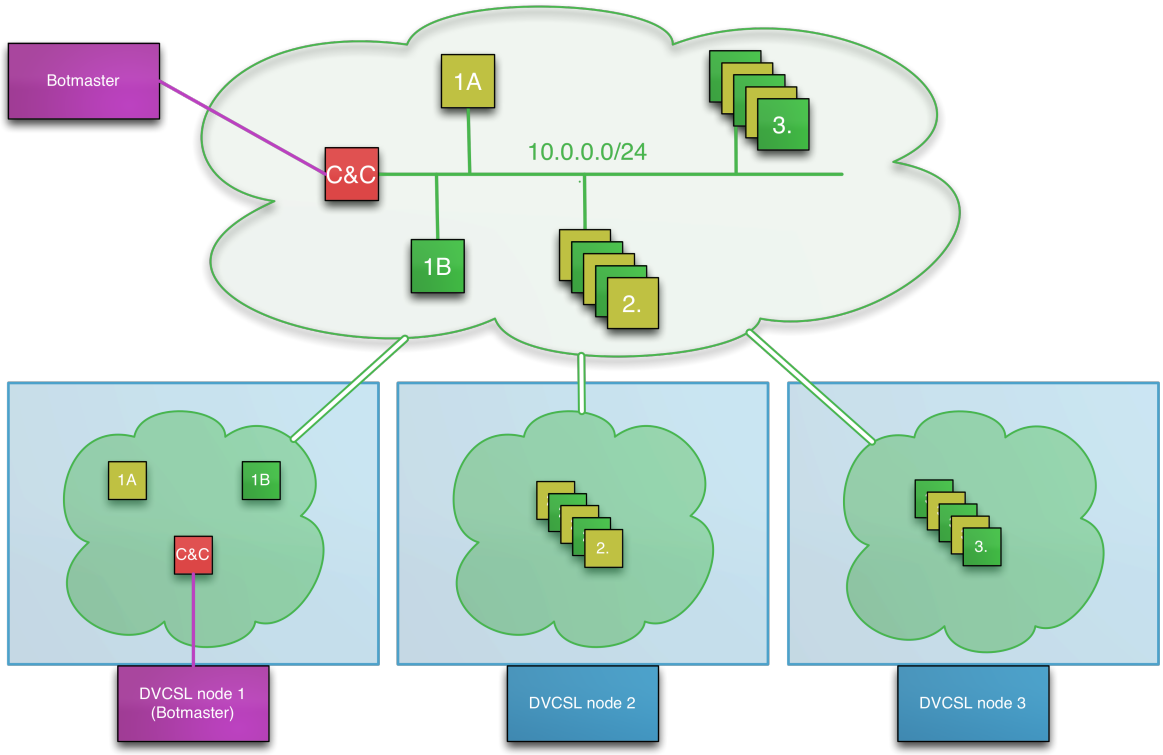
	UML-host	**IP-adres**	**hostnaam** (*.t75317.dvcsl.edu.)	**Netwerkservices**
DVCSL-node-1	router	10.0.0.1	server01	DNS/DHCP server C&C Systeem
		192.168.250.2		TAP-netwerk 192.168.250.1
	pc101 a en b	DHCP	pc*	DHCP client bot systeem
DVCSL-node-2	pc102 a t/m t	DHCP	pc*	DHCP client bot systeem
DVCSL-node-3	pc103 a t/m t	DHCP	pc*	DHCP client bot systeem

Als doelwit systeem voor deze botnetsimulatie is gekozen om hiervoor de webserver van de HybridBot C&C aan te vallen. Dan hoeft hiervoor geen speciale UML-host te worden geconfigureerd.

De startvolgorde van de DVCSL nodes is belangrijk. Als eerste moet de DVCSL-node-1 worden gestart en moet deze volledig operationeel zijn voordat de andere DVCSL nodes worden gestart. Als eerste wordt hierop de virtuele 'router' gestart, daarna worden er nog twee UML-hosts opgestart die als bot systemen kunnen dienen.

5.7.2. HYBRIDBOT BOTNET CONFIGURATIE

Tijdens het starten van de 'router' UML-host wordt de HybridBot C&C Systeem automatisch geconfigureerd, zodat deze direct beschikbaar is voor de HybridBot bots en de botmaster. Als de UML-hosts voor de eerste keer worden gestart, wordt automatisch de HybridBot bot geactiveerd en registreert deze zich bij het HybridBot C&C Systeem. Zie figuur 5.4 voor een grafische weergave van de DVCSL botnet test configuratie. Hierbij zijn de UML-hosts zowel in het groen als het geel weer gegeven. De reden hiervan is dat een UML-hosts start als een HybridBot doelwit systeem en na het starten automatisch de HybridBot bot wordt gestart waardoor het een HybridBot bot systeem is geworden.



Figuur 5.4: DVCSL botnet test

Als de gehele DVCSL botnet test configuratie is gestart dan is de HybridBot botnet volledig operationeel en kunnen de botnetsimulaties worden uitgevoerd. De botmaster heeft dan beschikking over 42 bots, twee van de DVCSL-node-1 en elk 20 bots van de DVCSL-node-2 en DVCSL-node-3.

Door het starten van deze HybridBot botnet zijn er al twee levensfases van een botnet automatisch gesimuleerd, zie figuur 5.2. De uitgevoerde levensfases die automatisch zijn uitgevoerd:

1. De botmaster infecteert een doelwit systeem met een HybridBot bot.
2. Een HybridBot bot maakt zich bekend bij het HybridBot C&C Systeem.

5.7.3. HYBRIDBOT BOTNETSIMULATIE

De uitgevoerde botnetsimulatie met de HybridBot botnet heeft betrekking op de volgende drie levensfases van een botnet van figuur 5.2. De uitgevoerde levensfases tijdens de HybridBot botnetsimulatie zijn:

3. De botmaster geeft opdrachten door via het HybridBot C&C Systeem.
4. Een HybridBot bot haalt de opdrachten op bij het HybridBot C&C Systeem.
5. ~~De HybridBot bots worden voorzien van updates via het HybridBot C&C Systeem.~~
   Niet geïmplementeerd binnen de HybridBot botnet.
6. De HybridBot bots vallen een doelwit systeem aan.

Deze drie levensfases kunnen door één botmaster actie worden uitgevoerd. Hieronder een samenvatting van deze acties.

DE BOTMASTER GEEFT EEN OPDRACHT

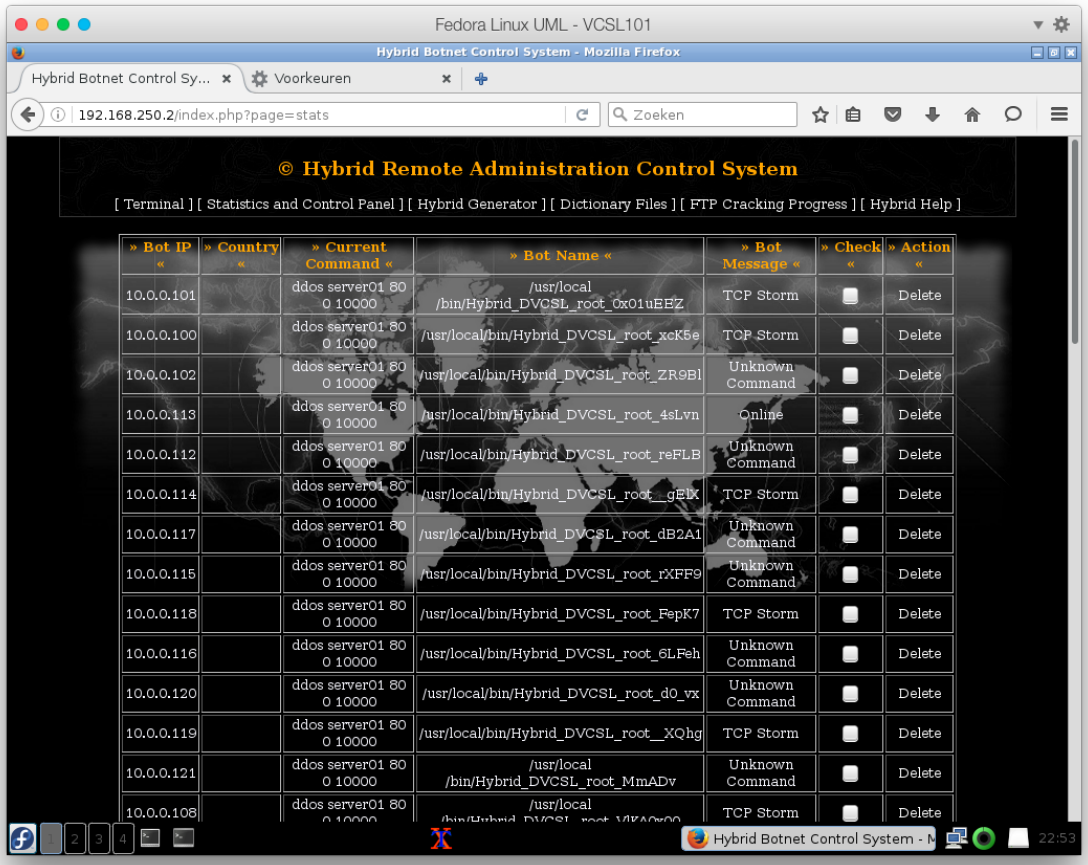
De botmaster gaat naar het C&C console zoals weergegeven in figuur 5.1. Door in het "Statistics and Control Panel" de onderstaande gegevens in te vullen geeft de botmaster de opdracht aan de bots om een aanval uit te voeren op het doelwit systeem, zie tabel 5.3 Botmaster opdracht. Door op de knop "Set Command" wordt deze actie klaar gezet voor alle geregistreerde bots.

Tabel 5.3: Botmaster opdracht

Veld	Waarde	Opmerking
Bot Name	All	Alle bots
Command	TCP Storm	Dit is een TCP DOS aanval en heeft de onderstaande vier parameters nodig
Argument 1	server01	Dit is hostnaam van het doelwitsysteem, in dit geval het C&C systeem
Argument 2	80	Dit is het poort nummer van de Apache webserver van het C&C
Argument 3	0	Hiermee wordt aangegeven dat er niet gewacht wordt op een antwoord, maar continu een nieuwe verbinding wordt opgezet
Argument 4	10000	Hiermee wordt aangegeven dat de bot 10.000 maal een connectie moet opzetten naar het doelwitsysteem

DE BOTS HALEN DE OPDRACHT OP BIJ HET C&C SYSTEEM

Op het moment dat de botmaster de actie heeft klaargezet, gaan de bots deze actie uitvoeren zodra ze deze hebben opgehaald van de C&C Systeem. In figuur 5.5 is te zien dat een aantal bots deze actie hebben opgehaald en aan het uitvoeren zijn.

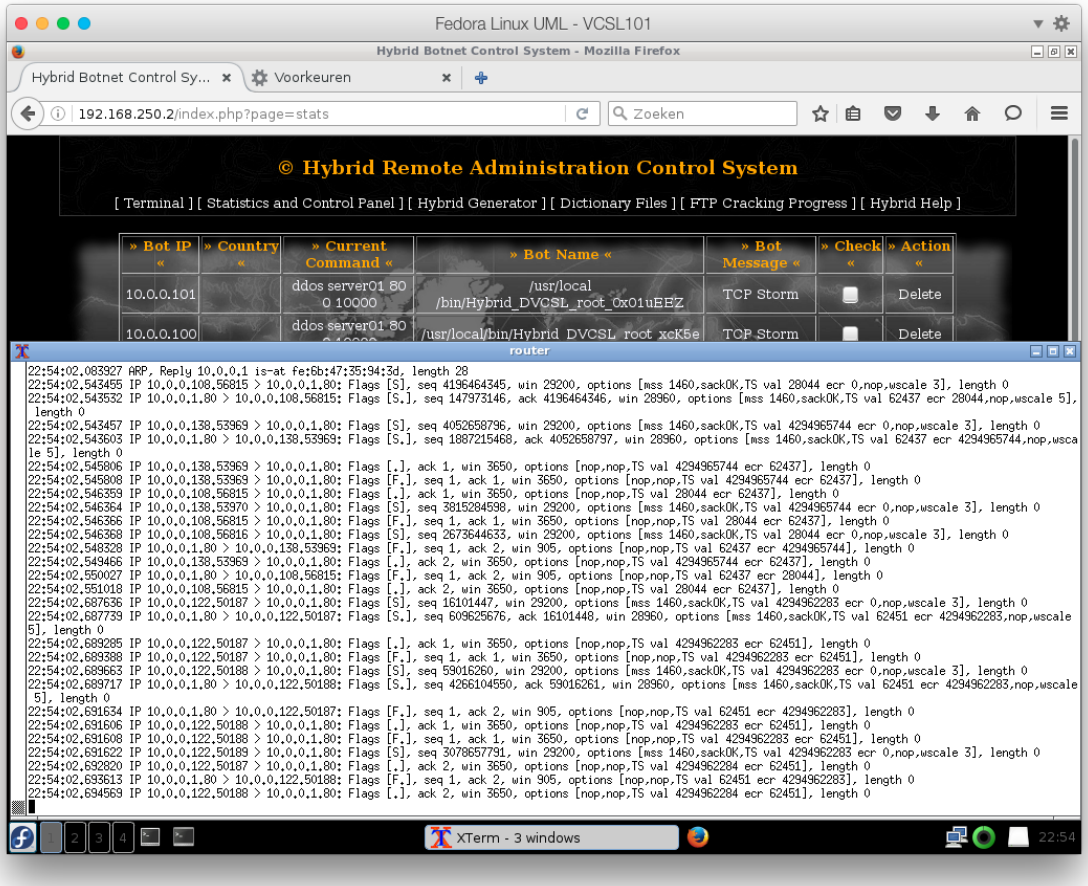


Figuur 5.5: Botnetsimulatie Opdracht

DE BOTS VALLEN EEN DOELWIT SYSTEEM AAN

Doordat het C&C Systeem zelf het doelwit systeem is van deze actie, kunnen de bots zelf ook moeilijker contact krijgen met het C&C Systeem. Als gevolg hiervan kunnen de meeste bots geen nieuwe opdrachten ophalen, maar worden actief op het moment dat het C&C Systeem weer beschikbaar is. Hierdoor blijft deze aanval heel lang doorlopen.

Op de 'router' UML-host is goed te zien aan het virtuele netwerk dat de C&C Systeem overspoeld worden met bevragingen, dit is te zien in het witte venster van figuur 5.6. Hierin worden er 19 'Hypertext Transfer Protocol' (HTTP) verzoeken gestuurd naar de webserver van het C&C Systeem, binnen 0,614569 seconden (22 : 54 : 02.694569 – 22 : 54 : 02.083927). Dit zijn 30 verzoeken per seconde. De webserver wordt hierdoor overspoeld met bevragingen, waardoor deze nog nauwelijks reageert op bevragingen van de HybridBot bots.



Figuur 5.6: Botnetsimulatie Doelwitsysteem

5.8. BOTNETSIMULATIE PROBLEMEN EN OPLOSSINGEN

Tijdens deze fase van het onderzoek zijn er twee problemen gevonden:

- Problemen netwerkverkeer
- Problemen HybridBot C&C

5.8.1. PERFORMANCE NETWERKVERKEER

Bij de analyse van het netwerkverkeer tussen de DVCSL nodes is het volgende probleem naar voren gekomen.

Al het virtueel netwerkverkeer wordt naar alle DVCSL nodes gestuurd, waardoor hetzelfde virtuele netwerkpakketje op het niveau van de DVCSL nodes meerdere malen verstuurd wordt. Hierdoor wordt de virtuele bandbreedte behoorlijk beperkt. In het geval dat er drie DVCSL nodes gebruik maken van dezelfde fysieke verbinding, dan moet een virtueel netwerkpakketje twee maal worden verstuurd. Daarmee is de virtuele bandbreedte maar de helft van de fysieke bandbreedte. Bij vijf DVCSL nodes is de virtuele bandbreedte nog maar een kwart van de fysieke bandbreedte.

Dit gedrag is onlosmakelijk onderdeel van het DVCSL, maar als hier onvoldoende rekening mee gehouden wordt kan dit tot performance problemen leiden. Dit deel van het gedrag

lijkt meer op de werking van een hub, door al het netwerkverkeer naar alle andere DVCSL nodes te sturen. In de werking van de UML-switch binnen het UML-netwerk verandert er niets en vormen de aan elkaar gekoppelde UML-switches één logische UML-switch. Hiermee is het gedrag richting de aangesloten UML-hosts te vergelijken met die van een switch.

Bij het inrichten van een DVCSL omgeving met vele DVCSL nodes moet zoveel mogelijk rekening worden gehouden dat de virtuele netwerkverbindingen overeen komen met de fysieke netwerkverbindingen. Het beste is dat elke verbinding tussen twee DVCSL nodes een niet gedeelde fysieke verbinding is.

Een oplossing hiervoor is om er voor te zorgen dat niet al het virtueel netwerkverkeer naar alle DVCSL nodes moet worden gestuurd. Hierbij kan er gebruik worden gemaakt van intelligente routingstechnieken en of compressie technieken.

Door de virtuele netwerktopologie zoveel mogelijk gelijk te houden met de fysieke netwerktopologie en voor elke DVCSL node een router op te nemen in de virtuele netwerkomgeving, kan er voor gezorgd worden dat alleen het virtuele verkeer dat voor een ander netwerksegment is bestemd alleen maar naar die andere DVCSL node wordt verstuurd. Deze oplossing is ook toegepast in de botnetsimulatie casus in Hoofdstuk 6 'DVCSL botnetsimulatie casus'.

Zoals in paragraaf 4.2.2 is aan gegeven, wordt al het netwerkverkeer naar een centrale DVCSL node gestuurd als er gebruik wordt gemaakt van de centrale autoriteit architectuur [Haa+11]. Als hier gebruik van wordt gemaakt dan kunnen de virtuele netwerktopologie en de fysieke netwerktopologie moeilijk aan elkaar gelijk worden gemaakt. Hierdoor is het gebruik van de centrale autoriteit minder geschikt voor het uitvoeren van botnetsimulaties.

5.8.2. PROBLEMEN HYBRIDBOT CNC

Tijdens deze fase van dit onderzoek is er een installatie probleem van de HybridBot botnet ontdekt. Hieronder een korte samenvatting. Voor de technische details van dit probleem en de oplossing hiervan, zie paragraaf B.2.3.

Het installeren van het C&C Systeem van de HybridBot botnet was redelijk simpel. Het bleek echter niet mogelijk om de C&C interface in figuur 5.1 werkend te krijgen. Er was geen duidelijke fout te vinden. Na analyse van de installatie en configuratie scripts was het probleem gevonden. De objecten in de MySQL database werden niet aangemaakt omdat bepaalde opties niet meer worden ondersteund in de huidige versie van de MySQL database.

5.9. ONDERZOEKVRAGEN

In dit hoofdstuk zijn de antwoorden gegeven op de laatste drie onderzoeksdeelvragen. In de volgende paragrafen worden deze toegelicht.

WELKE KENMERKEN VAN BOTNETS KUNNEN WORDEN GESIMULEERD IN HET DVCSL?

Zoals in de vorige paragraaf proefondervindelijk is vast gesteld zijn alle nodes en verbindingen van een botnet te virtualiseren in het DVCSL. Daarmee zijn alle kenmerken van een botnet te simuleren in het DVCSL.

WELKE KENMERKEN VAN BOTNETS KUNNEN NIET WORDEN GESIMULEERD IN HET DVCSL?

Zoals hierboven is vastgesteld kunnen alle kenmerken van het HybridBot botnet worden gesimuleerd in het DVCSL.

Zolang aan de gestelde ontwerp-principes en ontwerp-condities kan worden voldaan van het DVCSL, kunnen deze botnets worden gesimuleerd. Voor elke nieuwe botnet moet deze analyse opnieuw worden gedaan. Een belangrijk gegeven is dat de te simuleren botnet moet kunnen draaien in een Linux omgeving. Als dit niet mogelijk is, dan kan deze botnet niet volledig worden gesimuleerd in het DVCSL.

ZIJN ER AANPASSINGEN NODIG AAN HET DVCSL OM BOTNETSIMULATIES TE KUNNEN UITVOEREN?

Op basis van dit onderzoek zijn er geen punten gevonden waarop het DVCSL zou aangepast moeten worden.

Maar er zijn wel een aantal beperkingen gevonden die een grootschalig gebruik van het DVCSL kan beperken, en daarmee een grootschalig gebruik van het DVCSL voor botnetsimulaties onmogelijk maakt.

Doordat al het virtuele netwerkverkeer naar alle andere DVCSL nodes gestuurd word, heeft dit een behoorlijke impact op de virtuele netwerk bandbreedte. De virtuele bandbreedte neemt voor elke extra DVCSL node af, als deze van dezelfde fysieke netwerkverbinding gebruik maakt. Dit is weer te geven met de volgende formule.

$$\text{virtuele bandbreede} = \frac{\text{fysieke bandbreede}}{(\text{aantal DVCSL nodes} - 1)}$$

Dit probleem kan worden opgelost door niet al het virtuele netwerkverkeer te verzenden en door gebruik te maken van intelligente routeringstechnieken en/of gebruik te maken van compressie technieken om daarmee de hoeveelheid te verzenden gegevens te beperken op het netwerkniveau van de DVCSL nodes. Dit kan een onderwerp zijn voor vervolgonderzoek.

Door voor elke DVCSL node een router op te nemen in de virtuele netwerktopologie kan het te versturen virtuele netwerkverkeer worden beperk. Dit is toegepast in de botnetsimulatie casus van Hoofdstuk 6 'DVCSL botnetsimulatie casus'.

6

DVCSL BOTNETSIMULATIE CASUS

In dit hoofdstuk wordt er een concrete botnetsimulatie casus beschreven. Het doel van deze casus is een botnet aanval te simuleren zoals deze ook op het “echte” Internet zich voor doet.

In het vorige hoofdstuk is er al een botnetsimulatie uitgevoerd, waarmee het wetenschappelijk bewijs is geleverd dat een botnetsimulatie mogelijk is in het ‘Gedistribueerd Virtuele Computer Security Lab’ (DVCSL). Door deze botnetsimulatie casus is er ook een concrete toepassing waarbij het DVCSL ingezet kan worden bij botnetsimulaties.

Als eerste wordt de botnetsimulatie casus verder uitgewerkt. Daarna wordt de gebruikte botnetsimulatie casus configuratie beschreven waarin de gebruikte netwerktopologie en de gebruikte DVCSL configuratie worden toegelicht.

Daarna wordt de uitgevoerde botnetsimulatie verder toegelicht waarin ook een beperkte analyse is uitgevoerd van de verzamelde simulatie gegevens, waarna de conclusie van deze botnetsimulatie casus kan worden gegeven.

Als laatste wordt de laatste onderzoeksvraag beantwoord, waarmee ook de hoofd onderzoeksvraag beantwoord kan worden.

