Adaptiviteit en Botnets

D.Weeteling
Universiteit Utrecht
Cognitieve Kunstmatige Intelligentie
begeleiding: Lennart Herlaar & Gerard Vreeswijk

February 21, 2012

De afgelopen decennia is het internet een integraal onderdeel van de samenleving geworden. Een voor de A.I. interessante exponent van het internet is het fenomeen "botnets". Een botnet is een netwerk van agenten ("bots") dat als een enkel organisme opereert. De term "botnet" heeft een negatieve connotatie gekregen doordat deze netwerken veelal criminele doelen dienen (bijvoorbeeld spam versturen). Dit hoeft echter niet zo te zijn; er bestaan ook botnets die voor onder andere onderzoek worden ingezet. Oorspronkelijk werden botnets centraal aangestuurd. De beslissingen omtrent verdere acties werden genomen door een menselijke agent, de botherder. Echter, naarmate de tijd vordert laat de techniek toe dat het netwerk, in het verlengde van de individuele bots, steeds meer beslissingen zelfstandig neemt. Dit geldt o.m. voor beslissingen die betrekking hebben op defensie. Het netwerk wil zijn voortbestaan veilig stellen door zoveel mogelijk dreiging onschadelijk te maken. Welke technieken gebruikt worden door het netwerk/de individuele bots om deze doelen te bewerkstelligen is hetgeen er in deze literatuurstudie wordt onderzocht.

In de A.I. wordt onderzoek gedaan naar de adaptiviteit van kunstmatige systemen. Een botnet is een goed voorbeeld van een systeem dat vanuit de zwakke A.I. interessant is om te onderzoeken. De sterke A.I. kant zal verder geen rol spelen in deze paper; deze filosofische invalshoek is in dit paper ondergeschikt aan de zwakke A.I. invalshoek. Welke technieken worden er door de bots gebruikt om schijnbaar intelligent gedrag te verwezenlijken? Op welke wijze werken de individuele agenten samen om de doelen van het collectief, het botnet, te verwezenlijken? Dit zijn de zaken die worden besproken in deze paper.

Na deze inleiding wordt er in het eerste hoofdstuk gekeken naar de geschiedenis. Deze begint bij Internet Relay Chat (IRC) en hoe de eerste bots hiervoor geschreven werden. Gedurende het laatste decennium van de vorige eeuw zien we de botnets een steeds hoger ontwikkeld maar tevens onvriendelijker karakter krijgen, een ontwikkeling die ook in dit hoofdstuk uiteengezet wordt. Vervolgens wordt in het tweede hoofdstuk besproken hoe de moderne botnets functioneren en welke technieken er voor deze netwerken ontstaan zijn. Dit wordt toegelicht door een casus (Storm) te bestuderen die kenmerkend is voor de 21e eeuwse botnets. In het derde hoofdstuk wordt gekeken naar de technieken die bots gebruiken om te kunnen overleven in hun vijandelijke directe omgeving, het Operating System (OS). Hoe dit collectief aan bots zich aanpast aan zijn directe omgeving, het internet, is onderwerp van het vierde hoofdstuk.

1 Geschiedenis van de botnets

In dit hoofdstuk volgt een bespreking van de evolutie van de botnets. Dit begint eind jaren 80 van de vorige eeuw, met de komst van twee nieuwe verschijnselen: het opduiken van de eerste echte computer worm (Morris worm, 1988) en de uitvinding van het IRC protocol in 1989. Een worm bestaat uit code die zichzelf via een netwerk kan propageren, door (bestanden van) andere hosts op het netwerk te infecteren. De geïnfecteerde hosts bevat de volledige wormcode nodig voor replicatie, en zodoende is deze nieuwe host zelf ook weer in staat andere hosts op door hem bereikbare netwerken te infecteren. Een worm hecht zich niet per definitie aan andere uitvoerbare code, en hoeft geen payload [27] te hebben om toch schade aan te richten: de pogingen tot duplicatie via het netwerk leiden altijd tot het gebruik van extra bandbreedte en mogelijk zorgt dit ervoor dat netwerken geen normaal verkeer meer kunnen afhandelen. De invloed van wormtechnologie binnen botnets, relevant voor het tot stand kunnen komen van de gigantische moderne botnets, laat echter nog even op zich wachten; het prilste begin van de botnets kenmerkt zich door onschuldige scripts ter facilitatie van het beheer van IRC kanalen en het vermaak van de bezoekers [14].

1.1 Internet Relay Chat

Het open-source IRC protocol [17,18] is bedacht in 1989, en stelt de gebruikers in staat met elkaar te chatten volgens het client-server model: de individuele clients dienen te verbinden met de IRC server, welke de clients in de gelegenheid stelt met de andere clients data uit te wisselen. Een IRC server kan over meerdere kanalen beschikken die normaliter fungeren als chatrooms met een bepaald onderwerp. Bij het maken van een nieuw kanaal op de IRC server wordt de persoon die als eerste het kanaal betreedt gezien als channel operator. Deze persoon wordt daarmee de channel operator en verkrijgt hiermee hogere rechten binnen het kanaal dan de gewone bezoekers. Naarmate de populariteit van IRC groeide nam het aantal servers per IRC netwerk toe, wat de kwetsbaarheid van deze netwerken vergrootte: een enkele kwaadwillende of slecht geconfigureerde server op het IRC netwerk kon het gehele netwerk (tijdelijk) onbereikbaar maken. Dit is vanwege de opbouw van een IRC netwerk [28]. Een ander verschijnsel dat kwam met de toename van het aantal IRC gebruikers was het automatiseren van eenvoudige taken.

1.2 De opkomst van de IRC bots

Een IRC kanaal wat eenmaal aangemaakt was moest behouden kunnen worden en oorspronkelijk moest daarvoor de channel operator in het kanaal aanwezig blijven. Een eenvoudige oplossing is dan het scripten van een client die als simpele channel operator op kan treden, om zodoende het IRC kanaal open te houden en dus ervoor te zorgen dat een ander geen operator status op jouw IRC kanaal kon verkrijgen. Deze client kan dan gedraaid worden op een shellaccount [26]. De bots werden echter al snel geavanceerder: bots konden bijvoorbeeld de bezoekers vermaken met een quiz [14]. Een overzicht van enkele verschillende types IRC bots:

- *AnnoyBot:* Sluit zich aan bij een channel en dupliceert zichzelf om vervolgens het kanaal te bestoken met onzin-text. Op het moment dat de channel operator besluit de AnnoyBot er uit te gooien, blijft deze hardnekkig proberen om zich weer bij het kanaal aan te melden.
- *SpyBot:* Houdt bij wie wat zegt in een kanaal als zijn opdrachtgever er zelf niet is. In moderne IRC clients is dit soort functionaliteit standaard inbegrepen.
- *GuardBot:* Een bot die zijn best doet om zijn baas zoveel mogelijk te behagen als deze een IRC kanaal bezoekt; de bot zal pogingen tot het bannen van zijn baas proberen te verhinderen.
- *CollideBot:* Verandert zijn eigen nickname in die van een andere gebruiker van het kanaal; als de oorspronkelijke gebruiker om wat voor reden dan ook offline raakt, zal de bot de identiteit (inclusief privileges) van de oorspronkelijke naambezitter binnen het kanaal overnemen.