6.1. DE BOTNETSIMULATIE CASUS

Met de botnetsimulatie van Hoofdstuk 5 ‘Botnetsimulatie’ is vastgesteld dat botnetsimulaties mogelijk zijn in het DVCSL. Het doel van deze botnetsimulatie casus is om vast te stellen of het DVCSL geschikt is om realistische botnetsimulaties te kunnen simuleren en te analyseren.

De casus voor deze botnetsimulatie is als volgt te beschrijven. Er moet een botnet aanval kunnen worden gesimuleerd zoals deze ook op het echte Internet zich voordoet.

Het echte Internet bestaat uit vele ‘Internet service provider’ (ISP)’s die met elkaar zijn verbonden. Een ISP is een bedrijf die gebruikers toegang geeft tot het Internet. Één van de gevaren van botnets is dat deze een Internet dienst kan lam leggen door middel van een ‘Distributed Denial of Service’ (DDoS) aanval. Een voorbeeld van zo’n aanval is een ‘Domain Name System’ (DNS) server DDoS aanval.

De gekozen casus is om een DNS server DDoS aanval te simuleren in het DVCSL middels de HybridBot botnet.

Binnen het netwerk van een ISP bevindt zich een C&C Systeem die door een botmaster wordt beheerd. Deze botmaster voert een DDoS aanval uit op de DNS server van zijn ISP zodat deze niet meer beschikbaar is voor DNS bevragingen. Deze DDoS aanval wordt uitgevoerd vanaf meerdere ISP's, die met elkaar verbonden zijn door middel van een fictief Internet.

Op basis van de hierboven beschreven casus worden de drie DVCSL nodes ingezet als drie ISP's die met elkaar zijn verbonden met een UML-switch die het fictieve Internet representeert.

Voor deze botnetsimulatie is er een speciale DVCSL configuratie gemaakt. Deze wordt in de volgende paragraaf verder uitgewerkt.

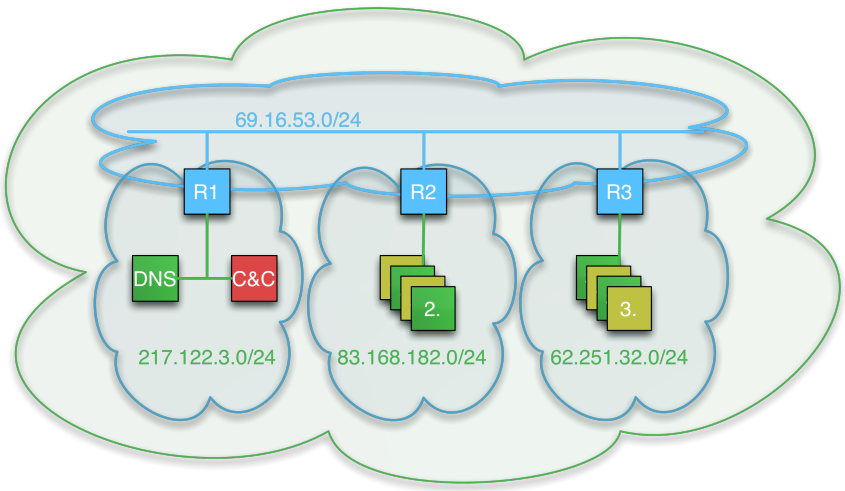
6.2. BOTNETSIMULATIE CASUS CONFIGURATIE

De gebruikte DVCSL configuratie voor deze casus bestaat uit een aantal DVCSL nodes die elk een ISP vertegenwoordigt. Het fictieve Internet waarmee deze ISP's met elkaar zijn verbonden wordt gevormd door een router UML-hosts in elke DVCSL node. Deze routers zijn met elkaar verbonden middels de DVCSL virtuele UML-switch techniek.

Deze configuratie maakt optimaal gebruik van de horizontale schaalbaarheid van het DVCSL. Door extra DVCSL nodes toe te voegen aan de DVCSL omgeving kunnen er extra ISP's worden toegevoegd aan de botnetsimulatie. In het geval dat deze casus wordt ingezet bij een cursus of training, kunnen studenten elk een ISP beheren. Hiermee kunnen ze op een interactieve manier de gevaren van botnets leren te herkennen en mogelijk te beperken.

6.2.1. TOPOLOGIE BOTNETSIMULATIE CASUS

Voor dit onderzoek zijn er drie DVCSL nodes beschikbaar. De eerste DVCSL node wordt gebruikt voor de ISP1 waarin het C&C Systeem van de HybridBot botnet is geconfigureerd en waarin de aan te vallen DNS netwerkservice zit. De andere twee DVCSL nodes zijn geconfigureerd als ISP's waar vandaan de aanval wordt uitgevoerd. Zie figuur 6.1 voor de grafische representatie van de gebruikte netwerktopologie.



Figuur 6.1: Netwerktopologie botnetsimulatie casus

De groene wolk representeert de logische 'User Mode Linux' (UML) virtuele netwerkomgeving, zoals deze ook in figuur 4.2 is aangegeven. In deze wolk is het fictieve Internet weergegeven in een horizontale wolk, waarin de verschillende UML router met elkaar zijn verbonden, middels het '69.16.53.0/24' netwerksegment. De verticale wolken representeren de verschillende ISP's, waarbij de linker wolk de ISP waarin het C&C Systeem is geconfigureerd. In dit figuur zijn de andere twee ISP's rechts hiervan weergegeven, maar dit kan uiteraard naar wens worden uitgebreid.

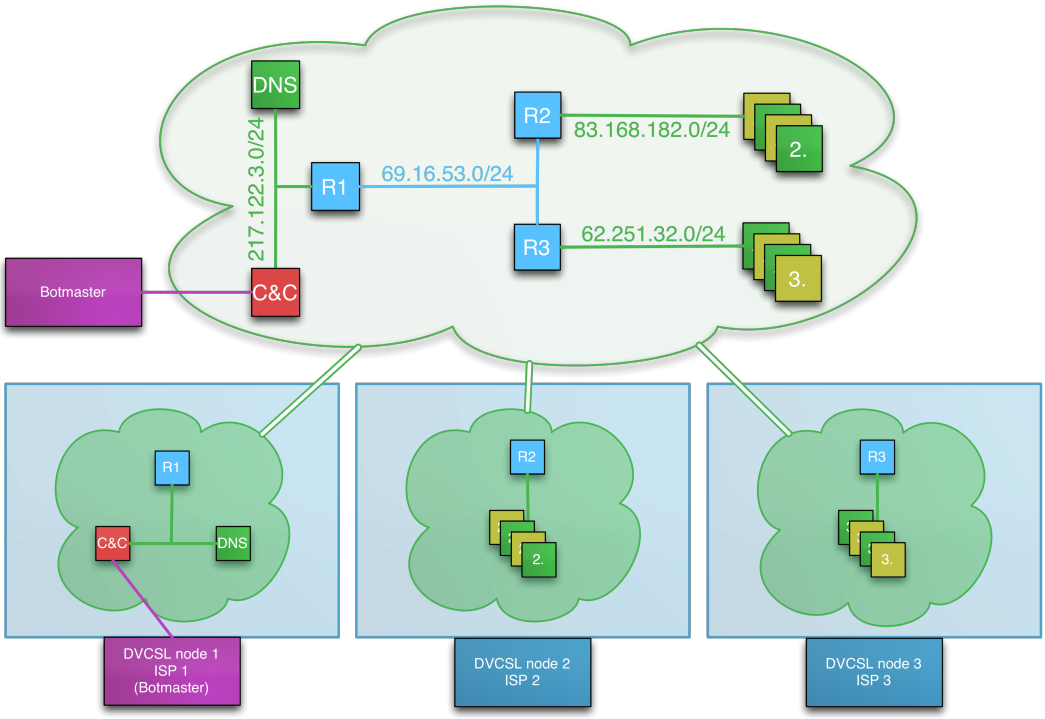
Elke ISP heeft zijn eigen netwerksegment en is zelf verantwoordelijk voor het uitgeven van zijn eigen IP-adressen. De UML routers zijn verantwoordelijk voor de juiste routering van het netwerkverkeer. Hiermee is er een netwerktopologie gemaakt die heel dicht bij het echte Internet komt.

De gekozen virtuele netwerktopologie ondervangt zoveel mogelijk de performance problemen zoals deze in paragraaf 5.8.1 zijn beschreven. Per DVCSL node is er een router UML-host opgenomen in het virtuele netwerk.

6.2.2. DVCSL BOTNETSIMULATIE CONFIGURATIE

Elke router UML-host geeft de IP-adressen uit aan de UML-hosts binnen zijn ISP netwerk. Hiervoor is op alle UML routers een 'Dynamic Host Configuration Protocol' (DHCP) netwerkservice geconfigureerd, die voor zijn ISP de IP-adressen uitgeeft.

Om de configuratie niet te complex te maken is er maar één DNS netwerkservice geconfigureerd. Deze zit in het ISP1 netwerk van het C&C Systeem en is ook het doelwit systeem van deze botnetsimulatie casus. Zie figuur 6.2 voor de grafische representatie van de DVCSL botnetsimulatie configuratie.



Figuur 6.2: DVCSL botnetsimulatie configuratie

Eigenlijk maakt het niet uit hoe het virtuele ISP netwerk is geconfigureerd, als het fictieve Internet verkeer maar via de eigen router UML-host wordt verstuurt en ontvangen. Hiermee geeft deze DVCSL configuratie veel mogelijkheden om naar eigen inzicht aanpassingen door te voeren.

Als basis voor deze configuratie is de DVCSL configuratie van paragraaf 5.7 gebruikt. Voor de technische details zie Bijlage C 'Botnetsimulatie casus', paragraaf C.1. De virtuele netwerkconfiguratie is weergegeven in tabel 6.1 DVCSL botnetsimulatie configuratie UML nodes.

Tabel 6.1: DVCSL botnetsimulatie configuratie UML nodes

	UML-host	**IP-adres**	**hostnaam** (*.t75317.dvcsl.edu.)	**Netwerkservices**
DVCSL-node-1	router	217.122.3.1	router.isp1	DHCP server
		69.16.53.1	router001.internet	Internet router
	master	217.122.3.2	master.isp1	C&C Systeem
		192.168.250.2	-	TAP-netwerk
	dnsserver	217.122.3.2	dnsserver.isp1	DNSserver
DVCSL-node-2	router	83.168.182.1	router.isp2	DHCP server
		69.16.53.2	router002.internet	Internet router
	pc102 a t/m t	DHCP	pc*.isp2	DHCP client bot systeem
DVCSL-node-3	router	62.251.32.1	router.isp3	DHCP server
		69.16.53.3	router003.internet	Internet router
	pc103 a t/m t	DHCP	pc*.isp3	DHCP client bot systeem

6.3. DVCSL BOTNETSIMULATIE

Op het moment dat alle ISP DVCSL nodes actief zijn, kan de botmaster de opdracht geven om de DNS server aan te vallen. Alle beschikbare HybridBot bots gaan dan vrijwel gelijk de DNS server aanvallen, waardoor deze niet meer beschikbaar is.

Net als in het vorige hoofdstuk is beschreven, zet de botmaster in het 'Commando en Controle' (C&C) console een opdracht klaar voor alle beschikbare bots. De ingevulde waarden zijn in tabel 6.2 Botmaster opdracht botnetsimulatie casus weergegeven. Zie paragraaf C.2 voor de technische details van deze botnetsimulatie.

De DNS netwerkservice maakt gebruik van zowel het 'User Datagram Protocol' (UDP) netwerkprotocol als het 'Transmission Control Protocol' (TCP) netwerkprotocol. De HybridBot botnet kan zowel een UDP aanval doen middels een 'UDP Storm' aanval als een TCP aanval middels een 'TCP Storm' of een 'SYN Storm'. Gekozen is om voor deze simulatie de 'SYN Storm' te gebruiken omdat deze ook vaak wordt gebruikt op het echte Internet.

Het poort nummer waarop de DNS service draait is poort 53. Het verschil tussen een 'TCP Storm' aanval en een 'SYN Storm' aanval is dat bij een 'SYN Storm' alleen het eerste berichtje van een TCP connectie wordt verstuurd. Hierdoor worden de TCP connecties niet netjes afgesloten en blijven deze voor een langere tijd actief. Als er heel veel van deze connecties zijn, kunnen op een gegeven moment geen nieuwe verbindingen worden gemaakt, waardoor het systeem niet meer te benaderen is.

Tabel 6.2: Botmaster opdracht botnetsimulatie casus

Veld	Waarde	Opmerking
Bot Name	All	Alle bots
Command	SYN Storm	Dit is een TCP/SYN DOS aanval en heeft de onderstaande vier parameters nodig
Argument 1	dnsserver.isp1.t75317.dvcsl.edu.	
		Dit is hostnaam van het doelwitsysteem, de DNS server
Argument 2	53	Dit is het poort nummer van de DNS netwerkservice
Argument 3	0	Hiermee wordt aangegeven dat er niet gewacht wordt op een antwoord, maar continu een nieuwe verbinding wordt opgezet
Argument 4	10000	Hiermee wordt aangegeven dat de bot 10.000 maal een connec- tie moet opzetten naar het doelwitsysteem

Het C&C Systeem heeft in deze casus een ander IP-adres gekregen, omdat er nu gebruik wordt gemaakt van IP-adressen die op het 'echte' Internet kunnen bestaan. Hiervoor is er een aanpassing gemaakt in de DNS configuratie. De HybridBot bots van dit onderzoek maken standaard gebruik van de DNS naam 'server01.t75317.dvcsl.nl'. In de DNS configuratie van deze casus is deze DNS naam doorverwezen naar 'master.isp1.t75317.dvcsl.edu'.

Deze botnetsimulatie casus kan op vele manieren worden geanalyseerd. Dit onderzoek beperkt zich tot alleen het bekijken van het netwerkverkeer van de UML router op de DVCSL-node-1 enDVCSL-node-2. Daarnaast wordt de tijd gemeten hoe lang een DNS bevraging duurt, zodat inzichtelijk kan worden gemaakt welke impact de botnet aanval heeft op de DNS server.

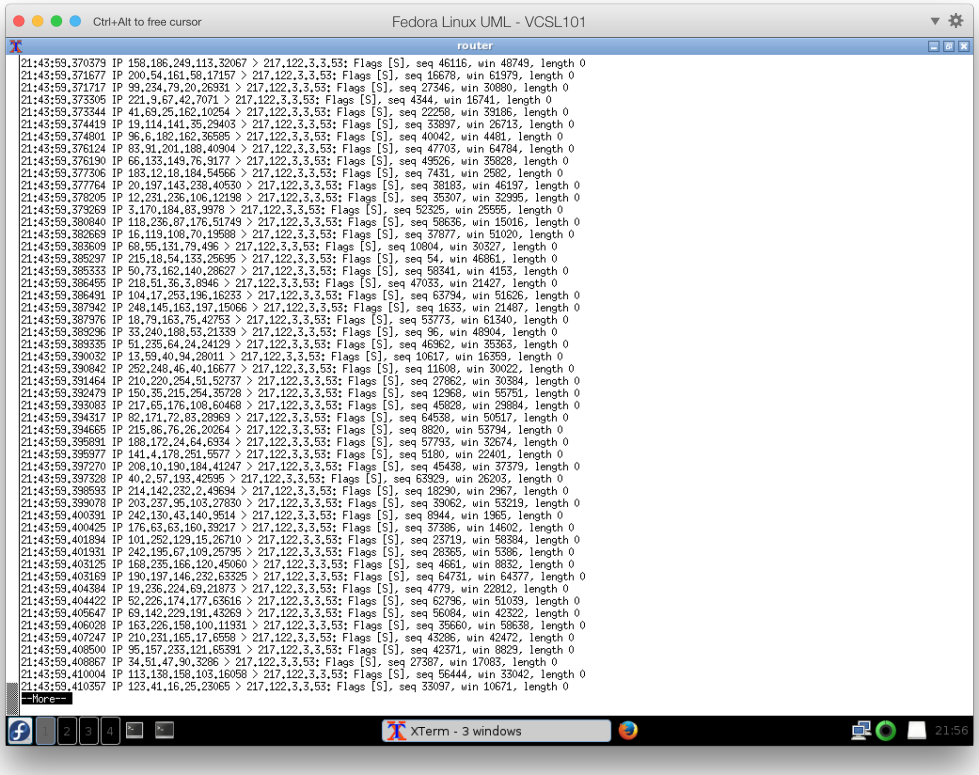
Tijdens deze botnetsimulatie is het netwerkverkeer opgeslagen op het bestandssysteem in speciale gegevensbestanden. Dit heeft de minste impact heeft op de performance van de verschillende componenten omdat er niets naar het scherm wordt gestuurd. Daarmee wordt de botnetsimulatie tot een minimum beïnvloed qua systeem resources. Tijdens een aantal proefsimulaties kwam de processor belasting van dit proces rond de 0% uit, met een uitschieter van 3%.

Na de botnetsimulatie worden deze gegevens weer ingelezen en kunnen deze vervolgens worden geanalyseerd zoals deze in de volgende drie paragrafen zijn uitgevoerd.

6.3.1. NETWERKVERKEER ANALYSE BOTNETSIMULATIE ISP1

Voor deze analyse is het netwerkverkeer van het ISP1 netwerk verkeer onderschept van de router UML-host van ISP1, deze is zichtbaar gemaakt in figuur 6.3.

In dit figuur zijn er 52 netwerkpakketjes zichtbaar die in een tijdbestek van ongeveer 0,034 seconden zijn onderschept. Het bijzondere van deze netwerkpakketjes is dat deze niet afkomstig lijken te zijn van de UML-hosts maar van willekeurige afzenders. Dit fenomeen heet 'Internet Protocol' (IP)-spoofing, waarbij er een ander afzender adres wordt gebruikt dan die door het systeem in gebruik is.



Figuur 6.3: DVCSL botnetsimulatie ISP1

De definitie van een aanvalspakketje is dat deze wordt gestuurd naar de DNS server op poort 53 waarbij het adres van de afzender middels IP-spoofing is gegenereerd.

Bij de analyse van het opgeslagen netwerkverkeer van het ISP1 netwerk zijn een aantal kengetallen geanalyseerd. Deze staan weergegeven in tabel 6.3 Analyse netwerkverkeer ISP1 netwerk.

Het verzamelen van de netwerkgegevens op de router UML-host van ISP1 is gestart op '21:43:45.443877' en heeft doorgelopen tot '21:46:04.927682'. In deze periode zijn er in totaal 174.412 netwerkpakketjes onderschept.

De aanval is gestart om '21:43:58.780924' en gestopt om '21:45:25.149460'. In deze periode van 86,37 seconden zijn er in totaal 167.652 verzoeken verstuurd via het fictieve Internet.

Dit komt overeen met ongeveer 1.941 aanvalspakketjes per seconde. Van de totaal onderschepte netwerkpakketjes is 96% gegenereerd door de HybridBot bots.

Een waardeoordeel kan aan deze analyse niet worden gegeven, maar dat is ook niet de intentie van dit onderzoek.

Tabel 6.3: Analyse netwerkverkeer ISP1 netwerk

Omschrijving	Waarde
Eerste onderschepte pakketje	'21:43:45.443877 IP 83.168.182.105.42409 > 217.122.3.2.80'
Laatste onderschepte pakketje	'21:46:04.927682 IP 217.122.3.2.80 > 62.251.32.119.51798'
Totaal onderschept	174.412
Eerste aanvalspakketje	'21:43:58.780924 IP 249.130.206.17.22078 > 217.122.3.3.53'
Laatste aanvalspakketje	'21:45:25.149460 IP 57.202.158.122.36685 > 217.122.3.3.53'
Aanvalspakketjes	167.652

6.3.2. NETWERKVERKEER ANALYSE BOTNETSIMULATIE ISP2

Voor deze analyse is het netwerkverkeer van het fictieve Internet onderschept van de router UML-host van ISP2. Deze gegevens zijn nagenoeg identiek zoals deze in figuur 6.3 zijn te zien.

Bij de analyse van het opgeslagen netwerkverkeer van het fictieve Internet zijn een aantal kengetallen geanalyseerd. Deze staan weergegeven in tabel 6.4 Analyse netwerkverkeer ISP2 Internet.

Tabel 6.4: Analyse netwerkverkeer ISP2 Internet

Omschrijving	Waarde
Eerste onderschepte pakketje	'21:43:45.424375 IP 83.168.182.105.52950 > 217.122.3.3.53'
Laatste onderschepte pakketje	'21:46:05.581236 IP 217.122.3.2.80 > 62.251.32.119.51798'
Totaal onderschept	173.799
Eerste aanvalspakketje	'21:43:59.434572 IP 249.130.206.17.22078 > 217.122.3.3.53'
Laatste aanvalspakketje	'21:45:25.802476 IP 57.202.158.122.36685 > 217.122.3.3.53'
Aanvalspakketjes	168.207

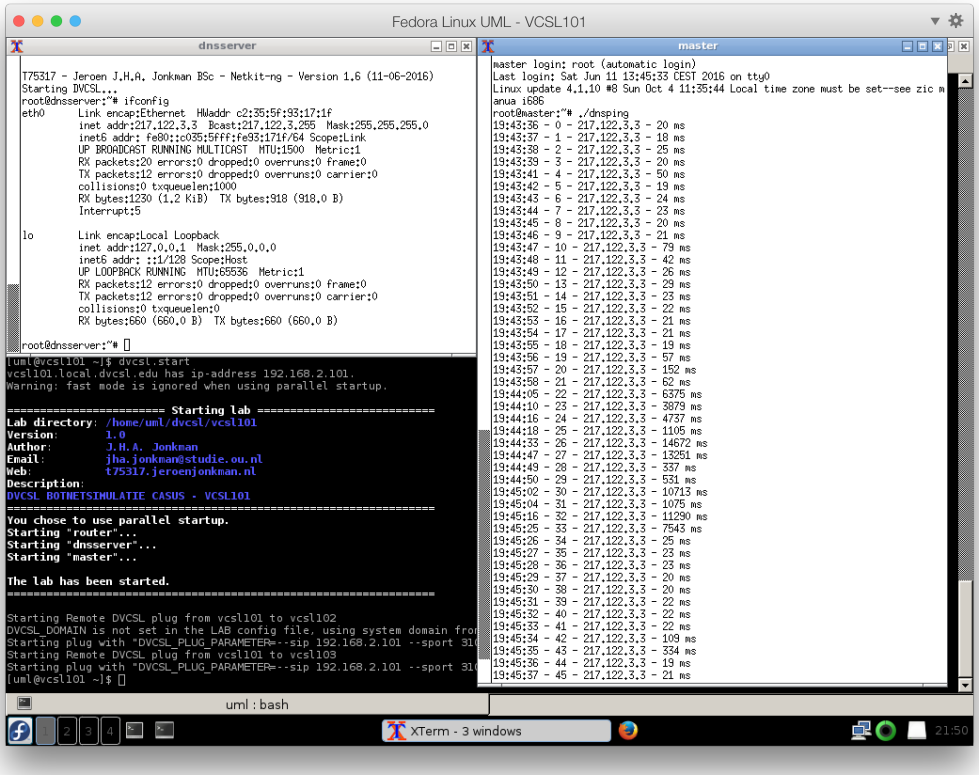
Het verzamelen van de netwerkgegevens op de router UML-host van ISP2 is gestart op '21:43:45.424375' en heeft doorgelopen tot '21:43:45.424375'. In deze periode zijn er in totaal 173.799 netwerkpakketjes onderschept.