Voorbeelden van de eerste generatie IRC bots zijn GM (1989, speelde een text-gebaseerd spelletje met de client) en Eggdrop (1993); deze laatste bot introduceerde het begrip 'botnet', zij het dat het met dit begrip net iets anders bedoelde dan wat wij in er 2012 onder verstaan [14].

1.3 Eggdrop

In 1993 werd de IRC bot Eggdrop geschreven. De bot diende ter beheer/bescherming van een IRC kanaal en was geschreven in C. Door middel van scripts in de taal TCL kon de functionaliteit door de eindgebruiker worden uitgebreid. Het volgende TCL script [19] kan bijvoorbeeld nieuwe gebruikers die het IRC kanaal binnenkomen groeten:

```
r01  set ngver "v0.6.4"
r02  set greetfile "greetfile.txt"
r03  bind join - * ngreet
r04  proc ngreet {nick host handle chan} {
r05      if {$handle == "*"} {
r06          global greetfile
r07          set file [open $greetfile r]
r08          set line 0
r09          while ![eof $file] {
r10              set lines([incr line]) [gets $file]
r11          }
r12          set line [rand $line]
r13          set randomline $lines($line)
r14          close $file
r15          puthelp "PRIVMSG $chan :\[$nick\] $randomline"
r16      }
r17  }
r18  putlog "\002newgreet\002 $ngver by toot, loaded!@"
```

Dit script zet eerst de benodigde globale variabelen, waaronder het path van

een .txt file met op elke regel een groet (regel 1-2). Dit bestand wordt door de functie ngreet (regel 4) gebruikt om een array met groeten te maken, waar er vervolgens random een van uit wordt gekozen (regel 12-13). Deze groet wordt ingebed in een string die met de puthelp functie naar de IRC server gestuurd wordt (regel 15). De server zal het begin van de string, "PRIVMSG" zien als opdracht om de groet $randomline naar de nieuwe gebruiker ($nick) op het kanaal ($chan) door te zenden.

De Eggdrop bot was geschreven voor het toezicht houden op een enkel IRC kanaal en voor meerdere IRC kanalen waren dus meerdere instanties van Eggdrop vereist. Eggdrop voorzag in de mogelijkheid om o.a. statistieken tussen de instanties van Eggdrop te delen en verhoogde privileges aan andere instanties van Eggdrop toe te bedelen. Deze functionaliteit werd door de Eggdrop software "botnet" genoemd, en daarmee was de term geboren.

1.4 De opkomst van de internet wormen

In de loop van de jaren 90 werden de bots steeds uitgebreider, en werden ze vaak ingezet voor de strijd om de heerschappij van IRC kanalen. Ook een ander relevant fenomeen kwam in deze late jaren 90 "IRC wars" context in de mode: de (Distributed) Denial of Service aanval [20]. Deze aanvallen werden in eerste instantie gebruikt voor het offline halen van IRC servers/netwerken, maar werden later ook tegen individuele IRC gebruikers ingezet, bijvoorbeeld met een smurf aanval [21].

Voor het kunnen bestaan van de moderne botnets is niet alleen een Command & Control (C&C) server nodig, maar ook een manier om zo veel mogelijk bots in het netwerk te krijgen. Eind jaren '90 ziet deze behoefte zich gesteund door de opkomst van de internet wormen. De eerste wormen die voor chaos zorgden (Christmas Tree EXEC (1987), Morris worm (1989)) mochten dan wel uit de hand gelopen studentengrappen zijn, eind jaren '90 begint er al een redelijke markt voor criminele hackers te ontstaan die hun werkterrein zien toenemen naar gelang het grote publiek de e-commerce ontdekt. Voor een groot aantal infecties van computers met malware die zodoende de botherder geld opleveren, is het wenselijk snel veel hosts te infecteren; een geschikte manier hiervoor is een internetworm die bij bezochte hosts heimelijk de botcode installeerd.

De eerste keer dat wormen en IRC bots samen kwamen in een enigzins gedocumenteerd geval is de in Borland Delphi geschreven PrettyPark worm (1999). Deze worm beschikte over een aantal kenmerken die, naast de botnet invloeden, standaard zouden worden voor de toekomstige bots [14]:

- systeem/netwerk informatie verzamelen
- login data verzamelen
- de mogelijkheid om zichzelf te updaten
- files down- en uploaden
- (D)DoS aanvallen uitvoeren
- verkeer omleiden (proxyserver)
- niet proberen nogmaals een besmette host te infecteren

De PrettyPark worm kwam met een eigen IRC client waarmee hij met een C&C server contact maken kon. Deze worm verspreidde zichzelf op twee verschillende wijzen: door een kopie van zichzelf te sturen naar alle emailcontacten, en door te verbinden met publieke IRC servers en de bezoekers daarvan een kopie van zichzelf toe te zenden. Merk op dat deze relatief vroege internetworm voor zijn verspreiding dus geen gebruik maakte van zero-day vulnerabilities [29].

1.5 Het volwassen worden van de wormen

In de late jaren '90 waarin PrettyPark kwam werd meer malware ontwikkeld die relevant is voor het kunnen ontstaan van de moderne botnets. De opkomst van uitgebreide Remote Administration Tools/Trojans (RATs) die gebruikers in staat stellen op afstand een computer heimelijk te besturen is hier een goed voorbeeld van. Voorbeelden van deze vroege RATs zijn NetBus (1998), Back Orifice (1998), en SubSeven (1999). De meeste moderne botnets maken per bot gebruik van vergelijkbare functionaliteit. Met de komst van de 21e eeuw zien we achtereenvolgens de komst van de GT-, SD-, Ago- en SpyBot families in respectievelijk 2000, 2002, 2002 en 2003 (sourcecode veelal te vinden op [22]; betreden op eigen risico). Al deze families maakten gebruik van een C&C structuur via IRC. De bots namen na installatie contact op met de C&C server [24]. Sommige vroege bots (AgoBot) scanden voor kwetsbare poorten om het botnet waarin zij verkeerden te vergroten; vanaf deze tijd is te zien dat worm technologie (reproductie via het netwerk, met of zonder actieve interactie van de gebruiker) en RAT technologie (uitgebreide informatie over de geïnfecteerde hosts) steeds onlosmakelijker met een C&C structuur op basis van IRC (gebundelde controle over alle geïnfecteerde hosts tegelijkertijd) werd. Samen zijn deze ontwikkelingen er debet aan dat in de 2e helft van het eerste decennium van de 21e eeuw er botnets konden beginnen te ontstaan die vele malen groter waren dan de auteur van Eggdrop ooit voor mogelijk had gehouden.

2 Moderne botnets

2.1 introductie

De botnets zijn steeds groter en geavanceerder geworden. Wat betreft de structuur van het netwerk zijn er vele technieken bijgekomen die de moderne botnets robuuster moeten maken. Hoewel het overgrote gedeelte van de moderne botnets nog steeds gebruik maakt van IRC netwerken voor het aansturen van het botnet, zijn er tevens tal van alternatieve C&C structuren bijgekomen zoals bijvoorbeeld Twitter [23], het web [14], of op basis van Peer2Peer [9, 12, 13, 14]. Er wordt vaak gebruik gemaakt van encryptie om ervoor te zorgen dat het verkeer in het netwerk niet of moeilijk te analyseren valt. Ook qua moraal is er veel veranderd: waar botnets in eerste instantie goedaardig waren (e.g. Eggdrop), zijn zij vervolgens via het IRC wars tijdperk (waarin zij bijvoorbeeld ingezet werden om IRC netwerken te DDoS-en en IRC kanalen af te pakken van hun oprichters) verworden tot het meest gebruikte instrument op internet om op criminele wijze geld te verdienen. Naast de gangbare zaken als spam zijn er tal van creatieve manieren bijgekomen zoals het beïnvloeden van aandelenkoersen

[6] of het “mijnen” van bitcoins [7].

Een belangrijke technische verandering is de manier waarop het netwerk aangestuurd wordt; de structuur van het botnet is iets wat de komende jaren veel gaan zou veranderen [12].

2.2 Moderne C&C structuren

De eerste aanpassingen ter vergroting van de robuustheid in IRC gebaseerde botnets waren voor de hand liggende zaken als het beveiligen van de toegang tot IRC kanalen met wachtwoorden en het binnen het kanaal onzichtbaar maken van de botclients voor iedereen behalve de botherder [14]. Er werden steeds vaker gehele IRC netwerken gebruikt; het kanaal dat diende voor het aansturen van het botnet was dan via alle IRC servers op het netwerk bereikbaar, wat voor C&C redundantie en dus vergroting van de robuustheid van het netwerk zorgt. Een vroege opvolger van C&C via IRC is de webapplicatie als C&C interface en het HTTP protocol voor communicatie binnen het netwerk [14]. Voor de structuur van het botnet is dit niet zo relevant, de C&C structuur blijft net zo gecentraliseerd als bij IRC.

Een zeer belangrijke ontwikkeling voor de botnets was de opkomst van Peer-to-Peer gebaseerde C&C structuren [12]. Door het decentraliseren van de aansturing van het netwerk werd veel gewonnen aan robuustheid. In een modern botnet hebben individuele peers slechts kennis van een relatief beperkt aantal andere peers, en zodoende leidt het reverse engineeren van de botcode (aangetroffen op een enkele peer) niet gelijk tot het ontdekken van alle andere peers. Een eventueel probleem met de P2P-gebaseerde C&C structuren is dat een gemodificeerde individuele bot door tegenstanders van het botnet ingezet zou kunnen worden om het botnet over te nemen. Immers, inherent aan de P2P basis kan elke bot in principe op elk moment moeten optreden als C&C server. Belangrijk voor botnets is verder DNS technologie; binnen botnets wordt gebruikt gemaakt van dynamische DNS, multihoming [14] en fast-flux DNS [30].

2.3 Casus: Storm

Storm was het grootste botnet dat op het internet aanwezig was in de periode 2007-2008 [1]. Over het formaat van het netwerk destijds lopen de schattingen uiteen: sommige onderzoekers houden 50 miljoen voor mogelijk terwijl anderen claimen dat er slechts een totaal van 1.5 miljoen infecties heeft plaatsgevonden [3]. De voornaamste taak van het netwerk was het versturen van spam; een tweede bezigheid was het uitvoeren van DDoS aanvallen. De naam 'Storm' is aan het botnet gegeven omdat de eerste infectie tot stand kwamen via emails met prikkelende subjects als 'Dodelijke storm maakt 139 slachtoffers', ten tijde van de storm Kyrill [5]. De voornaamste manier waarop het botnet zichzelf vergrootte is social engineering; er werden steeds nieuwe spamcampagnes uitgezet met actuele onderwerpen (de genoemde dodelijke storm, maar ook bijvoorbeeld kerstmis). Als de ontvanger van de email op de link klikt in de e-mail, wordt deze naar een site gestuurd waar geprobeerd wordt heimelijk software te installeren op het slachtoffer's computer door het misbruiken van niet gepatchte kwetsbaarheden in danwel de browser zelf danwel extensies van de browser (bijvoorbeeld de PDF reader). Dit noemt men browser exploitation [35], en tot eind 2007 was dit de enige manier waarop het Storm netwerk zichzelf vergrootte; vanaf 2008

werd er gebruik gemaakt van het exploiteren van kwetsbare netwerkservices [12]. Ook werden er rechtsreeks als mail-attachment EXE bestanden met de botcode gestuurd [11].

2.3.1 Storm kenmerken - Systeem niveau

De installatie van de kern K van de bot geschiedt via een eerste executable, de 'dropper' D. De binary D is bekend onder o.a. de namen Trojan.Peacomm en Peacomm.C [8], en is hevig geobfusceerd; de schrijvers van de code hebben hun best gedaan om het reverse engineeren zo moeilijk mogelijk te maken [11] . De aanvankelijke bot K is zelfstandig in staat om toe te treden tot het botnet (bootstrapping; [13]), waarna de resterende modules M_i gedownload kunnen worden. Deze modules hebben allen een specifieke functie voor de bot; er zijn bijvoorbeeld losse modules voor het versturen van spam en voor het uitvoeren van DDoS aanvallen. De installatie van K door D begint met het ontsleutelen van alle benodigde data. Deze eerste encryptie techniek die gebruikt wordt door de Storm bot is XOR encryptie [31]. Nu kan er een kernel-mode rootkit [§3.3] geïnstalleerd worden en is de verder noodzakelijke code voor K aanwezig. Deze code is met een tweede encryptie-algoritme genaamd TEA [34] versleuteld. Na ontsleuteling [10] wordt er een door D aangemaakte driver geïnjecteerd in het immer draaiende proces services.exe. Daarna wordt Windows Firewall aangepast om K te laten communiceren met andere peers in het Overnet (het P2P netwerk waar Storm gebruik van maakt; hierover meer in de volgende paragraaf), die K kan vinden op een lijst die door D is aangemaakt. Deze lijst heeft zo'n 150 [12] tot 250 [5] peers, waarvan sommige wel in het Storm botnet zitten en andere slechts gebruik maken van Overnet voor bijvoorbeeld filesharing. Als er met een Storm peer gecommuniceerd wordt, zal deze meer peers kenbaar maken; zo kan de bot ook vlak na het bootstrappen al communiceren met een groot gedeelte van de Storm peers.