De aanval is gestart om '21:43:59.434572' en gestopt om '21:45:25.802476'. In deze periode van 86,37 seconden zijn er in totaal 168.207 verzoeken verstuurd via het fictieve Internet. Dit komt overeen met ongeveer 1.948 aanvalspakketjes per seconde. Van de totaal onderschepte netwerkpakketjes is 97% gegenereerd door de HybridBot bots.

Een waardeoordeel kan aan deze analyse niet worden gegeven, maar dat is ook niet de intentie van dit onderzoek.

6.3.3. BEREIKBAARHEID VAN DE DNS SERVICE

Voor het starten van de botnetsimulatie is er een DNS ping commando gestart die voor, tijdens en na de uitgevoerde aanval de beschikbaarheid van de DNS service kan meten. Deze meting moet wel tijdens de botnetsimulatie draaien. Maar de performance impact van dit commando is minimaal, omdat tussen elke DNS bevraging er één seconde wordt gewacht met het versturen van de volgende DNS bevraging. Bij elke DNS bevraging wordt gewacht tot dat het antwoord is teruggekomen. Het verschil in tijd wordt dan vervolgens weer gegeven. De resultaten van deze meting zijn in figuur 6.4 weergegeven, in het rechter scherm van de master UML-host.



Figuur 6.4: DVCSL botnetsimulatie dnspimg

Zoals hierboven is aangegeven is de botnet aanval gestart om '21:43:58' lokale tijd. De dnsping commando, laat de systeemtijd zien en deze loopt twee uur achter, dit komt overeen met '19:43:58'.

De DNS ping bevragingen tot de botnet aanval zitten tussen de 18 en 152 milliseconde. De DNS bevraging van bijvoorbeeld '19:43:58' duurt 62 milliseconden. De bevraging van een seconde later duurt 6.375 milliseconden.

Tijdens de botnet aanval duren de DNS bevragingen vaak meer dat tien seconden. Hiermee is duidelijk zichtbaar dat de DNS server het behoorlijk zwaar heeft. Rond '19:44:49'

lijkt de DNS server weer sneller te reageren, maar daarna is hij weer onbereikbaar. Dit is mogelijk te verklaren doordat de bots die als eerste beginnen de DNS server onbereikbaar maken. De bots die de opdracht nog moeten ophalen kunnen het IP-adres van de C&C Systeem niet opvragen. Dus starten deze pas nadat de DNS server weer beschikbaar is. Een waardeoordeel kan aan deze analyse niet worden gegeven, maar dat is ook niet de intentie van dit onderzoek.

Op basis van deze analyse kan worden vastgesteld dat het gedrag van een botnet kan worden geanalyseerd op basis van een botnetsimulatie in het DVCSL. Doordat de HybridBot bots zelf de opdrachten moeten ophalen van de C&C Systeem, starten de bots niet allemaal tegelijk met de aanval. Een oplossing tegen deze 'SYN-storm' aanval is dat de ISP routers alleen het netwerkverkeer doorsturen dat afkomstig is van het 'eigen' ISP netwerk. Bij het inzetten van deze botnetsimulatie casus in het onderwijs kan dit een mogelijke vervolg opdracht zijn voor de studenten.

6.4. CONCLUSIE BOTNETSIMULATIE CASUS

Met deze botnetsimulatie casus is aangetoond dat het DVCSL kan worden gebruikt voor het uitvoeren van concrete botnetsimulaties. Verder onderzoek moet uitwijzen of het ook werkelijk ingezet kan worden in bijvoorbeeld beveiligingscursussen van de 'Open Universiteit' (OU).

Het belangrijkste resultaat van dit wetenschappelijk empirisch onderzoek is, dat is aangetoond dat praktijkgerichte botnetsimulaties mogelijk zijn in het DVCSL van de OU. Hiermee kan worden vastgesteld dat het DVCSL kan worden ingezet voor het uitvoeren van botnetsimulaties.

Een ander belangrijk punt is, dat het mogelijk is gegevens te verzamelen en deze op een later tijdstip te kunnen analyseren. Daarmee kan het DVCSL ook worden ingezet voor wetenschappelijk botnetsimulaties.

Ook voor deze simulaties geldt dat het een benadering is van de werkelijkheid. Hoe dicht deze benadering bij de werkelijkheid ligt is moeilijk te bepalen. Er zijn heel veel factoren die op het 'echte' Internet voorkomen waarmee rekening zou kunnen worden gehouden. Dit kan een onderwerp zijn voor vervolgonderzoek.

Één van deze vele factoren is de snelheid van het Internet. Kan deze 'echte' snelheid worden benaderd middels netwerkvirtualisatie in het DVCSL. De snelheid tussen twee DVCSL nodes is afhankelijk van de fysieke verbinding tussen deze nodes. De snelheid kan nooit sneller zijn dan de snelheid van de fysieke verbinding. Daarmee is de maximale snelheid gelimiteerd aan de fysieke snelheid, compressie technieken buiten beschouwing latend. Dit is op te lossen door de verticale schaalbaarheid van het DVCSL. De snelheid kan ook naar beneden worden aangepast door de netwerk interfaces van de UML-hosts aan te passen, bij het simuleren van bijvoorbeeld satelliet verbindingen. Maar ook hiervoor kan het DVCSL een mogelijke oplossing bieden, door deze vertraging in te bouwen in het 'plug' proces. Dan is dit op te lossen door de horizontale schaalbaarheid van het DVCSL.

Met deze botnetsimulatie is het functioneel gedrag van een botnet accuraat te simuleren in het DVCSL. Alleen de factor tijd is moeilijker accuraat te simuleren omdat door de virtualisatie deze ook wordt gevirtualiseerd. Hierdoor loopt deze niet gelijk met de 'echte tijd'.

Om een simulatie ook accuraat in tijd te laten zijn, moet deze op een of andere wijze worden mee gesimuleerd. Dit zou gedaan kunnen worden door de virtuele tijd sneller, gelijk of langzamer te laten lopen aan de 'echte tijd'. Door op een creatieve wijze gebruik te maken van de mogelijkheden van het 'Network Time Protocol' (NTP) behoord dit ook tot de mogelijkheden. Dit kan het een onderwerp zijn voor vervolgonderzoek.

Met dit resultaat kan ook de laatste onderzoeksvraag worden beantwoord.

6.5. ONDERZOEKSVRAGEN

In dit hoofdstuk is er een concrete botnetsimulatie casus uitgewerkt en uitgevoerd. Op basis van dit resultaat kan ook de laatste onderzoeksvraag worden beantwoord. Op basis van het antwoord op de drie onderzoeksvragen kan vervolgens de hoofd onderzoeksvraag worden beantwoord.

6.5.1. ONDERZOEKSVRAAG

Door de uitgevoerde botnetsimulatie casus met de HybridBot botnet, en de beantwoorde onderzoeksdeelvragen van het vorige hoofdstuk, kan het antwoord worden gegeven op de laatste onderzoeksvraag.

KUNNEN OP BASIS VAN DE KENMERKEN VAN BOTNETS EN DE KENMERKEN VAN HET DVCSL BOTNETSIMULATIES WORDEN UITGEVOERD MIDDELS NETWERKVIRTUALISATIE?

In dit hoofdstuk is er een concrete botnetsimulatie casus uitgevoerd in het DVCSL. De basis van deze casus is in het vorige hoofdstuk uitgewerkt waarbij de techniek van netwerkvirtualisatie is gebruikt.

Met dit wetenschappelijk empirisch onderzoek is vastgesteld dat door de analyse van de kenmerken van de HybridBot botnet en de kenmerken van het DVCSL kan worden vastgesteld dat botnetsimulaties mogelijk zijn in het DVCSL. Hierna is een succesvolle botnetsimulatie uitgevoerd om de mogelijke beperkingen van het DVCSL vast te stellen. Een van de gevonden oplossingen voor het beperken van het virtuele netwerkverkeer is toegepast in deze botnetsimulatie casus.

6.5.2. HOOFD ONDERZOEKSVRAAG

Door dit wetenschappelijk empirisch onderzoek kan het antwoord worden gegeven op de hoofd onderzoeksvraag.

ZIJN BOTNETSIMULATIES MOGELIJK IN HET DVCSL?

Op basis van de antwoorden op de drie onderzoeksvragen en de uitgevoerd botnetsimulatie casus is vastgesteld dat botnetsimulaties mogelijk zijn in het DVCSL. Er zijn beperkingen gevonden, maar wat de mate van impact hiervan is, is moeilijk te zeggen. Dit is afhankelijk waarvoor het DVCSL wordt ingezet voor het uitvoeren van botnetsimulaties.

7

CONCLUSIE

In dit hoofdstuk worden de onderzoeksvragen zoals deze zijn geformuleerd in Hoofdstuk 2 'Methode' beantwoord aan de hand van het onderzoeksmodel van dit wetenschappelijk empirisch onderzoek.

Na het beantwoorden van de hoofd onderzoeksvraag worden de slotconclusie en de aanbevelingen gegeven van dit onderzoek.

Dit hoofdstuk wordt afgesloten met een reflectie en een discussie.

7.1. ONDERZOEKSVRAGEN

De geformuleerde onderzoeksvragen van Hoofdstuk 2 'Methode' worden beantwoord aan de hand van het onderzoeksmodel van figuur 2.1. In de derde en vierde fase van dit onderzoek zijn de verschillende onderzoeksdeelvragen en onderzoeksvragen in detail beantwoord. In de paragrafen hieronder een samenvatting.

7.1.1. ANALYSE BOTNETS VIRTUALISATIE

In dit deel van het onderzoek is de eerste onderzoeksvraag beantwoord. Zie Hoofdstuk 3 'Botnets' voor de beantwoording van de bijbehorende onderzoeksdeelvragen.

WELKE ONTWERP-PRINCIPES EN ONTWERP-CONDITIES KUNNEN WORDEN VASTGESTELD VOOR DE VIRTUALISATIE VAN BOTNETS?

Door het beschrijven van de attributen en de kenmerken van een botnet en deze vast te leggen in een virtuele netwerktopologie, is het mogelijk de virtuele netwerktopologie van een botnet te vergelijken met een andere virtuele netwerktopologie.

Een virtuele netwerktopologie bestaat uit een structuur van nodes en verbindingen. Door het vastleggen van de attributen en de kenmerken van botnets aan de nodes en verbindingen, kan er een virtuele netwerktopologie worden vastgesteld. Met deze virtuele netwerktopologie kunnen de ontwerp-principes en ontwerp-condities worden vastgesteld die voor de virtualisatie van botnets van belang zijn.

De kenmerken die voor de virtualisatie van botnets van belang zijn, hebben betrekking op de architectuur, topologie en de levenscyclus van botnets.

7.1.2. ANALYSE VIRTUALISATIE DVCSL

In dit deel van het onderzoek is de tweede onderzoeksvraag beantwoord. Zie Hoofdstuk 4 'DVCSL' voor de beantwoording van de bijbehorende onderzoeksdeelvragen.

WELKE ONTWERP-PRINCIPES EN ONTWERP-CONDITIES KUNNEN WORDEN VASTGESTELD VOOR NETWERKVIRTUALISATIE IN HET DVCSL?

De kenmerken van het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) die van belang zijn voor de netwerkvirtualisatie hebben betrekking op de architectuur van het 'Virtuele Computer Security Lab' (VCSL), de architectuur van het DVCSL en de topologie van het DVCSL.

Op basis van deze kenmerken kunnen de ontwerp-principes en ontwerp-condities worden vastgesteld die van belang zijn voor de netwerkvirtualisatie in het DVCSL.

Een belangrijke ontwerp-principe van de architectuur van het DVCSL is dat deze zowel verticaal als horizontaal schaalbaar is, waardoor deze virtuele netwerkomgeving niet gelimiteerd is qua schaalbaarheid. Bij de topologie van het DVCSL is van belang dat de netwerktopologie van de DVCSL nodes bestaat uit een boomstructuur.

Door de kenmerken van de virtuele netwerktopologie te vergelijken met de kenmerken van de nodes en verbindingen van het DVCSL kan worden bepaald of de virtuele netwerktopologie is te implementeren in het DVCSL middels netwerkvirtualisatie. Hierbij moeten de kenmerken van de virtuele nodes en virtuele verbindingen met elkaar overeenkomen. Het resultaat van deze analyse is dat de virtuele netwerktopologie van botnets is te implementeren in het DVCSL. Hiermee is op een wetenschappelijke wijze aangetoond dat botnetsimulatiesmogelijk zijn in het DVCSL..

7.1.3. ANALYSE BOTNETSIMULATIE MOGELIJKHEDEN DVCSL

In dit deel van het onderzoek is de derde onderzoeksvraag beantwoord. Zie Hoofdstuk 5 'Botnetsimulatie' voor de beantwoording van de bijbehorende onderzoeksdeelvragen.

KUNNEN OP BASIS VAN DE KENMERKEN VAN BOTNETS EN DE KENMERKEN VAN HET DVCSL BOTNETSIMULATIES WORDEN UITGEVOERD MIDDELS NETWERKVIRTUALISATIE?

Door de bovenstaande analyse van het DVCSL is vastgesteld dat de belangrijkste voorwaarde waaraan een botnet moet voldoen is dat deze moet kunnen draaien in een Linux omgeving. Op basis van deze voorwaarde is er gezocht naar een geschikte botnets voor dit onderzoek. Daarnaast zijn er een aantal selectiecriteria aangegeven voor de topologie en levenscyclus van een geschikte botnet.

Van de gevonden botnets zijn de attributen en de kenmerken vastgelegd in een virtuele netwerktopologie. Deze zijn vervolgens vergeleken met de netwerkvirtualisatie kenmerken van het DVCSL. Daarnaast zijn er nog een aantal wenselijke eigenschappen aangegeven voor de mate van geschiktheid van een te kiezen botnet. Hiermee is bepaald of de gevonden botnets geschikt zijn om gevirtualiseerd te kunnen worden in het DVCSL.

Op basis van deze analyse is er één botnet gekozen waarmee er een succesvolle botnetsimulatie is uitgevoerd. Hiermee is vastgesteld dat op basis van de analyse van de netwerkvirtualisatie technieken kan worden bepaald of botnetsimulaties mogelijk zijn in het DVCSL.

Tijdens deze botnetsimulatie is er een beperking van het DVCSL naar voren gekomen die een grootschalig gebruik van het DVCSL kan beperken. Als er meerdere DVCSL nodes gebruik maken van dezelfde fysieke netwerkverbinding, dan wordt de virtuele netwerk bandbreedte beperkt met de volgende formule.

$$\text{virtuele netwerk bandbreede} = \frac{\text{fysieke netwerk bandbreede}}{(\text{aantal DVCSL nodes} - 1)}$$

Een van de mogelijke oplossingen hiervoor is om voor elke DVCSL node een router op te nemen in de virtuele netwerktopologie.

7.1.4. BOTNETSIMULATIE UITVOEREN IN HET DVCSL

Door in dit wetenschappelijk empirisch onderzoek beantwoorde onderzoeksvragen en de uitgevoerde botnetsimulatie in Hoofdstuk 6 'DVCSL botnetsimulatie casus' kan het antwoord worden gegeven op de hoofd onderzoeksvraag.

ZIJN BOTNETSIMULATIES MOGELIJK IN HET DVCSL?

Door het uitvoeren van een succesvolle botnetsimulatie casus, is met een praktijkvoorbeeld vastgesteld dat botnetsimulaties mogelijk zijn in het DVCSL.

De basis voor deze botnetsimulatie is een grondige analyse van de ontwerp-principes en ontwerp-condities van botnets en de ontwerp-principes en ontwerp-condities van het DVCSL vooraf gegaan. Hiervoor is de techniek gebruikt van netwerkvirtualisatie zoals de virtuele netwerktopologie.

Op basis van deze analyse is een botnetsimulatie uitgevoerd en zijn de resultaten hiervan geanalyseerd. Hiermee is op een wetenschappelijke wijze vastgesteld dat botnetsimulaties mogelijk zijn in het DVCSL, door gebruik te maken van de netwerkvirtualisatie technieken. Daarnaast kan worden vastgesteld dat met deze onderzoeksmethode in theorie kan worden bepaald of simulaties mogelijk zijn in het DVCSL. Door het werkelijk uit te kunnen voeren van deze simulaties in het DVCSL, kan worden vastgesteld dat deze simulaties ook in de praktijk mogelijk zijn. Hiermee heeft de gekozen onderzoeksmethode zich zowel in praktijk als in theorie bewezen.

Door als laatste een praktijk voorbeeld uit te werken in de vorm van een botnetsimulatie casus, is hiermee het ultieme bewijs geleverd dat botnetsimulaties mogelijk zijn in het DVCSL. Voor de gevonden beperking van het virtuele netwerkverkeer is er een mogelijke oplossing gevonden. In de botnetsimulatie casus is er voor elke DVCSL node een router UML-host opgenomen in de virtuele netwerktopologie.

7.2. BEPERKINGEN

In verschillende fasen van dit wetenschappelijk onderzoek zijn er beperkingen naar voren gekomen van de inzetbaarheid van het DVCSL bij botnetsimulaties. Hieronder zijn de belangrijkste beperkingen beschreven.

7.2.1. TYPE BOTNETS

Zoals hierboven is aangegeven moeten botnets voldoen aan de netwerkvirtualisatie kenmerken van het DVCSL. Waarbij er aan een belangrijke ontwerp-conditie moet worden voldaan, dat een botnet in een Linux omgeving moet kunnen draaien.

Voor dit onderzoek zijn er nog een aantal selectie criteria gebruikt zoals op het gebied van botnet topologie en levenscyclus. Deze hoeven voor andere onderzoeken niet te gelden. Waar wel rekening mee moet worden gehouden dat de resources binnen een DVCSL omgeving beperkt zijn. Hierdoor zijn botnets die veel geheugen en processor capaciteit nodig zijn minder geschikt om in het DVCSL te virtualiseren. Om de geschiktheid van een botnet voor virtualisatie in het DVCSL te bepalen zijn een aantal wenselijke eigenschappen geformuleerd waarbij de complexiteit, aanpasbaarheid en de beschikbare informatie van een botnet zijn mee genomen.

Hiermee worden heel veel botnets uitgesloten voor simulatie in het DVCSL. Misschien zijn er mogelijkheden om deze beperking te verminderen door bijvoorbeeld botnet emulatie technieken toe te passen zoals deze welk zijn te simuleren in het DVCSL. Dit kan een onderwerp zijn voor vervolg onderzoek.

7.2.2. BESTURINGSSYSTEMEN

Het DVCSL maakt in de basis gebruik van 'User Mode Linux' (UML), wat alleen kan draaien op Linux gebaseerde omgevingen. Hierdoor is er vaak een extra virtualisatie laag nodig om het DVCSL te kunnen gebruiken [VK09; Haa+11; Vra+11; HV12; Haa+14]. Dit gaat altijd ten koste van de totale beschikbare systeemresources, waarbij er duidelijke verschillen zichtbaar zijn in de gebruikte virtualisatie technieken.

Het aantal op Linux gebaseerde doelwit systemen neemt de laatste tijd behoorlijk toe, met name in de mobiele apparatuur [Zha+12]. In potentie kunnen alle Linux systemen als doelwit systeem worden gesimuleerd in het DVCSL. De niet Linux gebaseerde doelwit systemen kunnen nog niet worden gesimuleerd in het DVCSL.

7.2.3. PERFORMANCE NETWERKVERKEER

Bij de analyse van het netwerkverkeer tijdens de botnetsimulatie is er een performance beperking vastgesteld op het virtuele netwerkniveau. In het geval dat drie DVCSL nodes gebruik maken van dezelfde fysieke verbinding, wordt elk pakketje van het virtuele netwerkniveau twee maal verzonden op het fysieke netwerkniveau. Dit is conform de werking van het DVCSL, maar kan grote performance problemen geven op het virtuele netwerkniveau. Hierdoor is het gebruik van de centrale autoriteit minder geschikt voor het uitvoeren van botnetsimulaties[Haa+11].

Dit probleem heeft in eerste instantie betrekking op de horizontale schaalbaarheid van het DVCSL. Door meerdere DVCSL nodes toe te voegen aan de DVCSL omgeving die gebruik maken van dezelfde fysieke verbinding, heeft dit een negatieve invloed op de virtuele netwerk bandbreedte. De processor en geheugen capaciteit kan hier mee wel groeien, maar de netwerk capaciteit verslechterd. De virtuele netwerk bandbreedte kan worden uitgebreid door gebruik te maken van snellere verbindingen, maar deze verticale schaalbaarheid van het fysieke netwerk is ook beperkt. Aan de horizontale en verticale schaalbaarheid van het DVCSL zitten grenzen, maar door bewust met deze technieken om te gaan, hoeft dit niet negatief uit te pakken en kan het juist een groot voordeel zijn.

Door de virtuele netwerktopologie zoveel mogelijk gelijk te houden met de fysieke netwerktopologie en voor elke DVCSL node een router UML-host op te nemen in de virtuele netwerktopologie kan dit probleem tot een minimum worden beperkt.

Er is nog veel onderzoek mogelijk op deze terreinen en kan daarmee een onderwerp zijn voor vervolgonderzoek.

7.3. DISCUSSIE

Dit wetenschappelijk onderzoek heeft aangetoond dat botnetsimulaties mogelijk zijn in het DVCSL. Hiermee is aangetoond dat het DVCSL ingezet kan worden voor bestaand wetenschappelijk onderzoek op het gebied van botnetsimulaties.

Hierdoor is het mogelijk om botnetsimulaties te kunnen uitvoeren met minder systeem resources in minder tijd. Het configureren van een virtueel computer lab kost veel minder tijd als het opzetten van een fysiek computer lab [BB07; Cal+10; EG11]. Het DVCSL kan mogelijk worden ingezet bij de bestrijding van botnets.

Dit onderzoek is een eerste stap in dit nieuwe onderzoeksgebied. Een aantal van de hierboven genoemde beperkingen zijn misschien gemakkelijk op te lossen. Maar er zullen ook een aantal beperkingen zijn die niet zijn op te lossen.

Het DVCSL bied vele voordelen en mogelijkheden. Maar of deze opwegen tegen de beperkingen die het DVCSL nu nog heeft, hangt sterk af van het toepassingsgebied. Hiervoor is nog veel wetenschappelijk onderzoek nodig. In de volgende paragraaf worden deze verder toegelicht.

Dit onderzoek bouwt voort op bestaand onderzoek op het gebied van wetenschappelijk onderwijs [VK09]. De botnetsimulatie casus in het DVCSL kan worden gebruikt voor wetenschappelijk onderwijs naar botnets en de bestrijding en beperking hiervan.