De rootkit zorgt ervoor dat de botcode ongehinderd geïnjecteerd kan worden in services.exe nadat het OS reboot. Nu is K compleet en kan K de gewenste modules M_i downloaden door zichzelf kenbaar te maken aan de door D verstrekte peers. De benodigde modules M_i worden niet rechtstreeks verkregen van deze peers; deze peers sturen slechts URLs door, en deze URLs verwijzen naar de te downloaden modules. Hierna heeft K zichzelf gecompleteerd en is daarmee een volledig functionerende Storm bot.

2.3.2 Storm kenmerken - Netwerk niveau

Storm maakt gebruik van een P2P structuur zonder enkele centralisering genaamd Overnet. Echter, het gebruikte internetprotocol is het welbekende HTTP; er wordt dus gebruik gemaakt van P2P over HTTP [9]. Een botnet dat P2P gebruikt suggereert de aanwezigheid van twee wezenlijk verschillende types bots; bots die zowel als server en als client kunnen optreden (servents) en de bots die enkel als client kunnen optreden. Dit komt omdat niet alle hosts binnenkomende connecties kunnen accepteren; bots draaiende op dit type hosts kunnen in het botnet daarmee niet meer de rol als servent op zich nemen en zullen vervolgens enkel als client fungeren. Wel blijft de communicatie met de servents verlopen volgens het Overnet protocol. In Storm worden deze client-bots ingezet voor het versturen van spam (de primaire levenstaak van dit botnet) of DDoS aan-

vallen. Het Storm netwerk, inclusief hosts die niet als bot fungeren maar door de botherder worden gebruikt voor het aansturen van het netwerk, kan in vijf verschillende onderdelen ("niveau's") opgedeeld worden.

1. Een centrale C&C server; het vaste maar onvindbare punt op internet waar de communicatie van de botherder richting het botnet begint.
2. Een reeks proxy servers voor het verbergen van de C&C op niveau 1.
3. Een reeks "subcontrollers" die dienen als communicatie relays voor het contact tussen de botherder en de eigenlijke bots.
4. De servent-bots.
5. De client-only bots.

De niveau's 3-5 maken voor de onderlinge communicatie gebruik van het Overnet protocol. Overnet is ontwikkeld voor de filesharing applicatie eDonkey2000. Hoewel de applicatie wegens juridische strijd vroeg aan zijn einde is gekomen, bleef het Overnet protocol gebruikt worden voor sommige kleine projecten [5] en vond een nieuwe schare gebruikers in de Storm bots. Overnet als geheel doet dienst als distributed hashtable; het netwerk is als het ware een associatief array waarbij elke peer de value bij elke key kan verkrijgen door met de andere peers te communiceren. Overnet is een op filesharing gerichte implementatie van Kademlia, een P2P-gebaseerd <key, value> opslag- en distributiesysteem [12]. Kademlia definiëert voor elke peer een 160-bit node ID waarde. Voor elke node zullen de routing requests via nodes in zijn nabijheid verlopen, en deze nabijheid wordt uitgedrukt als de uitkomst van het bitwise XOR-en van zijn eigen node ID en die van zijn peers; de uitkomst is een getal $0 < X \leq 2^{160}$, en hoe lager dit getal hoe meer nabij de peer is. Een tweede relevant kenmerk van Kademlia is dat het netwerk bijhoudt hoe snel nodes handelen, en op basis daarvan trage nodes een volgende keer niet meer zal gebruiken voor het routen van verkeer. Een derde gedraging die Kademlia tot een robuust systeem maakt is dat communicatie op redundante wijze geschiedt; door parallel queries/replies te verzenden wordt voorkomen dat het uitvallen van individuele nodes leidt tot het mislopen van communicatie.

De lijst van peers wordt binnen Kademlia per peer gepresenteerd als een bucketarray met 160 buckets. Elke bucket correspondeert met een reeks afstanden, bijvoorbeeld bucket 23 bevat alle peers met een afstand $2^{23} \leq x < 2^{24}$. Elke bucket is geordend naar mate van uptime; nodes die langer meedoen in het P2P netwerk, genieten w.b.t. routing de voorkeur boven nodes die korter in het P2P netwerk zijn. Er zijn hiervoor twee redenen te noemen; de gemeten mate van uptime correleert positief met de te verwachten uptime [46], en een tweede reden is dat op deze wijze, indien de individuele buckets gevuld worden met kwaadaardige/dummie-nodes, alleen de "goede" nodes verkeer routen en er dus moeilijker op instigatie van een kwaadwillende partij een DoS voor het P2P netwerk bewerkstelligd kan worden d.m.v. het "vergiftigen" van de buckets [25].

Een value is binnen het Storm botnet een versleutelde pointer die verwijst naar een URL; op deze URL vindt de bot aanvullingen/updates zoals bijvoorbeeld tijdens het bootstrappen of uit te voeren instructies. In de alledaagse gang van zaken vraagt de bot dus aan zijn peers of zij een bepaald <key,value>

paar hebben, en dan zijn er twee mogelijkheden: deze value wordt direct geretourneerd, indien één van de aanvankelijk geraadpleegde peers dit <key,value> paar in zijn bezit heeft. De tweede mogelijkheid, de aanvankelijk geraadpleegde bots hebben zelf de gewenste value niet in hun bezit, leidt ertoe dat deze geraadpleegde peers nieuwe peers retourneren met een node ID zo nabij mogelijk de gevraagde key. Dit is een iteratief proces en leidt zodoende altijd tot het vinden van de bij een key behorende value indien een node in het netwerk over deze value beschikt. Indien geen enkele node over de value beschikt, uit zich dat bij een query in het herhaaldelijk terugkrijgen van dezelfde node IDs. Zodoende weet het netwerk of het wel of niet over een bepaalde value beschikt. De uiteindelijk value, indien gevonden, leidt naar een URL met instructies en de bot voert deze uit. Veelal zijn dit instructies om contact maken met een andere servent-bot in het netwerk. Er volgt dan een uitwisseling waarbij de client achtereenvolgens verzamelde data van de computer waarop hij draait meldt aan de servent, en vervolgens wacht op verdere instructies. Dit alles vindt plaats nadat de client zichzelf geauthenticeerd heeft; de client meldt zich bij de servent, deze zendt als challenge een bitstring en verwacht als response een bitstring die verkregen kan worden door de challenge string te XOR-en met een derde bitstring; de sleutel waar beide bots weet van hebben.