7.4. TOEKOMSTIG WERK

Naast de al eerder aangegeven onderwerpen voor vervolgonderzoek voor het verbeteren van het DVCSL, kan er onderzoek worden gedaan in combinatie met eerdere studies en bestaand wetenschappelijk onderzoek.

- Het uitbreiden van netwerk emulatie omgevingen die gebaseerd zijn op NetKit zoals bijvoorbeeld AutoNetKit[Ngu+11; Kni+13]. Door gebruik te maken van de DVCSL technieken, kunnen deze omgevingen horizontaal schaalbaar worden gemaakt. Hierdoor kunnen nog grotere virtuele netwerken worden gegenereerd en geëmuleerd.
- Het uitbreiden van digitale studiebegeleiding in combinatie met de beschreven botnetsimulatie casus [Haa+14].Studenten kunnen dan botnetsimulaties uitvoeren en ondersteuning krijgen van digitale studiebegeleiding.
- Onderzoek op het gebied van mobiele botnets, die gebruik maken van smartphones en tablets [Zha+12]. Kan het DVCSL mogelijk worden ingezet voor mobiele botnetsimulaties. Een groot deel van deze apparatuur is namelijk gebaseerd op het Linux besturingssysteem.
- Naast de al eerder genoemde verbeteringen aan het DVCSL zijn er ook nog andere onderzoeksgebieden die extra functionaliteit kunnen bieden aan het DVCSL. Onderzoek naar het versleutelen van het virtuele netwerkverkeer. Of in combinatie met de centrale autoriteit gebruikt te maken van multicasting virtuele netwerken [Haa+11].

7.5. REFLECTIE

Het doen van wetenschappelijk onderzoek heb ik met veel plezier en toewijding gedaan. Ondanks dat het niet altijd even gemakkelijke was. Door kleine tegenslagen en mijn dyslexie ben ik trots op het behaalde resultaat. Ook met een studiebeperking is het mogelijk gebleken te kunnen afstuderen. De extra tijd die hiervoor van tevoren voor gereserveerd was, bleek achteraf onvoldoende te zijn geweest.

Het vinden van een geschikte botnet voor het uitvoeren van de botnetsimulaties was heel lastig en een groot afbreuk risico van dit onderzoek. Het vinden van een bestaande operationele botnet is onmogelijk gebleken. Maar gelukkig waren er ook twee synthetische Open Source botnets beschikbaar. Één is hiervan gebruikt voor de leerzame botnetsimulaties.

Terug kijkend is het een zware maar zeer leerzame periode geweest. Nieuwe kennis en vaardigheden eigengemaakt op het gebied van netwerkvirtualisatie, botnetsimulaties en de technieken van het DVCSL. Maar door toewijding en veel steun vanuit mijn omgeving kan ik terug kijken op een geslaagde ontdekkingsreis.

7.6. BSD LICENSE

Alle gemaakte scripts en programma code zijn beschikbaar gesteld onder de BSD licentie voorwaarden.

BIBLIOGRAFIE

ACADEMISCHE ARTIKELEN

[Bai+09] Michael Bailey, Evan Cooke, Farnam Jahanian, Yunjing Xu en Manish Karir. "A survey of botnet technology and defenses". In: *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications and Technology*. IEEE, 2009, p. 299–304. ISBN: 0769535682.

[BB07] Paul Barford en Mike Blodgett. "Toward botnet mesocosms". In: *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*. USENIX Association, 2007, p. 6–6. URL: http://dl.acm.org/citation.cfm?id=1323128.1323134.

[Cal+10] Joan Calvet, Carlton R Davis, José M Fernandez, Jean-Yves Marion, Pier-Luc St-Onge, Wadie Guizani, Pierre-Marc Bureau en Anil Somayaji. "The case for in-the-lab botnet experimentation: creating and taking down a 3000-node botnet". In: *Proceedings of the 26th Annual Computer Security Applications Conference*. ACSAC '10. New York, NY, USA: ACM, 2010, p. 141–150. ISBN: 978-1-4503-0133-6. DOI: 10.1145/1920261.1920284. URL: http://doi.acm.org/10.1145/1920261.1920284.

[CB08] NM Mosharaf Kabir Chowdhury en Raouf Boutaba. "A Survey of Network Virtualization". In: *Computer Networks* 54.Elsevier (2008), p. 862–876.

[CJM05] Evan Cooke, Farnam Jahanian en Danny McPherson. "The zombie roundup: Understanding, detecting, and disrupting botnets". In: *Proceedings of the USENIX SRUTI Workshop*. Deel 39. 2005, p. 44.

[EG11] Nathan S Evans en Christian Grothoff. "Beyond Simulation: Large-Scale Distributed Emulation of P2P Protocols". In: *Proceedings of the 4th Workshop on Cyber Security Exper- imentation and Test (CSET)*. USENIX Association, 2011.

[Haa+11] Jens Haag, Tobias Horsmann, Stefan Karsch en Harald Vranken. "A Distributed Virtual Computer Security Lab with Central Authority". In: *Proceedings of the 11th Computer Science Education Research Conference*. CSERC '11. Open Universiteit, Heerlen, 2011, p. 89–95. ISBN: 978 90 358 1987 0. URL: http://dl.acm.org/citation.cfm?id=2043594.2043602.

[Haa+14] Jens Haag, Christian Witte, Stefan Karsch, Harald Vranken en Marko van Eekelen. "An exercise assistant for practical networking courses". In: *Proceedings of the 6th International Conference on Computer Supported Education*. 2014, p. 97–104.

[HV12] Jens Haag en Harald Vranken. "A Virtual Computer Lab As Learning Environment For Networking and Security Courses". In: *Proceedings of the 3rd Annual International Conference on Computer Science Education, Innovation and Technology. CSEIT 2012*. 2012, p. 61–68.

[Kni+13] Simon Knight, Hung Nguyen, Olaf Maennel, Iain Phillips, Nickolas Falkner, Randy Bush en Matthew Roughan. “An automated system for emulated network experimentation”. In: *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 2535378: ACM, 2013, p. 235–246. DOI: 10.1145/2535372.2535378. URL: http://dl.acm.org/citation.cfm?doid=2535372.2535378.

[Liu+11] Shangdong Liu, Jian Gong, Wang Yang en Ahmad Jakalan. “A survey of botnet size measurement”. In: *Networking and Distributed Computing (ICNDC), 2011 Second International Conference on*. IEEE, 2011, p. 36–40. ISBN: 1457704072.

[Ngu+10] Hung Nguyen, Matthew Roughan, Simon Knight, Nick Falkner, Olaf Maennel en Randy Bush. “How to build complex, large-scale emulated networks”. In: *TridentCom (6th: 2010: Berlin, Germany)* (2010). ISSN: 3642178502.

[Ngu+11] Hung Nguyen, Matthew Roughan, Simon Knight, Nick Falkner, Olaf Maennel en Randy Bush. “How to build complex, large-scale emulated networks”. In: *Testbeds and Research Infrastructures. Development of Networks and Communities*. Springer, 2011, p. 3–18. ISBN: 3642178502.

[PR08] Maurizio Pizzonia en Massimo Rimondini. “Netkit: easy emulation of complex networks on inexpensive hardware”. In: *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks and communities*. 1390585: ICST (Institute for Computer Sciences, Social-Informatics en Telecommunications Engineering), 2008, p. 1–10. URL: http://dl.acm.org/citation.cfm?id=1390576.1390585.

[Raj+06] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose en Andreas Terzis. “A multifaceted approach to understanding the botnet phenomenon”. In: *Proceedings of the 6th ACM SIGCOMM Conference on Internet measurement (IMC)*. IMC ’06. 2006: ACM, 2006, p. 41–52. ISBN: 1-59593-561-4. DOI: 10.1145/1177080.1177086. URL: http://doi.acm.org/10.1145/1177080.1177086.

[RMG13] Rafael A. Rodríguez-Gómez, Gabriel Maciá-Fernández en Pedro García-Teodoro. “Survey and taxonomy of botnet research through life-cycle”. In: *ACM Computer Survey*. 45.4 (2013), p. 1–33. ISSN: 0360-0300. DOI: 10.1145/2501654.2501659. URL: http://dl.acm.org/citation.cfm?doid=2501654.2501659.

[Sil+13] Sérgio SC Silva, Rodrigo MP Silva, Raquel CG Pinto en Ronaldo M Salles. “Botnets: A survey”. In: *Computer Networks* 57.2 (2013), p. 378–403. ISSN: 1389-1286.

[TA11] Amit Kumar Tyagi en G Aghila. “A wide scale survey on botnet”. In: *International Journal of Computer Applications* 34.9 (2011), p. 10–23.

[VK09] Harald Vranken en Herman Koppelman. “A virtual computer security lab for distance education”. In: *Proceedings of the 5th IASTED European Conference on Internet and Multimedia Systems and Applications*. Cambridge, UK: Acta Press, 2009, p. 21–27.

[Vra+11] Harald Vranken, Jens Haag, Tobias Horsmann en Stefan Karsch. “A Distributed Virtual Computer Security Lab”. In: *Proceedings of the 3rd International Conference on Computer Supported Education*. Deel Vol. 1. SciTePress, 2011, p. 110–119.

[Whi+02] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb en Abhijeet Joglekar. “An integrated experimental environment for distributed systems and networks.” In: *Proceedings of the 5th Symp. on Operating systems design and implementation (OSDI)*. Deel 36. ACM, 2002, p. 255–270. DOI: 10.1145/844128.844152. URL: http://doi.acm.org/10.1145/844128.844152.

[Yan+04] T Andrew Yang, Kwok-Bun Yue, Morris Liaw, George Collins, Jayaraman T Venkatraman, Swati Achar, Karthik Sadasivam en Ping Chen. “Design of a distributed computer security lab”. In: *Journal of Computing Sciences in Colleges* 20.1 (2004), p. 332–346. ISSN: 1937-4771.

[Zha+12] Shuang Zhao, Patrick P. C. Lee, John C. S. Lui, Xiaohong Guan, Xiaobo Ma en Jing Tao. “Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service”. In: *Proceedings of the 28th Annual Computer Security Applications Conference*. 2420968: ACM, 2012, p. 119–128. DOI: 10.1145/2420950.2420968. URL: http://dl.acm.org/citation.cfm?doid=2420950.2420968.

TECHNISCHE DOCUMENTATIE

[Bég15] François Bégin. *BYOB code*. Web Page. 2015. URL: http://public.francois.warpmail.net.

[Cro15] Cross. *Hybrid Botnet System 1.0*. Web Page. 2015. URL: https://packetstormsecurity.com/files/84727/Hybrid-Botnet-System-1.0.html.

[Igu15] Julien ‘Kartoch’ Iguchi-Cartigny. *Netkit-NG Homepage*. Web Page. 2015. URL: https://netkit-ng.github.io.

[Ngu+11] Hung Nguyen, Matthew Roughan, Simon Knight, Nick Falkner, Olaf Maennel en Randy Bush. “How to build complex, large-scale emulated networks”. In: *Testbeds and Research Infrastructures. Development of Networks and Communities*. Springer, 2011, p. 3–18. ISBN: 3642178502.

[UML15] UML. *User Mode Linux (UML)*. Web Page. 2015. URL: http://user-mode-linux.sourceforge.net.

[VD15] Piet Verschuren en Hans Doorewaard. *Het ontwerpen van een onderzoek*. Web Page. 2015. URL: http://www.ontwerpenvaneenonderzoek.nl.

ACRONIEMEN

C&C 'Commando en Controle', is de commando en controle omgeving waarmee een botnet aangestuurd en onderhouden kan worden. 16, 42, 46, 50, 51, 53, 54, 59, 99–102, 106, 108, *zie ook* botnet & C&C Systeem

C&C Systeem 'Commando en Controle Systeem', is het commando en controle systeem waarmee een botnet aangestuurd en onderhouden kan worden. x, xii, 10, 12, 14, 18, 19, 44–52, 54, 57, 58, 60, 64, 99, 100, 104, 108, *zie ook* botnet & C&C

DCSL 'Distributed Computer Security Lab', is een eerst ontwerp van een gedistribueerde computer security lab. 21, *zie ook* VCSL & DVCSL

DDoS 'Distributed Denial of Service', is een gedistribueerde aanval om een dienst of systeem te overspoelen met bevragingen. 56, 57, *zie ook* DoS

DHCP 'Dynamic Host Configuration Protocol'. 28, 29, 31–33, 49, 58, 84, 96, 97

DNS 'Domain Name System'. 28, 29, 31–33, 49, 56–61, 63, 64, 84, 89, 90, 96, 108

DoS 'Denial of Service' , is een aanval om een dienst of systeem te overspoelen met bevragingen. *zie ook* DDoS

DVCSL 'Gedistribueerd Virtuele Computer Security Lab', is een gedistribueerd versie van de VCSL, waarmee studenten kunnen samenwerken met opzetten en beveiligen van computer netwerken in een afgesloten virtuele omgeving. x–xiii, 1–8, 15, 16, 20–39, 42–45, 48–50, 53–59, 64, 65, 67–71, 80–96, 98–100, 104–109, *zie ook* DCSL & VCSL

FTP 'File Transfer Protocol'. 45, 103

GH 'Ghost Host'. 24, 34

HTTP 'Hypertext Transfer Protocol'. 52

IP 'Internet Protocol'. 35, 61, 90, 93, 94

IPv4 'Internet Protocol versie 4'. 29, 89

IPv6 'Internet Protocol versie 6'. 29

ISP 'Internet service provider'. 56–59, 61, 62, 64

LAMP 'Linux Apache MySQL Perl/PHP', is een webserver architectuur gebaseerd op open-source componenten. 42, *zie ook* Linux, Apache, MySQL, Perl & PHP

MTU 'Maximum Transmission Unit'. 35, 93, 94

NTP 'Network Time Protocol'. 65

NVE 'Network Virtualization Environment', bestaat uit een fysieke netwerk topologie in combinatie met een virtuele netwerk topologie bestaande uit virtuele nodes en virtuele verbindingen. 17, *zie ook* node, verbinding & netwerktopologie

OU 'Open Universiteit'. x–xiii, 1, 2, 5, 7, 20, 64

P2P 'Peer-to-Peer', is een netwerk tussen twee nodes waarbij er een virtueel netwerk over een bestaande netwerk kan worden getransporteerd. 21, *zie ook* node

Perl 'Practical Extraction and Report Language', is een Multiplatform programmeertaal. 41, 42, 44, 46, 47, *zie ook* LAMP

PHP PHP: Hypertext Preprocessor, is een Multiplatform scripttaal. 41, 42, 44, 46, 97, *zie ook* LAMP

RBE 'Remote Brige Endpoint'. 24, 26, 35

SMTP 'Simple Mail Transfer Protocol'. 28, 31

SYN 'Synchronous TCP flood'. 103

TCP 'Transmission Control Protocol'. 59, 60, 103

UDP 'User Datagram Protocol'. 24, 26, 28, 30, 38, 59, 82, 94, 103

UML 'User Mode Linux', is een micro Linux besturingssysteem waarbij de Linux Kernel in User Mode draait. 21–27, 29, 31–37, 58, 60, 69, 81–86, 89, 93, 99, *zie ook* Linux

VCSL 'Virtuele Computer Security Lab', is een software omgeving waarin studenten kunnen experimenteren met het opzetten en beveiligen van computer netwerken in een afgeloten virtuele omgeving. VCSL is gebaseerd op UML. x–xiii, 2, 3, 20–25, 37, 67, *zie ook* DVCSL & UML

VPN 'Virtual Private Network'. 21, 35, 93

WOORDENLIJST

Apache Apache is een OpenSource webserver. 42, 44, 46, 49, 97, 101, *zie ook* LAMP

AutoNetKit is hulpmiddel om binnen netkit geautomatiseerd een netwerkomgeving te genereren. 23, 70, *zie ook* NetKit

bot (komt van robot) is een computerprogramma dat op een autonome manier taken kan uitvoeren die normaal door mensen uitgevoerd worden. x, xii, 1, 10, 12, 14, 16, 18, 19, 41, 42, 44–52, 59, 60, 62, 64, 99–101, 103, 104, 108, 109

bot systeem is een slachtoffer systeem van een bot. 12, 18, 19, 45, 48–50, *zie ook* bot & doelwit systeem

botmaster is de persoon die het botnet beheerd en aanstuurt, ook welk botherder genoemd. x, xii, 1, 10, 12, 14, 18, 19, 41, 42, 45–52, 57, 59, 99–102, 106, *zie ook* botnet

botnet bestaat uit een network van een bot die bestuurd wordt door een botmaster. x–xiii, 1–8, 10–12, 14–21, 25, 28, 29, 36–51, 54–57, 59, 60, 63–71, 84, 98, 99, 102, *zie ook* Internet, bot & botmaster

botnetsimulatie is simulatie van een botnet in een afgesloten netwerk omgeving. xi, xiii, 3–6, 8, 9, 15, 16, 20, 33, 39, 40, 43, 44, 48–51, 54–60, 63–65, 67–71, 84, 93, 97–99, 104, 107–109, *zie ook* botnet

BYOB is een OpenSource, platform onafhankelijke botnet gebaseerd op Java architectuur. 41–44, 98, 99, *zie ook* bot, botnet & Java

doelwit systeem is een potentieel slachtoffer systeem van een bot. 11, 12, 14, 18, 19, 45, 47, 48, 50–52, 58, 69, *zie ook* bot

Fedora is een vrij beschikbare Linux distributie van de grootste Linux distributeur. 25, *zie ook* Linux

guest is binnen computer virtualisatie het gastbesturingssysteem. *zie ook* virtualisatie

host wordt gebruikt als synoniem voor een computersysteem. 17, 21, 22, 24, 27–29, 31, 32, 34–36, 88, 90, 92, 107, *zie ook* virtualisatie

host machine wordt binnen de computer virtualisatie, het gastheer systeem mee bedoeld, waarop de computer virtualisatie software draait. 21, 34, 88, 90, *zie ook* virtualisatie & host

hostnaam is de naam van het systeem. 27, 29, 31, 33, 80, 83, 86, 89–91, *zie ook* host

hub verbind aangesloten computers met elkaar door het netwerkverkeer naar alle aangesloten computers te sturen. 54, *zie ook* switch

HybridBot is een OpenSource, platform onafhankelijke botnet gebaseerd op de LAMP architectuur. 41–57, 59, 60, 62, 64, 65, 84, 98–102, 104, 106, 108, *zie ook* bot, botnet & LAMP

Internet is een globaal openbaar netwerk waar computersystemen over de gehele wereld met elkaar kunnen communiceren. x–xiii, 1–4, 6, 11, 25, 29, 31, 32, 36, 37, 40, 46, 48, 56–62, 64, 93, 96, 99, 102

IP-adres is het adres waarop een systeem bereikbaar is op het IP netwerk, zoals het Internet. 26, 27, 29–33, 58, 60, 64, 88–90, 93, 108, *zie ook* IP, IPv4, IPv6, hostnaam & Internet

Java is een objectgeoriënteerde programmeertaal die ontwikkeld is door Sun Microsystems. 41, 42, 44

Linux is een besturingssysteem die gebaseerd is op Unix. xi, xiii, 20, 22, 25, 26, 36–39, 55, 67–70, 84, 85, 95, 101, *zie ook* Fedora

MySQL is een OpenSource relationele databases. 42, 44, 46, 49, 54, 97, 100–102, *zie ook* LAMP

NetKit is virtuele omgeving gebaseerd op Linux waarmee experimentele computer netwerken opgezet kunnen worden. 20–23, 30, 36, 37, 70, 82, *zie ook* Linux & NetKit-ng

NetKit-ng is de volgende generatie van NetKit. 21, 25, 84, 86, 87, 95–98, *zie ook* NetKit

netwerk is een structuur van hosts en routers die samen een geheel vormen. 64, 68, 69, 88–90, 94, 96, *zie ook* host, router, Internet & netwerkvirtualisatie

netwerktopologie is een structuur van nodes en verbindingen tussen de nodes die samen een geheel vormen. x–xiii, 6, 7, 17–19, 22, 23, 25, 38, 54–58, 66–69, *zie ook* node & verbinding

netwerkvirtualisatie is het simuleren van een netwerk in een gevirtualiseerde omgeving. x–xiii, 1–8, 10, 16, 17, 19, 20, 36–38, 48, 64, 65, 67, 68, 71, *zie ook* netwerktopologie

node is een knooppunt waar verbindingen samen komen, dit zijn vaak routers, maar kunnen ook individuele systemen zijn. x–xiii, 6, 7, 17–19, 21–30, 32–38, 45, 48–50, 53–55, 57–59, 64, 66–69, 81–86, 88–96, 98, 104, 106, 107, *zie ook* verbinding

router is verantwoordelijk voor het routeren van het netwerkverkeer binnen een netwerk. xi, xiii, 17, 22, 23, 31–33, 49, 50, 52, 54, 55, 57–62, 64, 68, 69, 99, *zie ook* node

softwarerobot is software applicatie die automatische en automoon opdrachten uitvoerd zonder dat iemand hier specifiek opdracht voor geeft. x, xii, 1, 10, *zie ook* bot

switch verbind aangesloten computers met elkaar door het netwerkverkeer naar alleen de aangegeven computers te sturen. 22, 26, 54, *zie ook* hub

TAP-netwerk is een virtueel point-to-point netwerk tussen twee systemen, een dedicated netwerk hub. 49, 96, *zie ook* netwerk & hub

taxonomie wordt de indeling van individuele entiteiten in groepen mee bedoeld. 11

topologie hiermee wordt de samenhang van de verschillende componenten beschreven. x–xiii, 2, 7, 10–12, 14–19, 25, 38, 40–43, 48, 66, 67, 69, *zie ook* netwerktopologie

UML-host User Mode Linux machine. 21–26, 30–33, 44, 48–50, 52, 54, 57–59, 61–64, 68, 69, 80–82, 84, 93–96, 98, 99, *zie ook* Linux, UML & host

UML-netwerk User Mode Linux virtueel netwerk. 21, 22, 54, *zie ook* Linux, UML & netwerk

UML-switch User Mode Linux virtueel switch. 22–24, 26, 34, 36, 54, 57, 82, 94, *zie ook* Linux, UML & switch

verbinding is een connectie tussen twee knooppunten waar nodes met elkaar verbonden worden en is een representatie van een netwerk. x–xiii, 6, 7, 17–19, 37, 38, 45, 48, 54, 66, 67, *zie ook* node

virtualisatie is het op een softwarematige wijze implementeren van vaak fysieke componenten, zoals bijvoorbeeld PC en server hardware. 2, 4–6, 10, 11, 16, 19, 21, 22, 36–38, 66, 69, *zie ook* netwerkvirtualisatie

Windows is een besturingssysteem die gebaseerd is op Microsoft Windows. 37, 39

A

DVCSL CONFIGURATIE

In deze bijlage zijn alle technische details beschreven van het 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) [VK09; Haa+11; Vra+11; HV12; Haa+14]. Als eerste wordt de gebruikte DVCSL structuur beschreven. Daarna worden de technische details van de DVCSL omgevingen beschreven. Als laatste worden de verschillende DVCSL configuraties behandeld.

De DVCSL structuur en de DVCSL omgevingen en configuraties vormen samen de DVCSL configuratie. Voor het ontwerpen en realiseren van deze DVCSL configuratie zijn een aantal algemene uitgangspunten gedefinieerd om de kwaliteit hiervan te waarborgen. Deze algemene uitgangspunten zijn als volgt geformuleerd:

- één centrale configuratie,
- iedere UML-host heeft een eigen configuratie op basis van zijn hostnaam,
- documentatie in de gemaakte scripts zijn in het Engels,
- de gemaakte componenten zijn zoveel mogelijk platform onafhankelijk,
- en er moet zo veel mogelijk gebruik gemaakt worden van geautomatiseerde handelingen.

A.1. DVCSL STRUCTUUR

De DVCSL structuur is in feite niets anders dan een set aan configuratie bestanden met een aantal scripts om de verschillende DVCSL omgevingen en configuraties te kunnen beheren.

In de volgende paragrafen wordt de gebruikte DVCSL structuur verder uitgewerkt.

A.1.1. GEDEELDE CONFIGURATIE STRUCTUUR

Voor dit onderzoek wordt geen gebruik gemaakt van de Centrale autoriteit zoals deze is beschreven in paragraaf 4.2.2 [Haa+11]. Maar voor dit onderzoek wordt er gebruik gemaakt van een centrale gedeelde configuratie structuur.

De centrale gedeelde configuratie structuur moet beschikbaar zijn voor alle systemen die onderdeel zijn van dit wetenschappelijk empirisch onderzoek. Er zijn verschillende diensten die deze functionaliteit kunnen bieden, zoals Google Drive, Dropbox, Microsoft OneDrive en Apple iCloud Drive.

Op een zelf te bepalen centrale plaats moet de hier beschreven centrale gedeelde configuratie structuur beschikbaar worden gemaakt voor de betreffende DVCSL nodes. Voor dit onderzoek is gekozen om gebruikt te maken van de Apple iCloud Drive. De centrale gedeelde configuratie is beschikbaar gemaakt onder 'iCloud Drive/OU/T75317'.

Door deze centrale gedeelde configuratie structuur beschikbaar te stellen aan de DVCSL nodes middels een gedeelde folder. Kan hiermee is deze centrale gedeelde configuratie structuur ook beschikbaar binnen de DVCSL node. In de DVCSL node moet deze gedeelde folder beschikbaar worden gesteld aan de 'User Mode Linux' (UML)-gebruikerzodat deze ook beschikking heeft over deze centrale gedeelde configuratie structuur. Deze configuratie moet in de home directory van UML-gebruiker beschikbaar worden gemaakt onder ' /T75317'.

HUIDIGE STRUCTUUR GEDEELDE CONFIGURATIE

Hieronder is de gedeelde configuratie weergegeven zoals deze nu is opgebouwd in de directory ' /T75317', zie listing A.1. De 'dvcsl-*' directories hebben een vergelijkbare structuur, hierin bevinden zich dat verschillende DVCSL omgevingen. Zie paragraaf 4.4 voor de verschillende DVCSL configuraties.

Daarnaast is er nog een 'dvcsl.make' script gemaakt die op basis van de gekozen configuratie in 'dvcsl.conf' de juiste 'dvcsl-*' configuratie naar de lokale ' /dvcsl' directory kopieert.

Listing A.1: T75317 directory

```
/home/uml/T75317
  ├──bin
  ├──dvcsl-analyse
  ├──dvcsl-basis
  ├──dvcsl-basis-dhcp
  ├──dvcsl-basis-dhcp-dns
  ├──dvcsl-basis-dns
  ├──dvcsl-basis-remotes
  ├──dvcsl-botnet-casus
  ├──dvcsl-botnet-php
  ├──dvcsl-botnet-test
  ├──dvcsl-demo
  ├──dvcsl-update
  ├──dvcsl.conf
  └──sbin
```

BASIS STRUCTUUR

De basis structuur van elke DVCSL configuratie bestaat uit drie UML-lab omgevingen die afzonderlijk gestart kunnen worden middels het 'lstart' commando. Hierdoor worden de benodigde UML-systemen gestart die bij deze UML-lab configuratie behoren. Middels 'lhalt' kunnen deze weer worden gestopt. Daarnaast kunnen deze omgevingen ook weer worden opgeschoond door het 'lclean' commando.

Door op de verschillende UML-host omgevingen de juiste UML-lab omgeving te starten en deze met elkaar te koppelen door het juiste 'plug' commando is de DVCSL omgeving actief. Om dit proces te automatiseren zijn er drie scripts gemaakt die deze acties voor elke DVCSL node op de juiste wijze uitvoerd.

De 'dvcsl.start' commando start de juiste UML-lab omgeving op voor de betreffende DVCSL node. En start daarna het juiste 'plug' proces(sen) op die voor de DVCSL node actief moeten zijn, zoals hierboven beschreven. De configuratie die nodig is voor het starten van de juiste 'plug' proces(sen) zijn in de 'lab.conf' configuratie opgenomen. Zie listing A.2 voor een voorbeeld van deze configuratie.

Op basis van de 'DVCSL_' parameters kunnen de verschillende 'plug' processen worden gestart. De 'DVCSL_REMOTES' parameter bepaald hoeveel 'plug' processen er gestart worden en waar deze aan gekoppeld moeten worden.

Listing A.2: DVCSL lab.conf

```
LAB_DESCRIPTION="DVCSL Basis - VCSL101"
LAB_VERSION="1.0"
LAB_AUTHOR="J.H.A. Jonkman"
LAB_EMAIL="jha.jonkman@studie.ou.nl"
LAB_WEB="t75317.jeroenjonkman.nl"

DVCSL_REMOTES="vcsl102 vcsl103"
DVCSL_PORT_OFFSET="30000"
DVCSL_FSOCKED="vhub_uml_LAN.cnct"

machines="pc101a pc101b"
pc101a[0]=LAN
pc101b[0]=LAN
```

De 'DVCSL_PORT_OFFSET' bepaald de start offset van de benodigde 'User Datagram Protocol' (UDP) poorten die door de 'plug' gebruikt worden. Zie tabel A.1 Plug poort berekening voor de berekening van de bron en doel UDP poorten voor het 'plug' proces.

Tabel A.1: Plug poort berekening

Omschrijving	**Bron poort**	**Doel poort**
Host	vcsl101	vcsl102
Hostnaam (DNS bevraging)	vcsl101.local.dvcsl.edu	vcsl102.local.dvcsl.edu
IP-adres (DNS bevraging)	192.168.2.101	192.168.2.102
DVCSL_PORT_OFFSET	3000	3000
LOCAL_PORT_OFFSET Laatste deel van het lokale IP-adres	101	102
REMOTE_PORT_SESSION Laatste cijfer van het remote IP-adres	3	1
Plug UDP poort	$3000 + (101 \times 10) + 3$ **31013**	$3000 + (102 \times 10) + 1$ **31021**

Formule: $PORTNUMMER = DVCSL\_PORT\_OFFSET + (LOCAL\_PORT\_OFFSET \times 10) + REMOTE\_PORT\_SESSION$

De 'DVCSL_FSOCKED' parameter geeft aan welke UML-socket er gebruikt moet worden om een verbinding te maken met de UML-switch van de UML-lab omgeving.

A.1.2. DVCSL CONFIGURATIES

Er zijn verschillende DVCSL configuraties gebruikt tijdens dit onderzoek. De structuur van deze DVCSL configuraties zijn allemaal aan elkaar gelijk qua structuur. Ook de onderliggende DVCSL node configuraties zijn qua structuur vaak ook hetzelfde.

Binnen de verschillende DVCSL node configuraties is er een NetKit omging geconfigureerd. Deze bestaat uit de verschillende UML-host configuraties en de 'lab' configuratie zoals deze in listing A.2 is weergegeven.

Als voorbeeld configuratie wordt hieronder alleen de DVCSL configuratie 'dvcsl-basis' beschreven, alle andere DVCSL configuraties hebben dezelfde structuur.

DVCSL CONFIGURATIE - DVCSL-BASIS

In listing A.3 is de DVCSL configuratie 'dvcsl-basis' weergegeven. Elke DVCSL node heeft hierin een eigen UML-lab configuratie structuur. Op basis van de hostnaam van de DVCSL node kan de juiste UML-lab configuratie worden geactiveerd, gestart en gestopt.

Listing A.3: DVCSL configuratie - dvcsl-basis

```
dvcsl-basis
├── vcsl101
│   ├── lab.conf
│   ├── pc101a
│   │   └── etc
│   │       ├── network
│   │       │   └── interfaces
│   │       └── resolv.conf
│   ├── pc101a.startup
│   ├── pc101b
│   │   └── etc
│   │       ├── network
│   │       │   └── interfaces
│   │       └── resolv.conf
│   └── pc101b.startup
├── vcsl102
│   ├── lab.conf
│   ├── pc102a
│   │   └── etc
│   │       ├── network
│   │       │   └── interfaces
│   │       └── resolv.conf
│   ├── pc102a.startup
│   ├── pc102b
│   │   └── etc
│   │       ├── network
│   │       │   └── interfaces
│   │       └── resolv.conf
│   └── pc102b.startup
└── vcsl103
    ├── lab.conf
    ├── pc103a
    │   └── etc
    │       ├── network
    │       │   └── interfaces
    │       └── resolv.conf
    ├── pc103a.startup
    ├── pc103b
    │   └── etc
    │       ├── network
    │       │   └── interfaces
    │       └── resolv.conf
    └── pc103b.startup
```

Het activeren, starten en stoppen van deze configuraties wordt verderop beschreven.

DVCSL CONFIGURATIE - DVCSL.CONF

In listing A.4 is de gedeelde 'dvcsl.conf' configuratie te zien waarin de verschillende DVCSL configuraties zijn aangegeven. Door het '#' voor één van de regels weg te laten wordt deze DVCSL configuratie geselecteerd.

Listing A.4: dvcsl.conf

```
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-update
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-analyse
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-demo
DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-basis
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-basis-dns
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-basis-dhcp
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-basis-dhcp-dns
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-botnet-php
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-botnet-test
#DVCSL_CURRENT=${DVCSL_SHARE}/dvcsl-botnet-casus
```

DVCSL CONFIGURATIES

De volgende DVCSL configuraties zijn te activeren.

- dvcsl-analyse
  Deze DVCSL omgeving is gebruikt voor de analyse van de botnets.
- dvcsl-basis
  Dit is de basis DVCSL omgeving die als basis is gebruikt voor de andere configuraties.
- dvcsl-basis-dns
  In deze DVCSL is de 'Domain Name System' (DNS) netwerkservice geconfigureerd.
- dvcsl-basis-dhcp
  In deze DVCSL is de 'Dynamic Host Configuration Protocol' (DHCP) netwerkservice geconfigureerd.
- dvcsl-basis-dhcp-dns
  In deze DVCSL is zowel de DHCP en de DNS netwerkservice geconfigureerd.
- dvcsl-botnet-php
  In deze DVCSL is HybridBot botnet geconfigureerd.
- dvcsl-botnet-test
  In deze DVCSL is HybridBot botnet geconfigureerd en wordt automatisch gestart.
- dvcsl-botnet-casus
  In deze DVCSL is HybridBot botnet geconfigureerd voor de botnetsimulatie casus.
- dvcsl-demo
  Dit is een demo DVCSL omgeving, met maar één UML-host per DVCSL node.
- dvcsl-update
  DVCSL configuratie voor het bijwerken van de NetKit-ng omgeving.

A.1.3. DVCSL NODE CONFIGURATIES

In de onderstaande paragrafen wordt de DVCSL node configuratie beschreven. Hierin zijn de UML-gebruiker en de bijbehorende home directory uitgewerkt zoals deze binnen alle DVCSL nodes moeten bestaan.

DVCSL GEBRUIKERSNAAM

Voor dit onderzoek is de gebruikersnaam 'uml' gekozen voor de UML-gebruiker, waarbij het wachtwoord gelijk is gehouden aan de naam van de gebruiker. Ook heeft deze gebruiker 'sudo' rechten, zodat deze gebruiker ook de systeeminstellingen kan aanpassen.

DVCSL GEBRUIKER LOGIN

Na het starten van de DVCSL node kan er worden ingelogd als UML-gebruiker. Hierna verschijnt het bureaublad, met links onderin het scherm het menu. Rechts naast het menu is een overzicht van de vier bureaubladen te zien, daarnaast is een snel koppeling om een Linux console te kunnen starten.

DVCSL GEBRUIKER CONFIGURATIE

In de home directory van de UML-gebruiker is de NetKit-ng omgeving beschikbaar gesteld in ' /netkit-ng'. De gedeelde configuratie is middels een symbolische link beschikbaar gesteld in ' /T75317'. Deze verwijst naar de lokatie binnen de DVCSL node waar de gedeelde

configuratie beschikbaar is gesteld. Op basis van de inhoud van deze directory wordt de ingestelde DVCSL configuratie naar de ' /dvcsl' gekopieerd.

De home directory van de UML-gebruiker is weergegeven in listing A.5.

Listing A.5: UML home directory

```
/home/uml
  ├──dvcsl
  ├──netkit-ng
  └──T75317 -> /media/psf/iCloud/OU/T75317/dvcsl/
```

Na het activeren van de 'dvcsl-basis' configuratie ziet de ' /dvcsl' er uit als in listing A.6 is aangegeven.

Listing A.6: UML dvcsl directory

```
/home/uml/dvcsl/
  ├──bin
  ├──dvcsl.conf
  ├──html
  ├──sbin
  ├──vcsl101
  ├──vcsl102
  └──vcsl103
```

A.2. DVCSL COMMANDO'S

In deze paragraaf worden de verschillende DVCSL commando's uitgelegd. In tabel A.2 DVCSL commando's zijn deze commando's weergegeven en in welke volgorde deze uitgevoerd kunnen worden. Ook is hierin aangegeven welke alternatieve stappen er zijn.

Tabel A.2: DVCSL commando's

	Actie	Commando	Volgende stap	Alternatieve stap
Stap 1	Activeren	'dvcsl.make'	stap 2	stap 7
Stap 2	Starten	'dvcsl.start'	stap 3	stap 4/5/7
Stap 3	Klaar voor gebruik			stap 4/5/7
Stap 4	Stoppen	'dvcsl.stop'	stap 6	stap 5/7
Stap 5	Afbreken	'dvcsl.crash'	stap 6	stap 7
Stap 6	Gestopt			stap 2/7
Stap 7	Schonen	'dvcsl.clean'	stap 8	
Stap 8	Gestopt en geschoond			stap 1/2

Als de UML-gebruiker is aangemeld op de DVCSL node kunnen de bovenstaande commando's worden ingevoerd in een Linux console op de DVCSL node.

A.2.1. DVCSL MAKE COMMANDO

Door het uitvoeren van het 'dvcsl.make' commando wordt de DVCSL configuratie die in de 'dvcsl.conf' is geselecteerd geactiveerd en gekopieerd naar ' /dvcsl'. In listing A.7

is weergegeven hoe de 'dvcsl-basis' configuratie is geactiveerd middels het '`dvcsl.make`' commando.

De bestaande DVCSL configuratie in ' /dvcsl' wordt overschreven, ook als deze nog actief is.

Listing A.7: Het dvcsl.make commando

```
[uml@vcsl101 ~]$ dvcsl.make
Reading /home/uml/dvcsl/../T75317/dvcsl.conf...
Activeer DVCSL omgeving: dvcsl-basis...
[uml@vcsl101 ~]$
```

A.2.2. DVCSL INIT COMMANDO

Met het '`dvcsl.init`' commando wordt gecontroleerd of de basis instellingen van de DVCSL node juist zijn ingesteld. Dit hoeft de UML-gebruiker niet zelf te doen, maar de onderstaande commando's maken hier automatisch gebruik van. Hierdoor hoeft deze controle niet apart te worden opgenomen in alle onderstaande commando's, maar kan deze op één plek worden onderhouden.

A.2.3. DVCSL START COMMANDO

Met het '`dvcsl.start`' commando wordt de geactiveerde DVCSL configuratie gestart. In de ' /dvcsl' directory staan de verschillende DVCSL deel configuraties, voor elke DVCSL node een aparte NetKit-ng omgeving. Op basis van de hostnaam van de DVCSL node wordt de juiste NetKit-ng omgeving gestart door eerst het '`lstart`' commando uit te voeren en daarna de benodigde 'plug' processen te starten.

In listing A.8 is dit goed te zien, eerst wordt de NetKit-ng lab gestart en daarna de twee 'plug' processen.

Listing A.8: Het dvcsl.start commando

```
[uml@vcsl101 ~]\$ dvcsl.start
vcsl101.local.dvcsl.edu has ip-address 192.168.2.101.

======================== Starting lab ============================
Lab directory: /home/uml/dvcsl/vcsl101
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL Basis - VCSL101
==================================================================
You chose to use parallel startup.
Starting "pc101b"...
Starting "pc101a"...

The lab has been started.
==================================================================

Starting Remote DVCSL plug from vcsl101 to vcsl102
DVCSL_DOMAIN is not set in the LAB config file, using system domain from DNS.
Starting plug with "DVCSL_PLUG_PARAMETER=--sip 192.168.2.101 --sport 31012 --dip 192.168.2.102 --dport 31021 --
      fsock vhub_uml_LAN.cnct"
Starting Remote DVCSL plug from vcsl101 to vcsl103
Starting plug with "DVCSL_PLUG_PARAMETER=--sip 192.168.2.101 --sport 31013 --dip 192.168.2.103 --dport 31031 --
      fsock vhub_uml_LAN.cnct"
```

A.2.4. DVCSL STOP COMMANDO

Met het '`dvcsl.stop`' commando wordt de gestarte DVCSL configuratie gestopt. Dit wordt gedaan door eerst in de juiste directory het '`lhalt`' commando uit te voeren en daarna alle 'plug' processen te stoppen.

In listing A.9 is te zien hoe de NetKit-ng lab wordt gestopt, het stoppen van de 'plug' processen is niet te zien omdat dit niet wordt gelogd.

Listing A.9: Het dvcsl.stop commando

```
[uml@vcsl101 ~]\$ dvcsl.stop
vcsl101.local.dvcsl.edu has ip-address 192.168.2.101.
Stopping DVCSL on

======================== Halting lab ============================
Lab directory: /home/uml/dvcsl/vcsl101
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL Basis - VCSL101
=================================================================
Halting virtual machine "pc101a" (PID 4108) owned by uml [....... ]
Halting virtual machine "pc101b" (PID 4111) owned by uml [....... ]
Removing readyfor.test...

Lab has been halted.
=================================================================
```

A.2.5. DVCSL CRASH COMMANDO

Met het '`dvcsl.crash`' commando wordt de DVCSL configuratie afgebroken. Dit wordt gedaan door eerst in de juiste directory het '`lcrash`' commando uit te voeren en daarna alle 'plug' processen te stoppen.

In listing A.10 is te zien hoe de NetKit-ng lab wordt afgebroken, het stoppen van de 'plug' processen is niet te zien omdat dit niet wordt gelogd.

Listing A.10: Het dvcsl.crash commando

```
[uml@vcsl101 ~]$ dvcsl.crash
vcsl101.local.dvcsl.edu has ip-address 192.168.2.101.
Craching DVCSL on

======================== Crashing lab ===========================
Lab directory: /home/uml/dvcsl/vcsl101
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL Basis - VCSL101
=================================================================

============= Crashing virtual machine "pc101a" (PID 9791) =========
Virtual machine owner: uml
Virtual machine mconsole socket: /home/uml/netkit-ng/mconsole/pc101a/mconsole
Crashing... done.
Removing filesystem "/home/uml/dvcsl/vcsl101/pc101a.disk"... done.

============= Crashing virtual machine "pc101b" (PID 9788) =========
Virtual machine owner: uml
Virtual machine mconsole socket: /home/uml/netkit-ng/mconsole/pc101b/mconsole
Crashing... done.
Removing filesystem "/home/uml/dvcsl/vcsl101/pc101b.disk"... done.
Removing readyfor.test...

Lab has been crashed.
```

A.2.6. DVCSL CLEAN COMMANDO

Met het '`dvcsl.clean`' commando wordt de DVCSL configuratie afgebroken en geschoond. Dit wordt gedaan door eerst in de juiste directory het '`lcrash`' en '`lclean`' commando uit te voeren. Daarna worden alle 'plug' en 'uml_switch' processen gestopt.

In listing A.11 is dit te zien hoe de NetKit-ng lab wordt afgebroken en geschoond, het stoppen van de 'plug' en 'uml_switch' processen is niet te zien omdat dit niet wordt gelogd.

Listing A.11: Het dvcsl.clean commando

```
[uml@vcsl101 ~]$ dvcsl.clean
vcsl101.local.dvcsl.edu has ip-address 192.168.2.101.
Cleaning DVCSL on

======================== Crashing lab ============================
Lab directory: /home/uml/dvcsl/vcsl101
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL Basis - VCSL101
==================================================================
vcrash: no virtual machine named "pc101a" exists for user uml.
vcrash: no virtual machine named "pc101b" exists for user uml.
Removing readyfor.test...

Lab has been crashed.
==================================================================


====================== Cleaning up lab ===========================
Lab directory: /home/uml/dvcsl/vcsl101
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL Basis - VCSL101
==================================================================
Cleaning up lab inside "/home/uml/dvcsl/vcsl101"...

Cleaning completed.
==================================================================

plug: geen proces gevonden
uml_switch: geen proces gevonden
```

A.3. DVCSL OMGEVINGEN

Voor dit onderzoek is gebruik gemaakt van een DVCSL omgeving van drie DVCSL nodes op één host machine. Dit wordt de 'lokale DVCSL omgeving' genoemd, zoals afgebeeld in figuur 4.5. Daarnaast is er gebruik gemaakt van een DVCSL omgeving waarbij er drie host machines zijn gebruikt, voor elke DVCSL node één. Dit wordt de 'remote DVCSL omgeving' genoemd, zoals afgebeeld in figuur 4.6.

De basis van deze DVCSL omgevingen zijn gelijk, zoals deze zijn weergegeven in deze figuren. De meest linkse DVCSL nodes heeft twee DVCSL verbindingen, één met de middelste DVCSL node en één met de rechter DVCSL node. De beide andere DVCSL nodes hebben op hun beurt weer een DVCSL verbinding met de meer linkse DVCSL node. Voor elke van deze vier verbindingen is er één 'plug' proces nodig. Omdat zowel de IP-adressen als de poortnummers van de vier 'plug' processen geconfigureerd moeten worden is gekozen om de poortnummers middels een functie te bepalen. De berekening hiervan is weergegeven in tabel A.1 Plug poort berekening.

In de volgende twee paragrafen wordt kort de belangrijkste instellingen beschreven.