3 Defensieve technieken voor individuele bots

Moderne bots zijn adaptieve systemen die (i) interacteren met hun directe omgeving, het OS waarin de bot geïnstalleerd is. Ook interacteren zij met (ii) hun indirecte omgeving, het internet. In deze omgeving bevinden zich ook andere, equivalente entiteiten: de bots waarmee het botnet gevormd wordt. Met deze bots wordt frequent gecommuniceerd (bij P2P gebaseerde botnets). De gezamelijke bots en hun onderlinge communicatie constitueren een nieuwe entiteit, het botnet. Ook deze entiteit kan als een agent worden opgevat. De directe omgeving van deze agent is het internet. Beide “agentschappen” zullen afzonderlijk bekeken worden, te beginnen met individuele bots (dit hoofdstuk) en vervolgens het botnet als agent (volgende hoofdstuk).

Elke bot dient zich vanaf de installatie aan te passen aan zijn vijandelijke nieuwe omgeving, een operating system van Microsoft. In theorie kunnen er ook bots geschreven worden voor andere OS-en, maar in de praktijk komt dit slechts af en toe voor [15, 54]. Er is sprake van een voortdurende wapenwedloop waarin de schrijvers van bots technieken bedenken om hun code onontdekt en werkzaam te houden, waarop als reactie nieuwe tegenmaatregelen vanuit Microsoft, onderzoekers of anti-virus (AV) bedrijven volgen. De technieken die gebruikt worden door de bots om te overleven in hun omgeving, en ervoor te zorgen dat ze zo goed mogelijk hun opdrachten kunnen vervullen, zijn bij uitstek gedragingen die men “intelligent” placht te noemen; ondanks dat de processen die leiden tot de uiteindelijk waargenomen gedragingen geheel reductionistisch duidbaar zijn, is de algemene stellingname dat geavanceerde malware zich op “intelligente” wijze weet te verdedigen tegen pogingen tot verwijdering/detectie.

Een eerste voorbeeld van bot defensie is encryptie. Op verschillende wijzen wordt er d.m.v. het veranderen van de botcode getracht AV software niet te laten ontdekken dat de bot aanwezig is. Deze ver- en ontsleuteling kan op verschillende momenten plaatsvinden, en op verschillende wijzen. Een tweede

mogelijkheid is het nemen van anti-sandbox [55] maatregelen. Een derde voorbeeld van defensieve maatregelen die de bot neemt om zijn overlevingskansen te vergroten, zijn rootkit technieken. Samen zorgen deze drie eigenschappen ervoor dat de agent overleven kan in zijn omgeving. De eigenschappen zullen nu nader toegelicht worden, na een korte uitleg over de werking van antivirus software.

De primaire wijze waarop AV software tracht ongewenste software op te sporen is als volgt: er wordt gezocht binnen binaire code naar specifieke substrings waarvan het weet dat deze enkel voorkomen in ongewenste software. Deze substring wordt de signatuur genoemd. Het zoeken naar deze substrings gebeurt op twee fundamenteel verschillende wijzen: De eerste manier van het zoeken naar signaturen is door het lezen van de file vanaf de harddisk, en de resulterende string te scannen voor gezochte signaturen (static analysis). De tweede manier is als volgt: wanneer de code wordt uitgevoerd kan er gekeken worden welke bytes er in het geheugen komen te staan (dynamic analysis), en de AV software kan de eindgebruiker dan waarschuwen mochten deze bytes een ongewenste substring bevatten.

Naast op signatuur gebaseerde detectiemethoden wordt er door AV software ook gebruik gemaakt van heuristieken. Sommige AV software draait de te inspecteren code binnen een sandbox om in deze veilige omgeving de gedragingen van de code te bekijken. Mochten deze te veel lijken op gedragingen die eerder zijn vastgelegd bij reeds als malware geïdentificeerde code, dan kan de AV software ingrijpen door de code in "quarantaine" te stoppen (heuristiek-gebaseerde AV software).

3.1 Defensieve Techniek 1 - encryptie

Het is goed mogelijk dat de botcode, ingelezen vanaf harddisk, een geheel ander voorkomen heeft dan de verzameling eigenlijke instructies die uiteindelijk in het geheugen uitgevoerd gaan worden om bot te laten interacteren met zijn omgeving. De code zal dan een decryptiefunctie hebben om een ander deel van de code 'at runtime' te decrypten. De functionaliteit van de bot komt voort uit instructies die dus niet zijn terug te vinden wanneer de filecontent door AV wordt gelezen. Door per bot steeds een andere encryptiesleutel te gebruiken kan de versleutelde data per bot steeds een ander voorkomen hebben op disk, maar exact hetzelfde zijn wanneer gedecrypt in het geheugen (polymorphe encryptie). Een signatuur zou dan nog steeds bepaald kunnen worden, omdat de decryptiefunctie ongewijzigd blijft. Echter, er wordt dan geen rekening mee gehouden dat verschillende bots dezelfde decryptor zouden kunnen gebruiken en de verkregen signatuur is dan minder bruikbaar [39].

Voor de encryptie-functie kan een andere, afzonderlijke encryptietechniek worden toegepast; bijvoorbeeld het voor elke compilatie geautomatiseerd herschrijven van deze functie in de sourcecode. Door random variabledeclaraties toe te voegen, andere loopstructuren te gebruiken en extra functie aanroepen te gebruiken (waarvan het resultaat gelijk weer weggegooid kan worden) zal de decryptiefunctie binnen de gecompileerde code een steeds wisselende verschijningsvorm krijgen. Als er sprake is van een bot die zichzelf, inclusief en-/decryptie functionaliteit, kan herschrijven dan is er sprake van metamorphe code [37]. Een bot kan dit doen alvorens een kopie van zichzelf naar een vers geïnfecteerde host te sturen, met als gevolg dat de bot's binaire voorkomen steeds verandert.

3.2 Defensieve Techniek 2 - anti-sandbox maatregelen

De tweede defensieve maatregel is het proberen te constateren van een sandbox. Een sandbox is een gevirtualiseerde executieomgeving, waarin een proces niet met alle gewoonlijke rechten opereert [55]. Er mag bijvoorbeeld slechts in beperkte mate netwerkverkeer plaatsvinden, of informatie over het OS wordt verborgen gehouden. Een specifiek voorbeeld van een sandbox is een Virtual Machine (VM), een gevirtualiseerd OS wat draait binnen een gast OS [38]. Er zijn twee redenen waarom bots liever niet draaien binnen een sandbox omgeving. Er kan sprake zijn van een VM i.p.v. een echt OS, wat erop duidt dat de functionaliteit van de bot beperkt wordt (een VM is bijvoorbeeld over het algemeen minder online), en wat tevens kan wijzen op pogingen tot het reverse engineeren van de botcode [32]. De tweede reden is dat deze sandboxomgeving mogelijk tot stand komt doordat AV software de botcode op een veilige wijze wil runnen, om deze te inspecteren. Mocht de bot een sandbox architectuur als omgeving constateren, dan zal in veel gevallen de bot verder niet functioneren, proberen eventuele onderzoekers te misleiden door andere gedragingen te vertonen of een denial-of-service veroorzaken [41]. Een andere, geavanceerde vorm van defensie tegen een VM is ontsnappen aan de gecontroleerde omgeving (het gast OS) en trachten te draaien binnen het werkelijke OS. Maar op wat voor manier kan de bot vaststellen dat zijn omgeving een sandbox is?