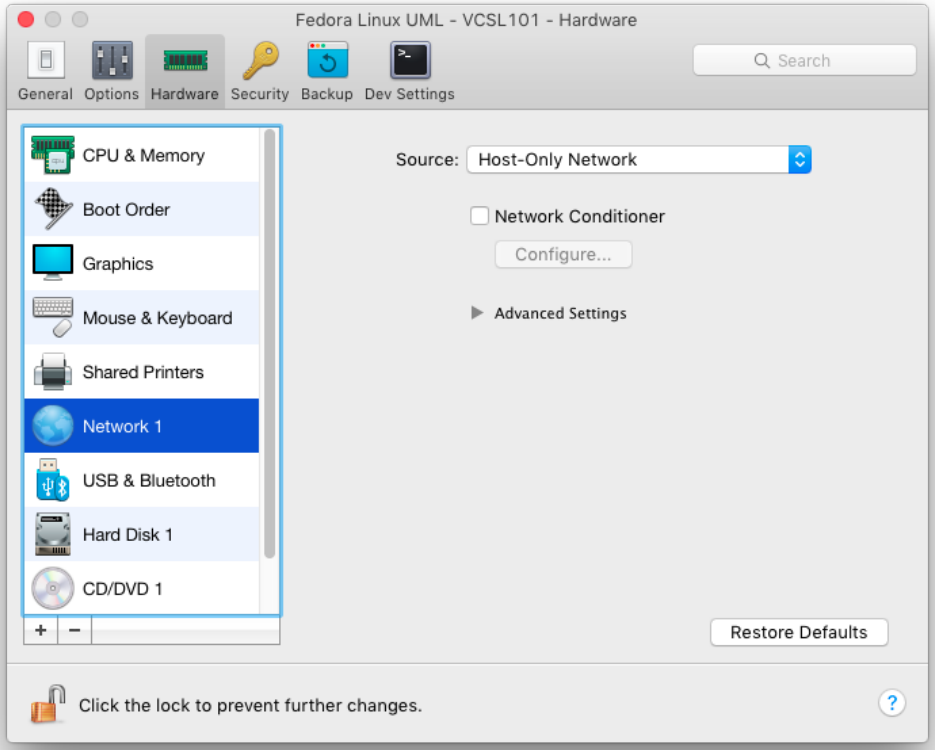
A.3.1. LOKALE DVCSL OMGEVING

Deze DVCSL omgeving maakt gebruikt van één host machine met daarop de drie DVCSL nodes. Er is geen vastgestelde start en stop volgorde voor de verschillende DVCSL nodes. Het activeren van de verschillende DVCSL nodes is beschreven in paragraaf A.1.2.

Alle DVCSL nodes op de host machine moeten met elkaar kunnen communiceren. Hiervoor is er vaak een 'host-gebonden' netwerk beschikbaar. Dit kan worden aangepast in de configuratie van de virtuele machine in de computer simulatie software. Een voorbeeld van deze configuratie is weergegeven in figuur A.1.

Bij de voor dit onderzoek gebruikte computer virtualisatie software wordt dit gedaan door voor het netwerk de 'Host-Only Network' te kiezen. In de instellingen van deze computer virtualisatie software is dit netwerk zo geconfigureerd dat deze gebruik maakt van het '192.168.2.0/24' netwerksegment.

Als de DVCSL node aan de juiste netwerkbron is gekoppeld kan deze worden gestart.



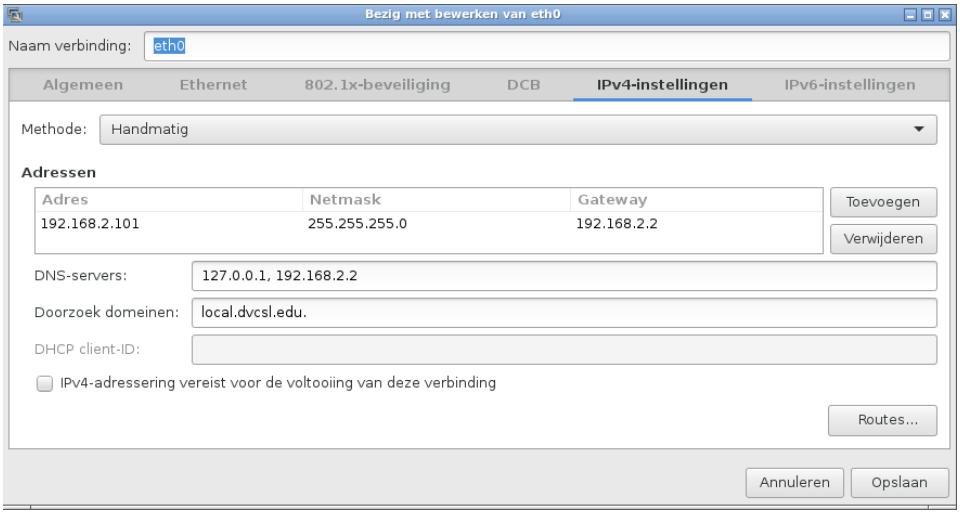
Figuur A.1: Lokale DVCSL netwerk

Er is geen vastgestelde start en stop volgorde voor de verschillende DVCSL nodes. Als een DVCSL node is gestart en niet van het juiste IP-adres is voorzien, dan kan dat via de UML-gebruiker worden aangepast. Via het 'menu -> Voorkeuren -> Netwerkverbindingen' kunnen de netwerkinstellingen worden aangepast. Door vervolgens de 'eth0' interface te selecteren en deze te bewerken, wordt het scherm van figuur A.2 zichtbaar en kunnen de 'Internet Protocol versie 4' (IPv4) instellingen worden aangepast.

De belangrijke instellingen hierbij zijn, dat de gekozen 'Methode' op 'Handmatig' staat. Dan kan het IP-adres en de bijbehorende 'Netmask' en 'Gateway' worden ingesteld.

Voor de DNS-server is het belangrijk dat '127.0.0.1' hier ook in staat omdat op de gebruikte DVCSL node ook een DNS netwerkservice draait. Deze zorgt er voor dat de hostnaam van de DVCSL node op de juiste hostnaam wordt gezet op basis van hier ingestelde IP-adres. Om de juiste hostnaam te kunnen opzoeken moet het 'Doorzoek domeinen' op 'local.dvcsl.edu.' worden gezet.

Als alle instellingen zijn aangepast en bewaard moet de DVCSL node worden herstart, zodat ook de juiste hostnaam kan worden gezet tijdens het starten van de DVCSL node.



Figuur A.2: Lokale DVCSL interface instellingen

A.3.2. REMOTE DVCSL OMGEVING

Deze DVCSL omgeving maakt gebruik van drie verschillende host machines met elk hun eigen DVCSL node. Er is geen vastgestelde start en stop volgorde voor de verschillende host machines en DVCSL nodes. Het activeren van de verschillende DVCSL nodes is beschreven in paragraaf A.1.2.

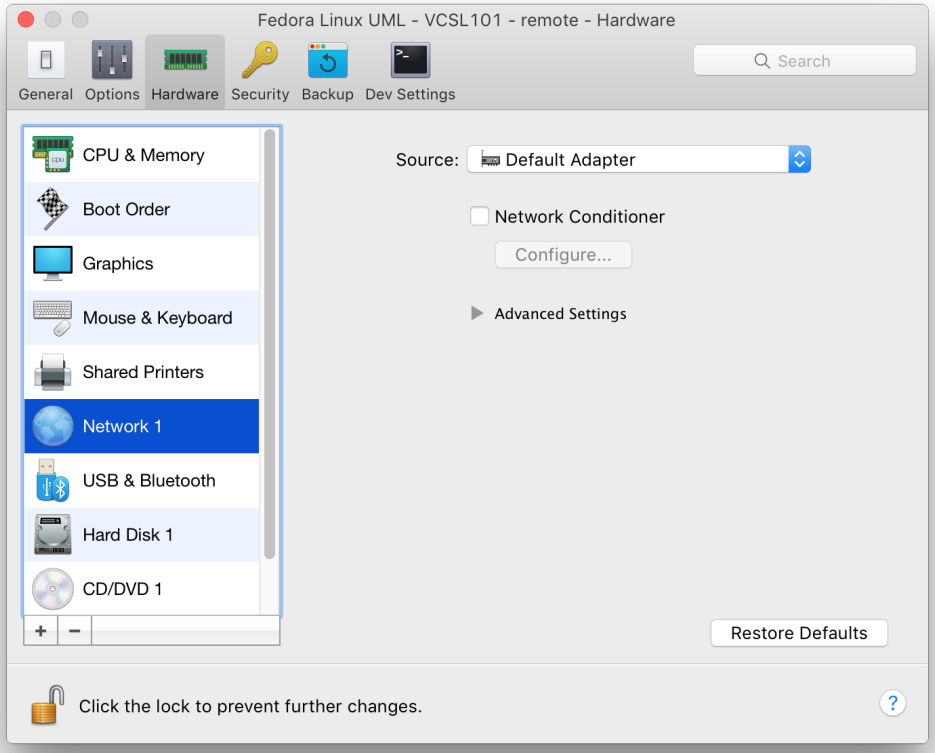
Alle host machines moeten elkaar kunnen bereiken via een gezamenlijk netwerk. Voor dit voorbeeld is gekozen om hiervoor een apart netwerk voor te gebruiken, het '172.16.248.0/24' netwerksegment. Hiervoor is er vaak een 'gedeelde netwerkinterface' beschikbaar. Dit kan worden aangepast in de configuratie van de virtuele machine in de computer simulatie software. Een voorbeeld van deze configuratie is weergegeven in figuur A.3.

Bij de voor dit onderzoek gebruikte computer virtualisatie software wordt dit gedaan door voor het netwerk de 'Default Adapter' te kiezen. Hierdoor wordt de ingestelde 'Internet Protocol' (IP) configuratie van de DVCSL node direct actief op de standaard netwerkinterface van de host machine. Hierdoor is de DVCSL node direct toegankelijk via het gezamenlijk netwerk. Het gebruik hiervan moet wel zijn afgestemd met de netwerkbeheerder, want vaak kan dit niet zomaar.

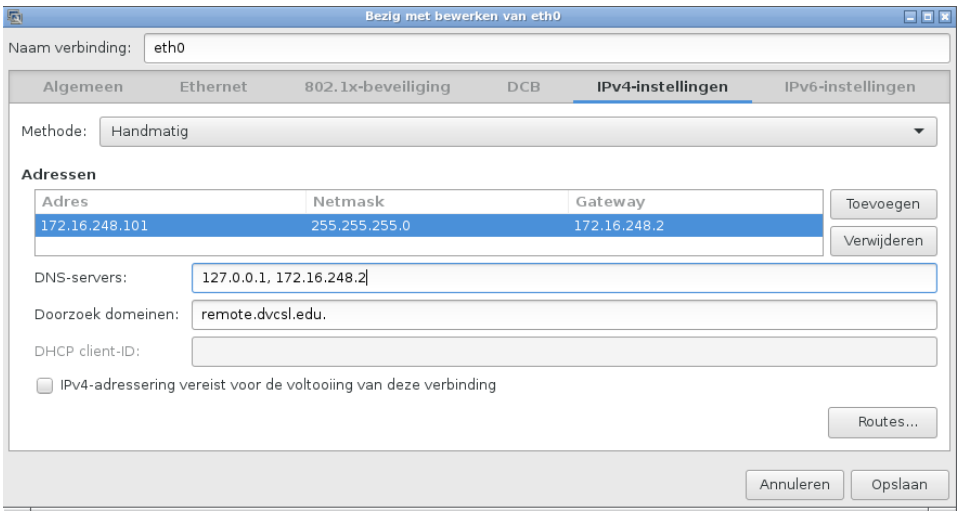
Als de DVCSL node aan de juiste netwerkbron is gekoppeld kan deze worden gestart. Als de DVCSL node is gestart en niet van het juiste IP-adres is voorzien, dan kan deze worden aangepast zoals hiervoor is beschreven. Alleen zijn de instellingen anders, deze zijn in figuur A.2 weergegeven.

Om de juiste hostnaam te kunnen opzoeken moet het 'Doorzoek domeinen' op 'remote.dvcsl.edu.' worden gezet. Als er andere IP-adressen worden gebruik voor de DVCSL nodes kan er voor gekozen worden deze op te nemen in de '/etc/hosts' van alle gebruikte DVCSL nodes. Of door de centrale DNS netwerkservice deze te laten beantwoorden, door deze hiervoor van de juiste configuratie te voorzien. Ook kan de DNS netwerkservice van alle gebruikte DVCSL nodes hierop worden aangepast.

Als alle instellingen zijn aangepast en bewaard moet de DVCSL node worden herstart, zodat ook de juiste hostnaam kan worden gezet tijdens het starten van de DVCSL node.



Figuur A.3: Remote DVCSL netwerk



Figuur A.4: Remote DVCSL interface instellingen

A.4. DVCSL PROBLEMEN EN OPLOSSINGEN

Tijdens het onderzoek naar de werking van het DVCSL zijn er twee problemen gevonden. Daarnaast is er nog een verbetering doorgevoerd in het 'plug' proces. De technische details van deze aanpassing zijn beschreven in de volgende drie paragrafen.

A.4.1. PROCESSOR BELASTING HOST MACHINE

Het 'plug' proces zorgde er voor dat de processor van de host machine volledig was belast als alle drie DVCSL nodes hierop actief waren. Dit is opgelost door een twee tal software aanpassingen in de 'thread.c' programma code.

In listing A.12 is de programma code weergegeven van de 'dispatch' functie zoals deze was voor de aanpassing en in listing A.13 met de aanpassing. Alle regels van listing A.12 zijn terug te vinden in listing A.13. Wat hier aan is toegevoegd is een wacht procedure, regel 2 t/m 10.

Listing A.12: Plug dispatch origineel

```
1 tv.tv_sec = 1;
2 tv.tv_usec = 0;
3 while(1)
4 {
5     rcv = recv(*(threadParm->fd), &pack, sizeof(pack), 0);
6     if (rcv > 0)
7     {
8         //Send received data to (remote) IP
9         long snd = sendto(*(threadParm->udp_fd), &pack, rcv, 0, (
              struct sockaddr*)threadParm->addr, sizeof(*(threadParm->
              addr)));
```

Listing A.13: Plug dispatch nieuw

```
1  while(true)
2  {
3      /* Waiting for data */
4      FD_ZERO(&read_fdset);
5      FD_SET(*(threadParm->uml_fd), &read_fdset);
6      tv.tv_sec = 1;
7      tv.tv_usec = 1;
8      int r = select(*(threadParm->uml_fd)+1, &read_fdset, NULL, NULL,
           &tv);
9      if (r > 0) /* Data ready to be read */
10     {
11         long rcv = recv(*(threadParm->uml_fd), &pack, sizeof(pack),
               0);
12         if (rcv > 0)
13         {
14             //Send received data to (remote) IP
15             long snd = sendto(*(threadParm->udp_fd), &pack, rcv, 0,
                   (struct sockaddr*)threadParm->addr, sizeof(*(
                   threadParm->addr)));
```

In listing A.14 is de programma code weergegeven van de 'inject' functie zoals deze was voor de aanpassing en in listing A.15 met de aanpassing. Een aantal regels van listing A.14 zijn terug te vinden in listing A.15. Een aantal zijn aangepast waardoor er wel wordt gewacht op nieuwe netwerkpakketjes.

Listing A.14: Plug inject origineel

```
while(1)
{
  FD_ZERO(&read_fdset);
  FD_SET(sock, &read_fdset);
  r = select(sock+1, &read_fdset, NULL, NULL, &tv);
  if (FD_ISSET(sock, &read_fdset))
  {
    FD_CLR(sock, &read_fdset);
    rcv = recvfrom(sock, &pack, sizeof(pack), 0, (struct sockaddr*)
        &addr, &len);
    if (rcv > 0)
    {
    snd = sendto(*(threadParm->fd), &pack, rcv, 0, (struct sockaddr
        *) threadParm->fd_addr, sizeof(*(threadParm->fd_addr)));
```

Listing A.15: Plug inject nieuw

```
while(true)
{
    FD_ZERO(&read_fdset);
    FD_SET(*(threadParm->udp_fd), &read_fdset);
    tv.tv_sec = 1;
    tv.tv_usec = 1;
    int r = select(*(threadParm->udp_fd)+1, &read_fdset, NULL, NULL
        , &tv); //Socket ready to read?
    if (r > 0)
    {
        long rcv = recv(*(threadParm->udp_fd), &pack, sizeof(pack),
            0);
        if (rcv >= 0)
        {
        long snd = sendto(*(threadParm->uml_fd), &pack, rcv, 0, (
            struct sockaddr*) threadParm->uml_addr, sizeof(*(
            threadParm->uml_addr)));
```

A.4.2. NETWERKPAKKET FRAGMENTATIE OP DE HOST MACHINE

Tijdens de analyse van de werking van het DVCSL waarbij de pakketgrote boven de 1458 bytes uit komt, is vast gesteld dat er netwerkpakket fragmentatie optreedt. Hierdoor wordt het te grootte netwerkpakket opgedeeld in kleinere netwerkpakketjes.

Dit is geen vreemd gedrag en komt vrijwel altijd voor als er gebruik gemaakt wordt van 'Virtual Private Network' (VPN) technieken. De oplossing hiervoor is het gebruik van een aangepaste 'Maximum Transmission Unit' (MTU) grootte op het IP niveau. Deze instellingen zijn ook toegepast bij de botnetsimulatie casus van Hoofdstuk 6 'DVCSL botnetsimulatie casus'. Hierbij is voor het 'fictieve' Internet de MTU grootte op 1458 is gezet.

In listing A.16 is deze fragmentatie te zien waarbij een het gevirtualiseerd netwerkpakket is opgedeeld in twee netwerkpakket op het netwerkniveau van de DVCSL node. Voor dit voorbeeld is de 'dvcsl-basis' DVCSL configuratie gebruikt. Het gebruikte commando in de UML-host is '`ping`' met de opties '-s 1500 <IP-adres>'. Voor het weergeven van de gegevens in listing A.16 is het commando '`sudo tcpdump`' gebruikt men de opties '-nni eth0', uitgevoerd als UML-gebruiker.

Hierin is goed te zien dat er eerst een te grootte netwerk pakket is verstuurd, waarbij er wordt aangegeven dat er een verkeerde lengte is gebruikt. Vervolgens een bericht 'ip-proto-

17' waarmee wordt aangegeven dat het ontbrekende deel nog komt. Daarna wordt het ontbrekende deel van het virtuele netwerkpakket verstuurd. Voor elk van de virtuele netwerkpakken wordt dit herhaald.

Listing A.16: De dvcsl-demo netwerk interface instellingen

```
13:24:55.269552 IP 192.168.2.103.37712 > 192.168.2.101.31013: UDP, bad length 1514 > 1472
13:24:55.269572 IP 192.168.2.103 > 192.168.2.101: ip-proto-17
13:24:55.269589 IP 192.168.2.103.37712 > 192.168.2.101.31013: UDP, length 62
13:24:55.269706 IP 192.168.2.101.35750 > 192.168.2.102.31021: UDP, bad length 1514 > 1472
13:24:55.269765 IP 192.168.2.101 > 192.168.2.102: ip-proto-17
13:24:55.269785 IP 192.168.2.101.35750 > 192.168.2.102.31021: UDP, length 62
```

In listing A.17 is er een alternatieve MTU grootte voor de UML-host ingesteld. Hierbij wordt voor de 'eth0' netwerkinterface de MTU op 1459 gezet. Deze configuratie is onderdeel van de 'dvcsl-demo' DVCSL configuratie.

Listing A.17: De dvcsl-demo netwerk interface instellingen

```
uto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
  address 10.0.0.111
  netmask 255.255.255.0
  gateway 10.0.0.1
  mtu 1459
```

In listing A.18 is er geen fragmentatie meer te zien. Wel worden de virtuele netwerkpakketten nog opgedeeld, maar dit is het standaard gedrag van het IP verkeer. Voor dit voorbeeld is de 'dvcsl-demo' DVCSL configuratie gebruikt. De gebruikte commando's zijn hetzelfde als hier boven zijn gebruikt.

Listing A.18: De dvcsl-demo netwerk interface instellingen

```
13:30:04.158837 IP 192.168.2.103.54979 > 192.168.2.101.31013: UDP, length 1466
13:30:04.158862 IP 192.168.2.103.54979 > 192.168.2.101.31013: UDP, length 110
13:30:04.158993 IP 192.168.2.101.49239 > 192.168.2.102.31021: UDP, length 1466
13:30:04.159060 IP 192.168.2.101.49239 > 192.168.2.102.31021: UDP, length 110
```

A.4.3. OPNIEUW VERZENDEN DVCSL VERKEER

Het 'plug' proces verzendt eenmalig een UDP bericht, als dit niet goed gaat wordt hier niets mee gedaan. Dit geldt zowel voor het verzenden naar een andere DVCSL node, als voor het verzenden naar de UML-switch. Bij een zware netwerkbelasting is de kans groot dat een netwerkpakketje niet wordt verzonden. Als deze niet wordt verzonden, dan wordt het virtuele netwerkpakketje ook niet verzonden. Hierdoor raken er dus virtuele netwerkpakketjes kwijt en moeten deze mogelijk opnieuw worden verzonden. Dit kan leiden tot vertragingen of zelfs het afbreken van verbindingen in het de virtuele netwerk.

Dit probleem is ontdekt tijdens het analyseren van het netwerkverkeer en het aanzetten van logging in de programma code van het 'plug' proces.

In listing A.13 en is de programma code weergegeven van de 'dispatch' functie zoals deze was voor deze verbetering en in listing A.19 de verbetering. Wat hieraan is een lus toegevoegd waarin vijf keer wordt geprobeerd om het virtuele netwerkpaketje te verzenden. In listing A.13 is regel 12 t/m 15 vervangen met regel 1 t/m 8 van listing A.19.

In listing A.15 en is de programma code weergegeven van de 'inject' functie zoals deze was voor deze verbetering en in listing A.20 de verbetering. Wat hieraan is een lus toegevoegd

waarin vijf keer wordt geprobeerd om het virtuele netwerkpaketje te verzenden. In listing A.15 is regel 11 t/m 13 vervangen met regel 1 t/m 8 van listing A.20.

De keuze van vijf keer is arbitrair, maar of het nu twee of tien keer moet zijn is moeilijk te zeggen. Maar er moet zeker een eindig aantal maal zijn, om het 'plug' proces niet in een oneindige lus te laten lopen.

Listing A.19: Plug dispatch verbeterd

```
1 if (rcv > 0)
2 {
3     //Send received data to (remote) IP in 5 attempts
4     int retry = 0;
5     long snd = -1;
6     while( snd == -1 && retry++ < 5 ) {
7        snd = sendto(*(threadParm->udp_fd), &pack, rcv, 0, (struct
             sockaddr*)threadParm->addr, sizeof(*(threadParm->addr)));
8    }
```

Listing A.20: Plug inject verbeterd

```
1 if (rcv >= 0)
2 {
3     //Send received data to uml_switch in 5 attempts
4     int retry = 0;
5     long snd = -1;
6     while( snd == -1 && retry++ < 5 ) {
7        snd = sendto(*(threadParm->uml_fd), &pack, rcv, 0, (struct
             sockaddr*) threadParm->uml_addr, sizeof(*(threadParm->
             uml_addr)));
8     }
```

A.5. DVCSL UPDATE

Voor het bijwerken van de DVCSL NetKit-ng is er een speciale procedure gemaakt. Hiermee kan de NetKit-ng omgeving blijvend worden aangepast. Hiermee kunnen bijvoorbeeld updates worden geïnstalleerd en nieuwe software worden toegevoegd en bestaande software worden verwijderd. Deze wijzingen blijven bewaart en is het startpunt van de DVCSL omgevingen zoals deze hierboven zijn aangegeven.