Een voorbeeld is het aanroepen van obscure functies die een moderne versie van Windows slechts ondersteunt i.v.m. backward-compatibility. Deze zgn. "legacy" functies zijn vaak niet ondersteund binnen de sandbox architectuur [11], en zodoende kan de bot vaststellen dat hij draait binnen een gevirtualiseerde omgeving. Ook kan er door de botschrijver per sandbox omgeving gekeken worden wat de beperkingen in functionaliteit zijn, en checks hiervoor in de bot verwerken [40]. Een derde mogelijkheid is het zoeken naar specifieke processen die enkel in (specifieke) sandboxomgevingen voorkomen [42]. Als vierde en laatste categorie is er een gevarieerde verzameling VM-specifieke technieken [41]. Een klassiek voorbeeld hiervan is dat VMWare [43] de Interrupt Descriptor Table op een specifiek adres in het gast OS's geheugen zet; een enkele instructie kan dit verifiëren en zodoende is eenvoudig door een bot vast te stellen of deze in een gevirtualiseerde omgeving draait (Joanna Rutkowska, 2004).

3.3 Defensieve Techniek 3 - rootkits

Naast AV software en gevirtualiseerde omgevingen is een derde vijand van de bot zijn omgeving zelf; het OS waarin hij verkeert. Windows OS-en zijn door de jaren steeds bewuster geworden van beveiliging, en zijn zodoende steeds meer gewapend tegen indringers. De bot heeft baat bij het op een dusdanige wijze beïnvloeden van relevante systeem tools (bijvoorbeeld Windows Task Manager) dat deze tools niet langer de bot's bestaan kenbaar maken aan de eindgebruiker. De eindgebruiker is hier de (menselijke) gebruiker van de computer waarop het OS draait waarbinnen de bot zich onzichtbaar wenst te maken a.d.h.v. de rootkit. Het systeem moet normaal lijken te werken, maar in werkelijkheid bepaalde informatie achterhouden danwel manipuleren. In het theoretisch "ideale" geval is de bot niet op te merken door het OS, en door de eindgebruiker slechts theoretisch door zeer rigide middelen als het analyseren van buiten het OS om verkregen memorydumps. De software die hiervoor gebruikt wordt

noemt men een rootkit. Een relevant onderscheid is dat tussen user-mode rootkits en kernel-mode rootkits.

Een user-mode rootkit onderschept communicatie op userland [44] niveau; het gedeelte van het geheugen wat niet speciaal voor de kernel is gereserveerd. Een voorbeeld is het in het geheugen veranderen van per applicatie geladen DLL bestanden. Windows API functies die in deze DLL's staan kunnen zo een onbetrouwbaar resultaat geven. Een eenvoudig voorbeeld:

- Een applicatie A wil bekijken of een bepaalde registry key K voorkomt in het Windows register [44]. Een bot B die zich in hetzelfde OS als deze monitoringsapplicatie A bevindt heeft er baat bij dat K niet door A gevonden wordt; B's aanwezigheid zou erdoor verraden kunnen worden. Zodoende dat de bot vanaf installatie vergezeld wordt door een rootkit die hierbij helpt;

- A roept i.v.m. het checken van K's aanwezigheid de functie RegEnumKey aan, die te vinden is in ADVAPI32.DLL. De rootkit heeft deze functie herschreven nadat de DLL in het geheugen geladen is op een wijze die ertoe leidt dat de rootkit de functie aanroep onderschept;

- Nadat de functie aanroep onderschept is wordt de oorspronkelijke functie door de rootkit zelf aangeroepen. Het verkregen resultaat wordt door de rootkit gefilterd; de entries die K's aanwezigheid kunnen verraden worden verwijderd uit de verkregen lijst, terwijl de overige entries ongewijzigd blijven;

- Dit gefilterde resultaat is wat de eindgebruiker te zien krijgt. De getoonde informatie zal B's aanwezigheid niet verraden; de rootkit heeft zijn werk gedaan.

Een nadeel hierbij is dat het "patchen" per applicatie afzonderlijk moet gebeuren; voor elke applicatie moeten aanroepen van ingeladen DLL functies, die door de rootkit als ongewenst zijn aangemerkt, individueel onderschept worden. Dit komt omdat alle user-mode applicaties draaien binnen hun eigen stuk geheugen [36].

Er moeten door de rootkit veel applicaties gemonitord worden, en voor al deze applicaties moet er steeds het in gebruik zijnde geheugen op een gelijksoortige wijze aangepast worden. Stel bijvoorbeeld dat in het bovenstaande voorbeeld de applicatie A' ook de functie RegEnumKey aanroept. Om ook de output van A' aan te passen zou de rootkit de procedure toegepast op A moeten herhalen.

Een robuustere en handigere manier van het onderscheppen van Windows API functie aanroepen is het onderscheppen van de overeenkomstige system calls [45] op kernel-niveau, bijvoorbeeld als volgt: in het gedeelte van het geheugen gereserveerd voor de kernel is de System Service Descriptor Table te vinden, een tabel met alle adressen van de aanwezige system calls. Door deze tabel te wijzigen kan de rootkit ervoor zorgen dat er slechts één keer geheugen hoeft te worden aangepast, i.t.t. dat dit per applicatie dient te gebeuren. Dit is een techniek voor kernel-mode rootkits; de rootkit heeft toegang tot het kernelgedeelte van het geheugen en kan zodoende op een doeltreffender wijze te werk gaan. Voor een uitgebreide beschouwing van rootkits, zie [47].

3.4 Defensieve Technieken - conclusies

De technieken die moderne bots aanwenden voor hun individuele defensie zijn hiermee kort toegelicht; elk van de drie genoemde onderdelen van defensie behelst een groot wetenschappelijk onderzoeksgebied. Gezamenlijk zorgen deze onderdelen ervoor dat men spreekt van intelligent defensief gedrag van de bot. Door de omgeving te monitoren kan deze, wanneer noodzakelijk, aangepast worden door de agent. Dit doet hij om zijn secundaire verlangen om onontdekt te blijven te bewerkstelligen. Zijn primaire verlangen, het vervullen van zijn taken als onderdeel van het botnet, kan vervolgens vervuld worden.

4 Adaptieve Technieken van Botnets

Met wat voorstellingsvermogen kan een botnet gezien worden als een entiteit die zijn eigen verlangens en doelen heeft, en beschikt over middelen om deze doelen te bereiken. Een botnet communiceert niet met andere, equivalente entiteiten. Daarmee onderscheidt het botnet zich van een standaard agent, bijvoorbeeld de individuele bots die samen het botnet vormen. De adaptieve eigenschappen van het botnet dienen danwel voor het kunnen 'overleven' binnen het internet, danwel voor het vervullen van de dagelijkse bezigheden (cybercrime). Beiden zullen nu afzonderlijk besproken worden.

4.1 Botnets - zelfbehoud door zelfregulering

De eerste categorie adaptieve technieken dient ter regulering van de structuur van het botnet: het netwerk vergroot zijn eigen robuustheid. Een eenvoudig voorbeeld is het geautomatiseerd dupliceren van de C&C server software in combinatie met fastflux DNS. Een URL kan op deze wijze steeds naar een ander IP adres verwijzen. De eigenlijke host die de server draait en dus een cruciale rol in het netwerk vormt kan zo steeds wijzigen. Door deze voortdurende aanpassing wordt het onmogelijk gemaakt voor onderzoekers om uitgebreid verkeer uit te wisselen met de C&C server; deze draait slechts een korte tijd op een bepaald IP adres. Dit is voldoende voor bijvoorbeeld het verzenden van een enkele instructie, maar te weinig om de server uitgebreid te bestuderen.

Bij P2P netwerken zijn er tal van opties voor een netwerk om zijn eigen structuur aan te passen, zoals te zien is in het Kademlia protocol: als bij-effect van de "core-business" (het routen van queries/replies tussen peers) worden voorkeuren voor toekomstige routingen vastgelegd. Er is hier duidelijk sprake van een systeem dat 'leert' van zijn daden. Een uitbreiding van dit systeem is een punten systeem waarbij elke bot bijhoudt hoe adequaat zijn peers reageren, of er ongesigneerde commando's verstuurd worden (hetgeen duidt op pogingen tot intrusie/detectie) en of de individuele gedragingen van de bot niet teveel afwijken. Het rapportcijfer waarmee een bot zijn peers beoordeelt representeert de voorkeur voor deze peer wanneer de bot informatie naar de rest van het netwerk moet doorsturen [50].

4.2 Botnets - zelfbehoud door offensieve defensie

Behalve door het aanpassen van zijn interne structuur kan het botnet zijn overlevingskansen vergroten door eventuele dreigingen van buitenaf te detecteren en

onschadelijk te maken, het liefst in een zo vroeg mogelijk stadium. Een botnet heeft geen partners binnen zijn leefomgeving, enkel vijanden. Het is dan ook niet verwonderlijk dat botherders rekening houden met zowel de detectie als preventie van indringers.

4.2.1 Offensieve defensie - Intrusion Detection

Een eerste mogelijkheid is dat het netwerk ongebruikelijk verkeer waarneemt vanaf enkele hosts en kan besluiten de hosts die dat verkeer initiëren te frustreren. Dit is een gedraging die is waargenomen in 2007/2008 bij het bestuderen van het Storm botnet [5,48]. Maar hoe kan het netwerk ongelegitimeerd verkeer herkennen? Een mogelijkheid is dat wanneer een nieuwe bot zich meldt bij een C&C server, de C&C server enkele specifieke kenmerken ("footprint") van deze bot opslaat; bij een volgend bezoek van de bot aan de server, zal de server de bot herkennen. Mocht een server bemerken dat een reeds bekende bot herhaaldelijk dezelfde file download, een aan het botnet gelinkte webserver crawled [49] of anderszins verdacht gedrag vertoont dan is voor de server, en daarmee het botnet als geheel, aannemelijk dat deze bot slechte intenties heeft. De host waarop de bot draait (en zijn netwerk) zijn hiermee automatisch verdacht. In respons kan het botnet dan tegenmaatregelen nemen. Het Storm botnet deed dit door de verdachte host te DDoS-en, maar ook subtielere sabotage is denkbaar; bijvoorbeeld het misinformeren van de bot door sommige informatie niet door te sturen en sommige informatie wel.

Een tweede situatie is het constateren van een poging tot infiltratie; met grote hoeveelheden (gemodificeerde) bots kan een aanvallende partij invloed uitoefenen op het routen van queries/replies binnen het netwerk. Dit is een verschijnsel wat voor kan komen in P2P-gebaseerde botnets. Hierin weten de individuele bots over het bestaan van minstens enkele andere peers in het netwerk, en in sommige gevallen beschikken de bots over een lijst met alle andere peers. Deze lijsten worden bijgehouden; immers, in botnets is er een hoog verloop. Er worden voortdurend computers opgeschoond en er komen voortdurend nieuwe infecties bij. Door relatief veel nieuwe bots te introduceren kan een kwaadwillende partij proberen het botnet te "vergiftigen"; er kan substantieel veel verkeer worden gerout via nodes die gemodificeerd zijn door de aanvallers. Hierbij kan men er van uitgaan dat de hosts waarop de infiltrant-bots draaien gevirtualiseerd zijn: de infiltrant-bots worden normaliter ingezet door partijen die rechtshalve geen fysieke botnetten kunnen inzetten (universiteiten, AV bedrijven). Deze partijen gebruiken dan gevirtualiseerde OS-en. Op enkele fysieke machines kunnen zo 1000-en bots draaien met ieder een eigen extern IP adres [50].

Een defensieve maatregel hiertegen is elke nieuwe bot een computationeel middelmatig zware taak te laten uitvoeren alvorens de bot daadwerkelijk in het botnet op te nemen. Dit leidt ertoe dat de fysieke hosts die de grote verzamelingen VM's draaien niet voldoende rekenkracht hebben om voor alle VM's de rekentaak tijdig op te lossen. Hiermee vallen de legers van infiltrant-bots door de mand, en zodoende zullen zij in het minste geval verder niet mogen meedoen in het botnet, en mogelijk zelfs een DDoS aanval op hun lokale netwerk bewerkstelligen.

4.2.2 Offensieve defensie - Intrusion Prevention

Het liefst zou het botnet pogingen tot intrusie voorkomen. Een voorbeeld hoe dit te doen is als volgt. Het netwerk houdt verschillende AV bedrijven in de gaten door publicaties te scannen voor specifieke substrings; er wordt regelmatig gezocht via zoekmachines naar documenten op de websites van de AV-bedrijven/onderzoeksinstanties. Mocht een AV bedrijf bovengemiddeld veel publiceren waarin de gezochte strings voorkomen, dan laat dit een dreiging zien. Immers, het is te verwachten dat het AV bedrijf de wens koestert het netwerk uit te schakelen, en i.v.m. dat doel de publicaties heeft uitgegeven.