Belangrijk is hierbij in de gaten te houden is dat alle DVCSL nodes dezelfde uitgangspunt hebben qua NetKit-ng configuratie. Dit kan op twee manieren, door de hoofd DVCSL node bij te werken en hiervan nieuwe kopie of klonen van te maken, of door alle DVCSL nodes op dezelfde manier bij te werken.

A.5.1. DVCSL UPDATE CONFIGURATIE

Om de NetKit-ng omgeving te starten en de wijzingen bewaard moeten blijven, moet de UML-host op een speciale manier worden gestart. Als eerste moet de 'dvcsl-update' configuratie worden geactiveerd. In tegenstelling tot alle andere omgevingen kan deze niet worden gestart middels het '`dvcsl.start`' maar moet deze worden gestart zoals dit in listing A.8 is aangegeven.

Voor de onderstaande commando's en procedures in deze paragraaf is wel enige Linux kennis en kunde vereist. Voor de volledigheid zijn de onderstaande procedures opgenomen in

deze bijlage.

A.5.2. DVCSL BIJWERKEN

Op het moment dat deze update UML-host is gestart kunnen de onderstaande commando's worden uitgevoerd. Voor een aantal van deze commando's is er een 'echte' Internet verbinding nodig, hiervoor wordt er gebruik gemaakt van het TAP-netwerk.

Om de DNS service van het 'echte' Internet te kunnen bereiken moet deze bekent gemaakt wordt binnen de UML-host door deze in '/etc/resolv.conf'/ te zetten. Door dit bestand aan te passen met de juiste DNS server zoals aangegeven in listing A.21 kunnen de systemen op het 'echte' Internet worden opgezocht. De standaard netwerk routing wordt via het TAP-netwerk verstuurd en daarmee is het 'echte' Internet te bereiken, mist de DVCSL node waarop deze 'dvcsl-update' configuratie is gestart ook een Internet koppeling nodig.

Listing A.21: DVCSL bijwerken resolv.conf

```
root@update:~# cat /etc/resolv.conf
nameserver 192.168.250.1
```

Listing A.22: DVCSL bijwerken resolv.conf

```
[uml@update ~]$ cat dvcsl/vcsl101/lab.start
vstart -q -W --eth0 LAN --eth1 tap,192.168.250.1,192.168.250.2 -M 256 update
```

Listing A.23: DVCSL bijwerken start LAB

```
[uml@update ~]$ ./dvcsl/vcsl101/lab.start
```

Het installeren van updates in de NetKit-ng omgeving kan door middel van het 'aptitude' commando. Door het installeren van updates worden de geïnstalleerde netwerkservices ook weer automatisch gestart. Dit is niet de bedoeling en daarom moeten deze weer worden gedeactiveerd middels het 'update-rc.d' commando. Deze gehele procedure staat weergegeven in listing A.24.

Listing A.24: DVCSL bijwerken aptitude

```
root@update:~# aptitude update
...
root@update:~# aptitude safe-upgrade
...
Wilt u verder gaan? [Y/n/?] Y
...
root@update:~# update-rc.d isc-dhcp-server remove
root@update:~# update-rc.d bind9 remove
root@update:~# update-rc.d mysql remove
root@update:~# update-rc.d apache2 remove
root@update:~# update-rc.d dnsmasq remove
```

Voor dit onderzoek is er in de standaard NetKit-ng een aantal extra software onderdelen geïnstalleerd. Het installeren van nieuwe software in de NetKit-ng omgeving kan door middel van het 'apt-get' commando. Voor de DHCP netwerkservice is de 'isc-dhcp-server' software geïnstalleerd. In listing A.25 is deze procedure beschreven.

De volgende softwarecomponenten zijn geïnstalleerd in de NetKit-ng omgeving om de botnetsimulaties mogelijk te maken:

- isc-dhcp-server
  De DHCP netwerkservice software.
- mysql-server
  De MySQL database server software.
- php5, php5-mysql, libapache2-mod-php5
  De PHP software om PHP webpagina's weer te geven in de Apache webserver.

Listing A.25: DVCSL bijwerken DHCP installatie

```
root@update:~# apt-get install isc-dhcp-server
```

Daarnaast is er in de NetKit-ng omgeving nog een geavanceerde analyse hulpmiddel geïnstalleerd, de 'atoptool'. Hiervoor moet er een extra software bron worden toegevoegd in '/etc/apt/sources.list'. Door het toevoegen van de laatste regel zoals deze in listing A.26 is weergegeven, kan de 'atoptool' worden geïnstalleerd. In listing A.27 is deze procedure beschreven.

Listing A.26: DVCSL bijwerken ATOP bron

```
root@update:~# cat /etc/apt/sources.list

deb http://security.debian.org/ wheezy/updates main contrib non-free
deb http://ftp.fr.debian.org/debian wheezy main contrib non-free
deb http://ftp.de.debian.org/debian wheezy main
```

Listing A.27: DVCSL bijwerken ATOP installatie

```
root@update:~# apt-get install atop
```

B

BOTNET CONFIGURATIE

In deze bijlage zijn alle technische details beschreven van de gebruikte botnets voor dit onderzoek. Als eerste worden de geanalyseerde synthetische botnets beschreven. Als laatste worden de technische details van de uitgevoerde botnetsimulaties beschreven.

B.1. DE SYNTHETISCHE BOTNETS

Voor dit onderzoek zijn er twee Open Source synthetische botnets geanalyseerd, zie paragraaf 5.2 voor de uitgevoerde analyse [Cro15; Bég15].

Voor de technische analyse is de 'dvcsl-analyse' 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) omgeving gebruikt. Deze DVCSL omgeving kan worden geactiveerd zoals in paragraaf A.1.2 is beschreven. Hierin is geen standaard NetKit-ng lab geconfigureerd en daardoor kunnen de DVCSL commando's van tabel A.2 DVCSL commando's tot 'Stap 3' alleen worden uitgevoerd.

Na dat deze omgeving is geactiveerd is de structuur van listing B.1 beschikbaar onder ' /dvcsl'. Hierin zijn de twee synthetische botnets opgenomen. De BYOB botnet in de directory 'BYOB-botnet' en de HybridBot botnet in de directory 'HybridBot-botnet'. Daarnaast zijn er een aantal start en één stop script opgenomen in de 'vcsl101' DVCSL node directory.

Listing B.1: De dvcsl-analyse boom

```
/home/uml/dvcsl/
├──BYOB-botnet
│   ├──CandC
│   └──JAVA_workspace
├──HybridBot-botnet
│   └──web
├──html
│   ├──index.html
│   ├──t75317.css
│   └──t75317.png
└──vcsl101
    ├──analyse
    ├──lab-FrankenBot.start
    ├──lab-HybridBot.start
    ├──lab-minimal.start
    └──lab.stop
```

Middels het '`lab-minimal.start`' commando, wordt er een UML-host gestart met een werkgeheugen van 54MB. Dit is het minimum om een UML-host te kunnen starten. Middels het '`lab.stop`' commando kan deze weer worden gestopt.

BYOB BOTNET

Middels het 'lab-FrankenBot.start' commando, wordt er een UML-host gestart met een werkgeheugen van 64MB. Dit is het minimum om een UML-host te kunnen starten waarin de BYOB bot is gestart.

HYBRIDBOT BOTNET

Middels het 'lab-HybridBot.start' commando, wordt er een UML-host gestart met een werkgeheugen van 55MB. Dit is het minimum om een UML-host te kunnen starten waarin de HybridBot bot is gestart.

B.2. UITGEVOERDE BOTNETSIMULATIES

De uitgevoerde botnetsimulaties zijn uitgevoerd met de HybridBot botnet. Hieronder worden eerst de technische details van de HybridBot botnet uitgewerkt. Als laatste wordt nog stil gestaan bij de gevonden problemen in 'install.php'.

B.2.1. HYBRIDBOT BOTNET

Voor het C&C Systeem voor de HybridBot botnet is geïnstalleerd in de '/var/www/' directory van de router UML-host. In listing B.2 is de inhoud hiervan weergegeven.

Listing B.2: Hybrid C&C boom

```
./var/www
├──GeoIP
│   ├──GeoIP.dat
│   └──geoip.inc
├──README.txt
├──bot
│   └──test.txt
├──dict
├──getcmd.php
├──images
│   ├──bg.png
│   └──world.png
├──index.html
├──index.php
├──install.php
└──screenshots
    ├──hybrid_v1.png
    ├──hybrid_v1a.png
    ├──hybrid_v1b.png
    └──hybrid_v1c.png
```

Er zijn twee belangrijke webpagina's voor de botmaster, de 'install.php' en de 'index.php'. Met de eerste webpagina kan de HybridBot 'Commando en Controle' (C&C) omgeving worden geconfigureerd. Als dat gedaan is kan de botmaster de HybridBot botnet beheren middels de tweede webpagina.

De botmaster kan de HybridBot C&C Systeem benaderen via een standaard Internet browser. Deze kan worden gestart door in het 'menu' van de aangemelde 'User Mode Linux' (UML)-gebruiker te kiezen voor 'Internet -> Firefox'. Standaard wordt hierin de webpagina geopend die bij de geactiveerde DVCSL configuratie hoort. Voor de DVCSL botnet configuraties zijn hierin deze beide C&C webpagina's opgenomen.

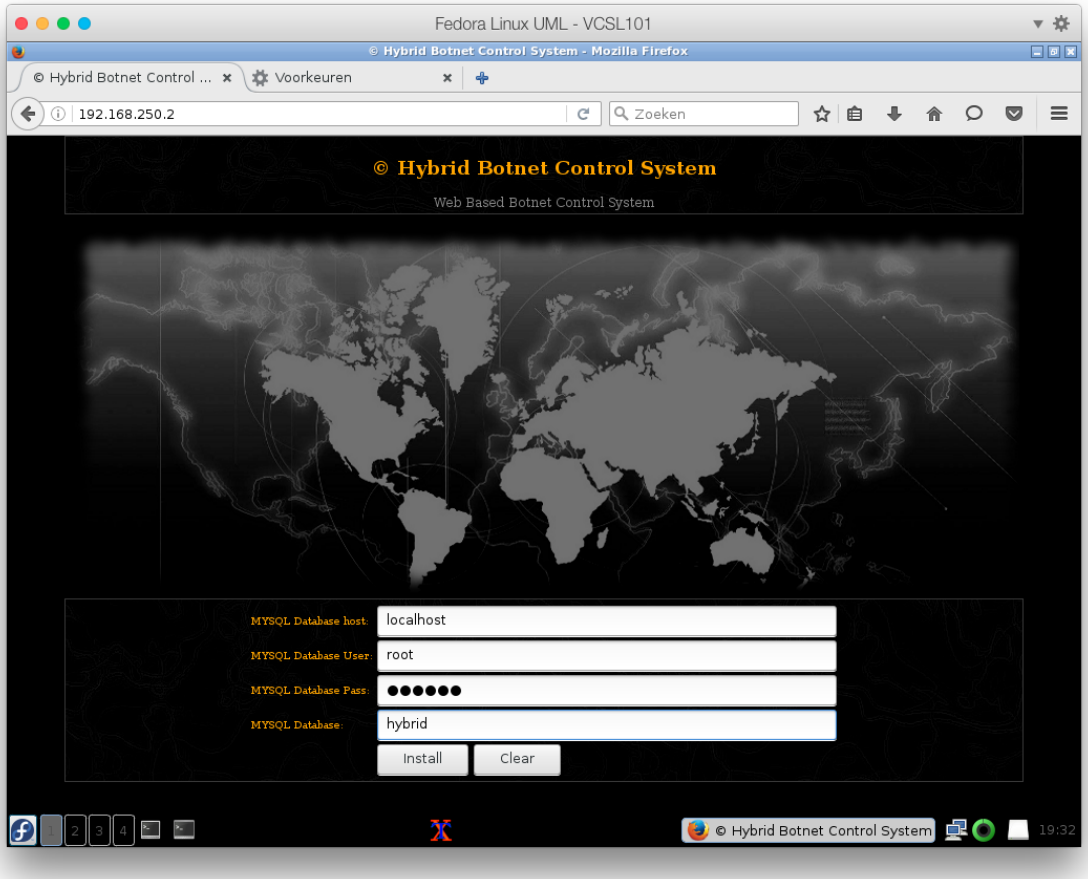
HYBRIDBOT CONFIGURATIES

Voor dit deel van het onderzoek zijn er twee DVCSL configuraties gebruikt. Door de 'dvcsl-botnet-php' DVCSL configuratie te actieveren wordt de basis HybridBot botnet configuratie gebruikt. Hierin moet de HybridBot C&C nog worden geconfigureerd.

Door de 'dvcsl-botnet-test' DVCSL configuratie te actieveren wordt een volledig geïnstalleerde HybridBot C&C gebruikt. Ook worden de HybridBot bots hierin automatisch gestart.

HYBRIDBOT INSTALLATIE

Als de 'dvcsl-botnet-php' DVCSL configuratie is geactiveerd, moet de HybridBot C&C nog worden geconfigureerd. Dit kan worden gedaan door botmaster naar de 'Command & Control - Installatie' te gaan. Hierna verschijnt de C&C installatie webpagina zoals deze is weergegeven in figuur B.1.



Figuur B.1: HybridBot Installatie C&C database

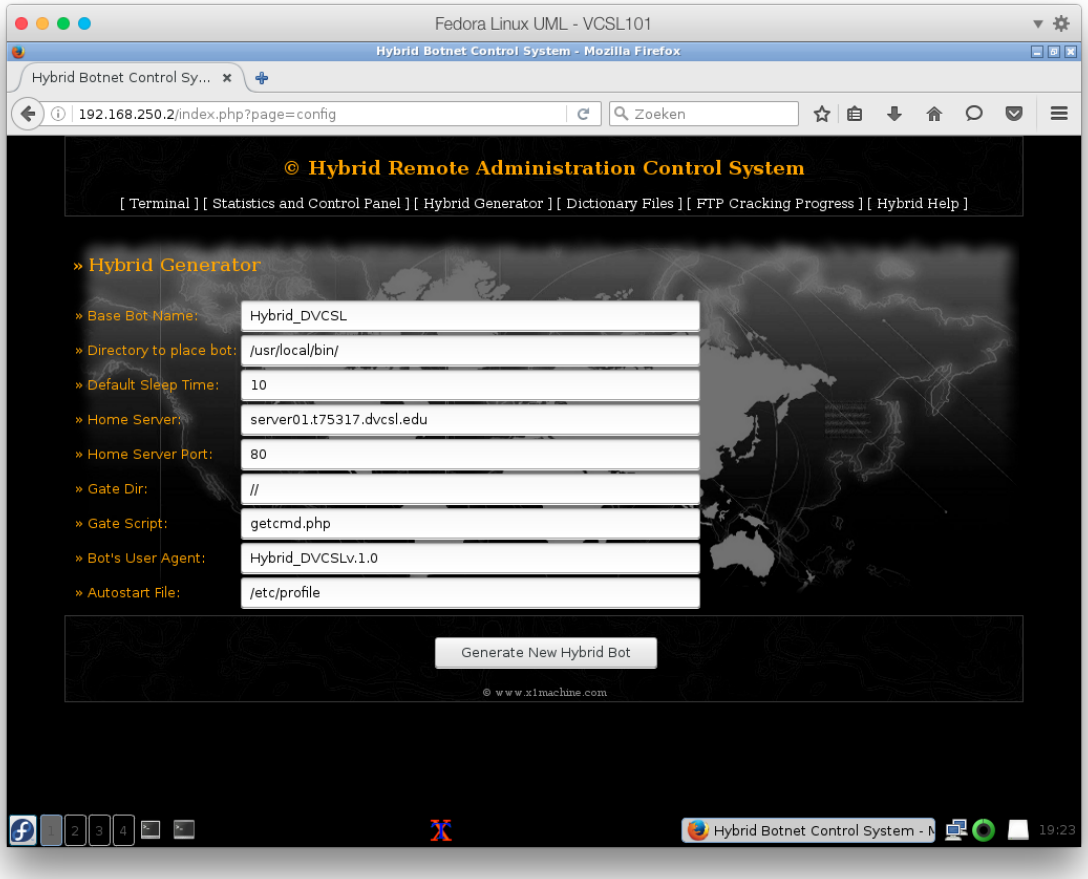
De gebruikte instellingen zijn weergegeven in tabel B.1 HybridBot installatie en zijn voorzien van de bijbehorende opmerkingen. Door hierna op 'Install' te klikken worden de MySQL database voor het C&C Systeem geconfigureerd. In eerste instantie ging deze stap niet goed en kon de HybridBot niet worden beheerd. Dit probleem is opgelost door een aanpassing in de installatie script. Deze aanpassing wordt verderop beschreven.

Tabel B.1: HybridBot installatie

Veld	Waarde	Opmerking
MYSQL Database host	localhost	De MySQL database server is geïnstalleerd op hetzelfde systeem als de Apache webserver
MYSQL Database User	root	Hiervoor is de standaard Linux beheer account gebruikt
MYSQL Database Pass	123456	Het bijbehorende wachtwoord, zoals deze is gebruikt bij het aanmaken van de MySQL database
MYSQL Database	hydrid	Deze lege database is hiervoor aangemaakt

HYBRIDBOT BOT

Voor dit onderzoek is er een HybridBot bot gemaakt via de HybridBot C&C console. Hiervoor moet de HybridBot C&C zijn geïnstalleerd. Daarna kan het HybridBot C&C console worden gestart door botmaster naar de 'Command & Control - Home' te gaan. Door hier voor 'Hybrid Generator' verschijnt de C&C webpagina waarmee een HybridBot bot gemaakt kan worden, zoals deze is afgebeeld in figuur B.2.



Figuur B.2: HybridBot generator

HYBRIDBOT OPDRACHTEN

In figuur B.3 zijn de mogelijke aanvalsmethoden van de HybridBot botnet weergegeven. Hierbij is voor alle HybridBot opdrachten een kleine beschrijving gegeven. Een aantal van deze opdrachten zijn aanvalsopdrachten, deze zijn in Hoofdstuk 5 'Botnetsimulatie' en Hoofdstuk 6 'DVCSL botnetsimulatie casus' al aan bod gekomen.

B.2.2. HYBRIDBOT COMMANDO EN CONTROLE CONSOLE

De HybridBot C&C console is beveilig met een gebruikersnaam en wachtwoord, deze staan middels een 'md5' codering in de 'index.php'. Deze is voor dit onderzoek niet aangepast, waardoor is de gebruikersnaam van de botmaster nog steeds '1' is, net als het wachtwoord.

B.2.3. PROBLEMEN HYBRIDBOT

Door het handmatig draaien van de installatie scripts was het al snel duidelijk dat er een probleem was met het aanmaken van de tabellen in de MySQL database. De oorzaak hiervan was dat de gebruikte nieuwere versie van MySQL de opgegeven database type niet meer op de in de scripts aangegeven manier ondersteunde. De gebruikte opties in de originele SQL statement voor het aanmaken van de tabellen is weergegeven in listing B.3.

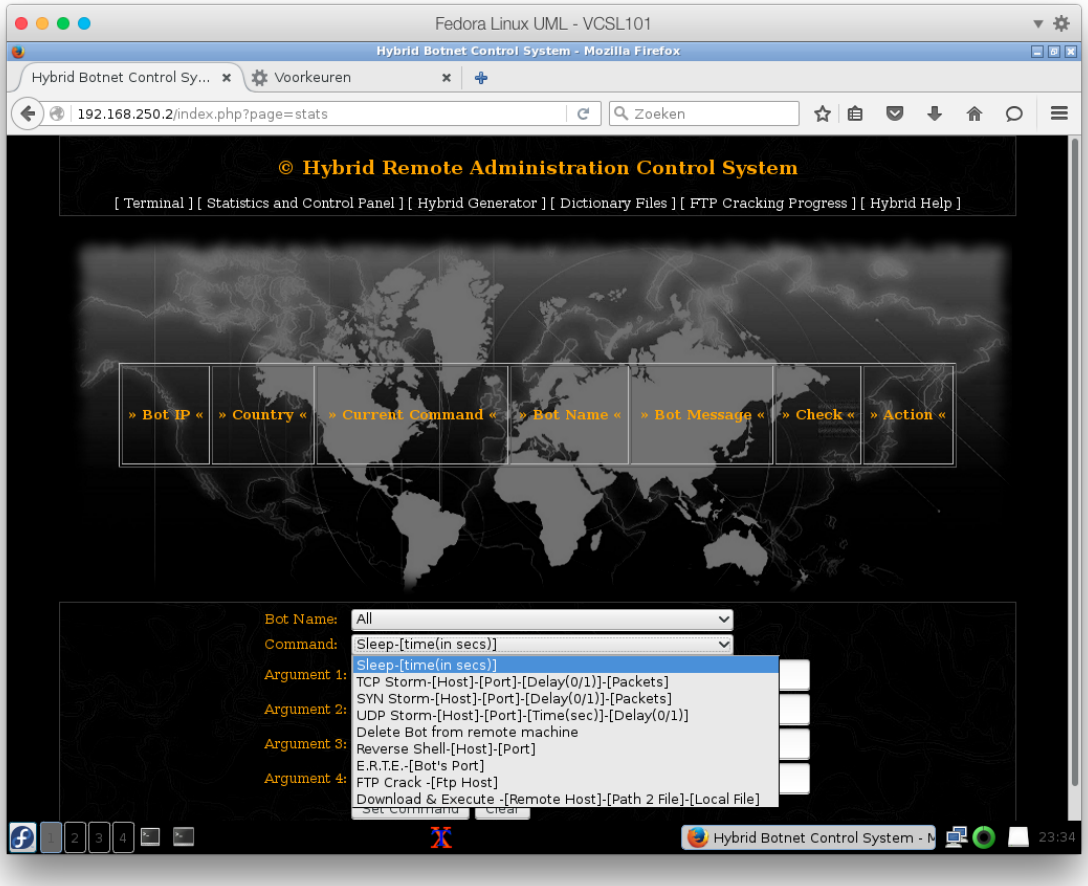
Deze opties zijn uit alle SQL statements gehaald, waarna de installatie van de C&C interface ook weer via een standaard Internet browser kan worden gedaan. De aanpassing in de SQL statements is weergegeven in listing B.4. Het verschil zit in regel 6, waarbij 'TYPE=MyISAM' uit al deze SQL statements is verwijderd.

Listing B.3: Install.php zonder aanpassing

```
1     $install = "CREATE TABLE `$table` (".
2            "`id` int(10) NOT NULL auto_increment,".
3            ...
4            " PRIMARY KEY (`id`),".
5            " UNIQUE KEY `id`(`id`)".
6            ") TYPE=MyISAM COMMENT='' AUTO_INCREMENT=1 ;";
```

Listing B.4: Install.php met aanpassing

```
1     $install = "CREATE TABLE `$table` (".
2            "`id` int(10) NOT NULL auto_increment,".
3            ...
4            " PRIMARY KEY (`id`),".
5            " UNIQUE KEY `id`(`id`)".
6            ") COMMENT='' AUTO_INCREMENT=1 ;";
```



- Sleep-[time(in sec)]
  Laat de bots slapen voor een opgegeven tijd in seconden
- TCP Storm-[Host]-[Port]-[Delay(0/1)]-[Packets]
  De bots starten een 'Transmission Control Protocol' (TCP) aanval naar de opgegeven system
- SYN Storm-[Host]-[Port]-[Delay(0/1)]-[Packets]
  De bots starten een 'Synchronous TCP flood' (SYN) aanval naar de opgegeven system
- UDP Storm-[Host]-[Port]-[Time(sec)]-[Delay(0/1)]
  De bots starten een 'User Datagram Protocol' (UDP) aanval naar de opgegeven system
- Delete Bot from remote machine
  Verwijderen van de bots
- Reverse Shell-[Host]-[Port]
  De bots starten een commando shell op de opgegeven poort
- E.R.T.E.-[Bot's Port]
  De bots starten een E.R.T.E. proces op de opgegeven poort
- FTP Crack -[Ftp Host]
  De bots starten een 'File Transfer Protocol' (FTP) aanval naar de opgegeven systeem
- Download & Execute -[Remote Host]-[Path 2 File]-[Local File]
  De bots starten een download en voeren het opgegeven bestand uit

Figuur B.3: HybridBot opdrachten

C

BOTNETSIMULATIE CASUS

In deze bijlage zijn alle technische details beschreven voor het uitvoeren van de botnetsimulatie casus van Hoofdstuk 6 'DVCSL botnetsimulatie casus'.

C.1. BOTNETSIMULATIE CASUS CONFIGURATIE

In paragraaf 6.2 is de beschrijving van de botnetsimulatie casus uitgewerkt. In deze paragraaf wordt aangegeven hoe deze 'Gedistribueerd Virtuele Computer Security Lab' (DVCSL) configuratie, geactiveerd, gestart en gestopt kan worden.

Als eerste moet de DVCSL configuratie worden geactiveerd en daarna kan de DVCSL configuratie worden gestart. Na het starten van de DVCSL configuratie kunnen de botnetsimulaties worden uitgevoerd. Na het uitvoeren van de botnetsimulaties kan de DVCSL configuratie worden gestopt.

C.1.1. ACTIVEREN CONFIGURATIE

Voor dit deel van het onderzoek is er één DVCSL configuratie gebruikt. Door de 'dvcsl-botnet-casus' DVCSL configuratie te activeren zoals in Bijlage A 'DVCSL Configuratie', paragraaf A.1.2 staat beschreven kunnen de onderstaande acties worden uitgevoerd.

Het resultaat van dit commando staat weergegeven in listing C.1.

Listing C.1: Activeren van de dvcsl-botnet-casus

```
[uml@vcsl101 ~]$ dvcsl.make
Reading /home/uml/dvcsl/../T75317/dvcsl.conf...
Activeer DVCSL omgeving: dvcsl-botnet-example...
```

C.1.2. STARTEN VAN DE OMGEVING

Als op alle DVCSL nodes de 'dvcsl-botnet-casus' DVCSL configuratie is geactiveerd, kan de DVCSL configuratie worden gestart. Als eerste moet de DVCSL-node-1 worden gestart, anders kunnen de HybridBot bots niet bij de centrale C&C Systeem komen.

Nadat DVCSL-node-1 is gestart kunnen de andere DVCSL nodes worden gestart. In deze casus zijn dat DVCSL-node-2 en DVCSL-node-3. Er zit geen verschil in het starten en stoppen van DVCSL-node-2 en DVCSL-node-3, daarom zijn in de onderstaande voorbeelden alleen die vanDVCSL-node-2 beschreven.

STARTEN VAN DVCSL-NODE-1

Als eerste moet de DVCSL configuratie op DVCSL-node-1 worden gestart. Het starten van de DVCSL omgeving staat beschreven in paragraaf A.2.3. Het resultaat van dit commando staat weergegeven in listing C.2 en grafische weergegeven in figuur C.1.

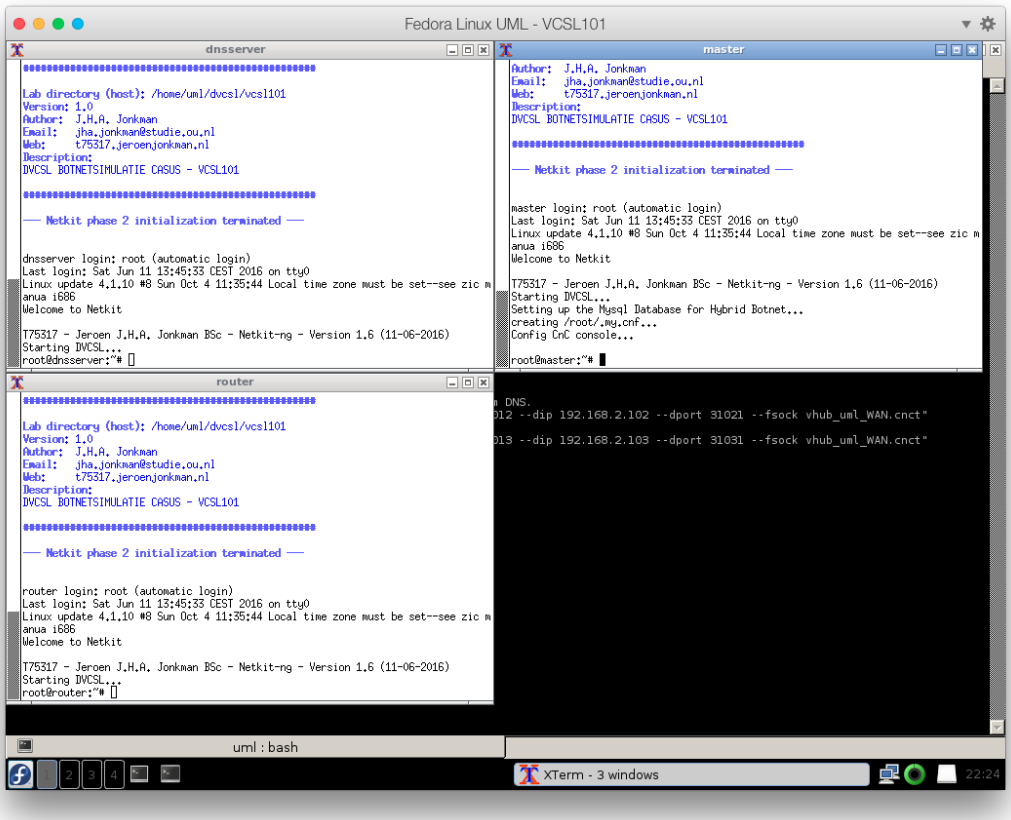
Listing C.2: Starten van VCSL101

```
[uml@vcsl101 ~]$ dvcsl.start
vcsl101.local.dvcsl.edu has ip-address 192.168.2.101.
Warning: fast mode is ignored when using parallel startup.

======================== Starting lab ============================
Lab directory: /home/uml/dvcsl/vcsl101
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL BOTNETSIMULATIE CASUS - VCSL101
==================================================================
You chose to use parallel startup.
Starting "dnsserver"...
Starting "router"...
Starting "master"...

The lab has been started.
==================================================================

Starting Remote DVCSL plug from vcsl101 to vcsl102
DVCSL_DOMAIN is not set in the LAB config file, using system domain from DNS.
Starting plug with "DVCSL_PLUG_PARAMETER=--sip 192.168.2.101 --sport 31012 --dip 192.168.2.102 --dport 31021 --
      fsock vhub_uml_WAN.cnct"
Starting Remote DVCSL plug from vcsl101 to vcsl103
Starting plug with "DVCSL_PLUG_PARAMETER=--sip 192.168.2.101 --sport 31013 --dip 192.168.2.103 --dport 31031 --
      fsock vhub_uml_WAN.cnct"
```



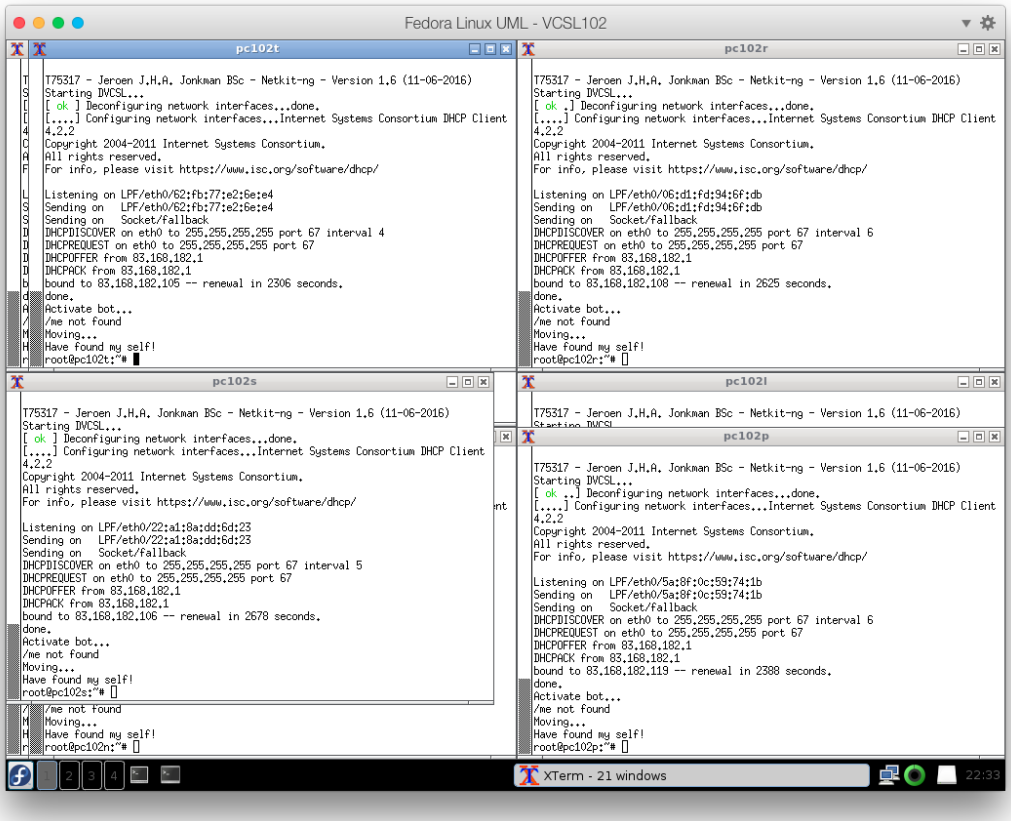
Figuur C.1: DVCSL botnet start vcsl101

Na het starten van de 'dvcsl-botnet-casus' DVCSL configuratie op DVCSL-node-1 is de HybridBot 'Commando en Controle' (C&C) beschikbaar voor de botmaster.

STARTEN VAN DVCSL-NODE-2 EN DVCSL-NODE-3

Als DVCSL-node-1 volledig is gestart, dan kunnen de andere DVCSL nodes worden gestart.

Hiervoor moet de 'dvcsl-botnet-casus' DVCSL configuratie eerst geactiveerd worden op de DVCSL node. Daarna kan de DVCSL node worden gestart. Dit gaat op dezelfde wijze als voor DVCSL-node-1. Het resultaat van dit commando op DVCSL-node-2 staat weergegeven in listing C.3 en grafische weergegeven in figuur C.2.



Figuur C.2: DVCSL botnet start vcsl102

Listing C.3: Starten van VCSL102

```
[uml@vcsl102 ~]$ dvcsl.start
vcsl102.local.dvcsl.edu has ip-address 192.168.2.102.
Warning: fast mode is ignored when using parallel startup.

======================== Starting lab ============================
Lab directory: /home/uml/dvcsl/vcsl102
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL BOTNETSIMULATIE CASUS - VCSL102
==================================================================
You chose to use parallel startup.
Starting "router"...
...
Starting "pc102t"...

The lab has been started.
==================================================================

Starting Remote DVCSL plug from vcsl102 to vcsl101
DVCSL_DOMAIN is not set in the LAB config file, using system domain from DNS.
Starting plug with "DVCSL_PLUG_PARAMETER=--sip 192.168.2.102 --sport 31021 --dip 192.168.2.101 --dport 31012 --
      fsock vhub_uml_WAN.cnct"
```

C.1.3. STOPPEN VAN DE OMGEVING

De DVCSL nodes kunnen na de botnetsimulaties weer worden gestopt. Als eerste moeten alle DVCSL configuraties worden gestopt. Vervolgens kunnen dan de DVCSL nodes worden gestopt.

STOPPEN VAN DE DVCSL CONFIGURATIE

Als de DVCSL configuratie is gestart, dan kan deze worden gestopt door op alle DVCSL nodes de DVCSL configuratie te stoppen, zoals in paragraaf A.2.4 is aangegeven. Het resultaat van dit commando staat weergegeven in listing C.4. Het beste is om de DVCSL-node-1 als laatste te stoppen.

Listing C.4: Stoppen van VCSL101

```
[uml@vcsl101 ~]$ dvcsl.stop
vcsl101.local.dvcsl.edu has ip-address 192.168.2.101.
Stopping DVCSL on

======================== Halting lab ============================
Lab directory: /home/uml/dvcsl/vcsl101
Version: 1.0
Author: J.H.A. Jonkman
Email: jha.jonkman@studie.ou.nl
Web: t75317.jeroenjonkman.nl
Description:
DVCSL BOTNETSIMULATIE CASUS - VCSL101
==================================================================
Halting virtual machine "router" (PID 3550) owned by uml [....... ]
Halting virtual machine "dnsserver" (PID 3518) owned by uml [....... ]
Halting virtual machine "master" (PID 3612) owned by uml [..................]
Removing readyfor.test...

Lab has been halted.
==================================================================
```

STOPPEN VAN DE DVCSL NODES

Het stoppen van de DVCSL node kan op vele manieren. Één van de manieren is om het DVCSL node te stoppen en gelijk uit te zetten. Dit kan door op de DVCSL nodes het '`poweroff`' commando te geven. Hierdoor worden de virtuele machines ook gelijk gestopt.

C.2. DVCSL BOTNETSIMULATIE

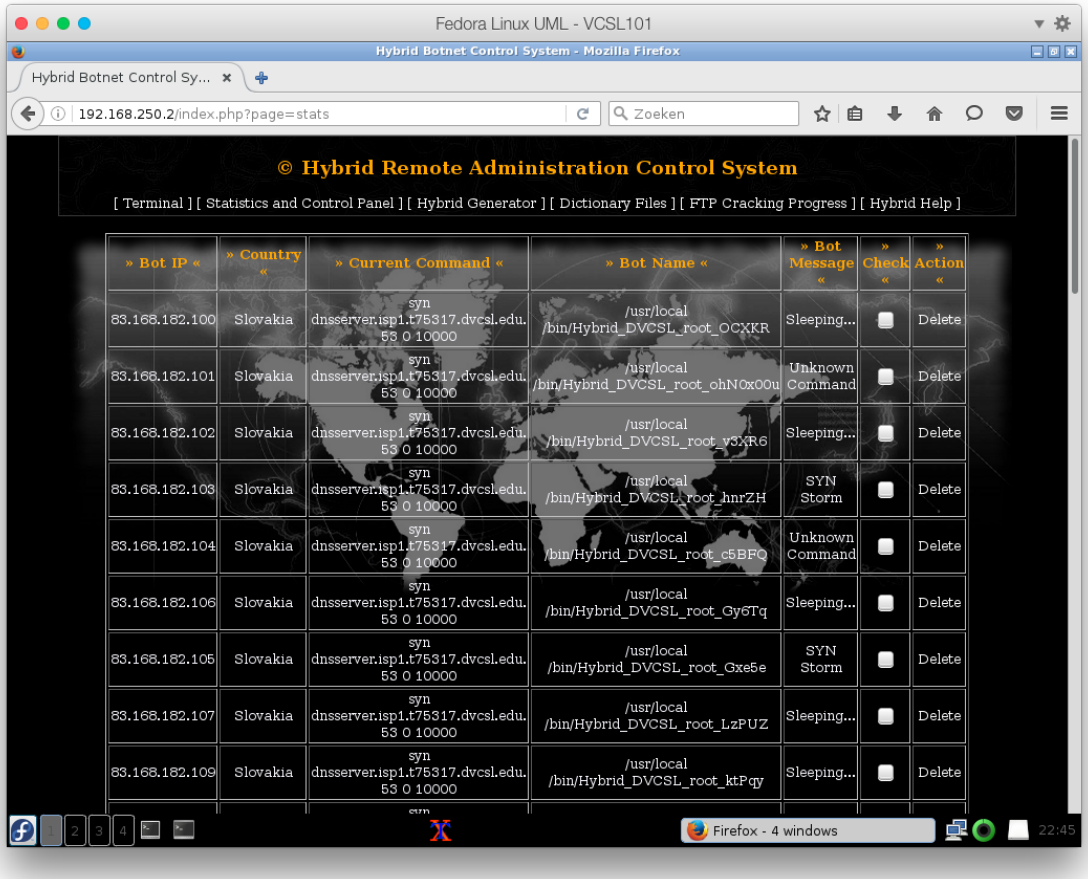
In paragraaf 6.3 is de beschrijving van de uitgevoerde botnetsimulatie casus weergegeven. In deze paragraaf wordt aangegeven hoe deze botnetsimulatie kan worden gestart.

Als basis van deze DVCSL configuratie is gebruik gemaakt van de 'dvcsl-botnet-test' configuratie, waarbij de HybridBot C&C console volledig is geïnstalleerd en geconfigureerd zoals beschreven in paragraaf B.2.

C.2.1. HYBRIDBOT COMMANDO EN CONTROLE CONSOLE

Deze botnetsimulatie casus maakt gebruik van 'echte' IP-adressen waardoor er meer gegevens kunnen worden getoond. Van de actieve bots kan bijvoorbeeld het land worden aangegeven, dit wordt gedaan op basis van het IP-adres. Voor het '83.168.182.0/24' netwerksegment is dat 'Slowakia'. Voor het '62.251.32.0/24' netwerksegment is dat 'Netherlands'.

In figuur C.3 is de C&C console weergegeven tijdens het uitvoeren van de botnetsimulatie waarin deze informatie zichtbaar is.



Figuur C.3: DVCSL botnet C&C Console

C.2.2. HYBRIDBOT BOTNETSIMULATIE

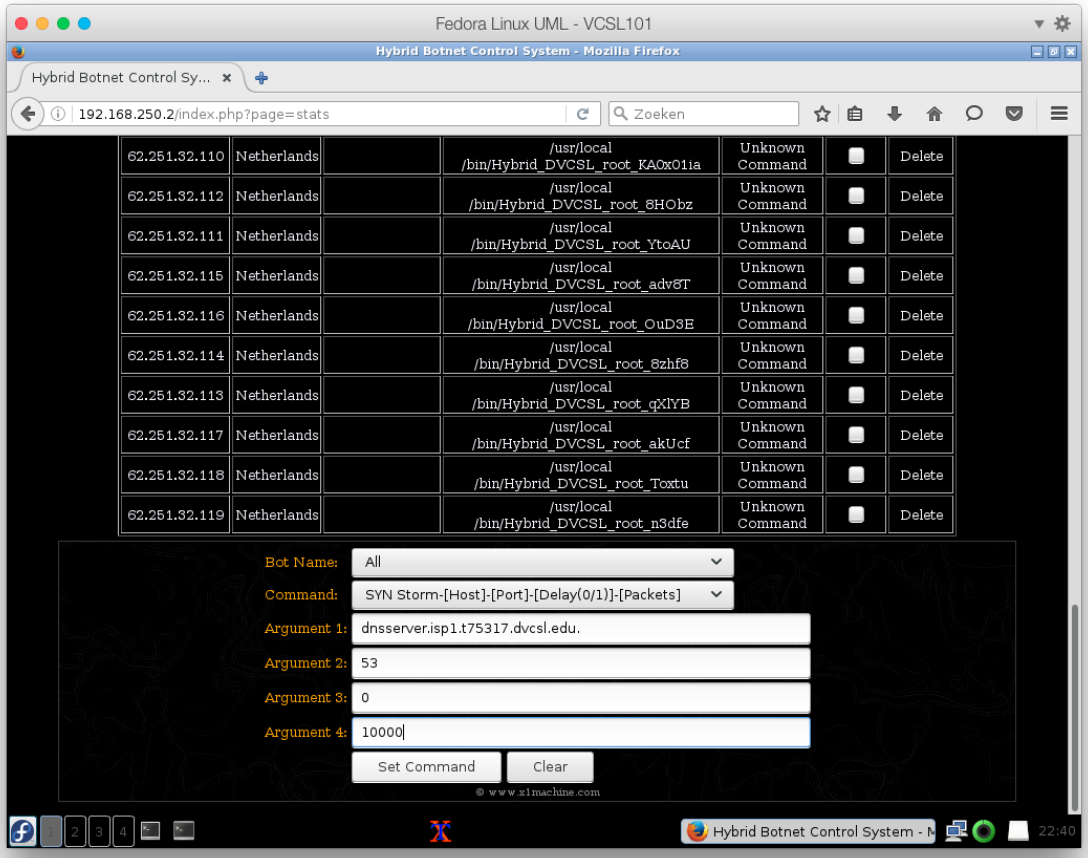
In figuur C.4 zijn de gegevens zichtbaar zoals deze in paragraaf 6.3 zijn aangegeven. Door op 'Set Command' te klikken wordt deze actie klaar gezet. De HybridBot bots gaan dan deze actie uitvoeren door op gezette tijden te controleren of er nieuwe acties klaar staan.

Omdat een aantal HybridBot bots direct met de 'Domain Name System' (DNS) service aanval beginnen, kunnen de andere bots niet meer het IP-adres van de HybridBot C&C Sys-

teem opvragen en starten daardoor later. Dit is ook zichtbaar in figuur C.3 waarbij een aantal bots al klaar zijn en wachten op de volgende opdracht ('Sleeping...'). Een aantal bots moeten nog beginnen aan deze opdracht ('Unkown Command') en een aantal bots zijn de aanval aan het uitvoeren ('SYN Storm').

Deze botnetsimulatie duurt maar enkele minuten, maar door bij 'Argument 4' bijvoorbeeld '100000' in te vullen duurt deze aanval een behoorlijke tijd langer. Hoeveel langer is moeilijk te zeggen omdat dit afhangt van de beschikbare systeem resources.

Na het uitvoeren van deze botnetsimulatie kan de DVCSL configuratie worden gestopt en eventueel weer worden herhaald door deze opnieuw te activeren en te starten.



Figuur C.4: DVCSL botnet C&C Console voorbeeld