Door vervolgens de aan desbetreffend AV bedrijf toebehorende IP adressen/netwerken aan te vallen, wordt onderzoek doen voor dit bedrijf bemoeilijkt. Niet alleen het onderzoek naar het aanvallende botnet wordt lastiger, alle andere zaken waarmee het AV bedrijf zich bezighoudt en waarvoor het internet gebruikt wordt, worden gehinderd. Een bijkomend effect is dan het ontoegankelijk raken van het AV bedrijf's front-end, de publiek toegankelijke website: ervan uitgaande dat de website op een host in het aangevallen netwerk draait, raakt bij de DDoS aanval ook de website onbereikbaar. Dit leidt tot gezichtsverlies bij het AV bedrijf, en mogelijk tot misgelopen inkomsten: het AV bedrijf kan zichzelf niet goed verdedigen en dit maakt een slechte indruk op (potentiële) klanten. Deze defensieve maatregel is voor zover bekend nooit "in het wild" waargenomen, maar is een duidelijk realiseerbaar voorbeeld van adaptief gedrag in botnets. Er is sprake van een feedbackloop waarin de agent de dreiging uit zijn directe omgeving probeert in te perken; op het moment dat het aantal publicaties per aangevallen partij voldoende daalt zullen de aanvallen stoppen.

Alle bovenstaande gedragingen dienen puur voor het overleven van het botnet in zijn vijandelijke omgeving, het internet. Door deze basisbehoeften te vervullen, kan het botnet vervolgens beter zijn alledaagse bezigheden verrichten: het versturen van spam, het hosten van kwaadaardige websites etc.

4.3 Botnets - intelligente gedragingen buiten zelfbehoud

Behalve de overlevingstechnieken zijn er anno 2012 ook "alledaagse bezigheden" waarbij het botnet als intelligent systeem opereert. Een voorbeeld is het faciliteren van phishingactiviteiten. Stel voor dat vanuit een botnet een spamoperatie op touw wordt gezet om van ING klanten hun sessiedata/inloggegevens te verkrijgen. Nadat een percentage van de klanten gedupeerd is, kan het botnet de ING bank aanvallen met een DDoS. Dit leidt ertoe dat de gedupeerde klanten moeilijker inzicht kunnen krijgen in het feit dat zij bestolen zijn (er is immers tijdelijk geen internetbankieren mogelijk) waardoor de aanvallers langer onontdekt blijven, met alle gevolgen van dien.

Een ander voorbeeld van geavanceerd gedrag dat niet met zelfbehoud van doen heeft, is het aansturen van pump-and-dump fraude [51]. In eerste instantie wordt het botnet op verschillende wijzen ingezet om de koers van een bepaald aandeel kunstmatig te verhogen. Er wordt een gerichte spamcampagne opgezet, bots houden de markt in een aandeel levendig door regelmatig (kleine hoeveelheden) te in- en verkopen, er wordt met door bots gestolen informatie toegang verschaft tot specifieke brokeraccounts om daarmee het aandeel op te kopen, enzovoorts. Als het aandeel voldoende in waarde is gestegen, kan de vervolgfase

in werking worden gesteld: het aandeel wordt massaal verkocht, waarbij het botnet kan dienen om de internationale geldstroom te maskeren.

De technieken die gebruikt worden door botnets nemen toe in complexiteit naar gelang het internet zich ontwikkelt: er is sprake van een proces waarbij twee systemen elkaar non-stop beïnvloeden. Een recent voorbeeld is dat in respons op de "mobilisering" van het internet er zaken als Android-gebaseerde botnets ontstaan [52,53]. Het is te verwachten dat er steeds meer smartphones gebruikt gaan worden om bots op te draaien; o.m. omdat mensen zich bij deze computers minder bewust zijn van de gevaren van het draaien van onbekende code, en omdat de aantrekkingskracht voor botherders toeneemt naarmate er meer mensen smartphones gaan gebruiken. Zaken als encryptie en "lerende" P2P netwerken zijn voorbeelden van botnet-eigenschappen die al vaak worden toegepast, maar waarop nog veel geïnnoveerd wordt. Een inspirerende ontwikkeling voor botontwikkelaars is dat web-applicaties een steeds grotere rol spelen, ten koste van het gebruik van desktop-applicaties. Inherent aan de OS-onafhankelijkheid van deze web-apps is de mogelijkheid tot gestandaardiseerde aanvalsvectoren; een kwetsbaarheid in de web-applicatie kan aldus een grotere schare potentiële slachtoffers bereiken dan een kwetsbaardheid in een vergelijkbare desktop-applicatie. Een goed voorbeeld van multiplatform malware is Koobface [54], hoewel de onpatchbare kwetsbaarheid die hier misbruikt wordt de menselijke nieuwsgierigheid betreft.

5 Conclusie

Er is veel veranderd sinds de komst van de eerste IRC bots. Zowel de complexiteit en het formaat van de botnets nemen toe naarmate wij als wereld meer en meer het internet gaan gebruiken, een ontwikkeling waarvan het einde nog lang niet in zicht is. In deze scriptie is er gekeken naar de technieken die gebruikt worden binnen bots/botnets voor zover die bijdragen aan hun schijnbare intelligentie. Het scala aan mogelijkheden voor zowel de individuele bots als de botnets is hiermee niet uitgeput; zolang het internet zich blijft ontwikkelen blijven er nieuwe, steeds geavanceerdere technieken voor bot(net)s bijkomen.

Intelligent gedrag van (netwerken van) bots wordt binnen de A.I. traditiegetrouw op een meer theoretische wijze belicht; d.m.v. modale logica's kan er geredeneerd worden over agenten (bots) en hun onderlinge verhoudingen. Door puur te kijken naar gedragingen van adaptieve sytemen die uit praktijkgerichte motieven ontstaan, en die op een "natuurlijke" wijze meegegroeid zijn met hun leefomgeving, is er een ander perspectief op adaptieve systemen mogelijk. Tevens leidt dit ertoe dat de scriptie geschikt is voor zowel mensen met een AI-achtergrond als mensen met een CS-achtergrond.

Er zijn tal van afzonderlijke wetenschappelijke disciplines die in meer of mindere mate met botnets te maken hebben (AI, SEO, cryptografie, forensisch onderzoek, system/network engineering). Alleen al de (P2P) netwerkstructuur van botnets is bron van veel onderzoek [9,12,50], en ook in de toekomst zal er onderzoek gedaan worden naar o.a. het vergroten van de robuustheid van deze netwerken en de wijze waarop informatie binnen het netwerk zo onopgemerkt mogelijk gecommuniceerd kan worden. Er zal daarop weer vervolgonderzoek komen naar de mogelijkheden om toch het botnet te kunnen monitoren, waarop er weer innovaties onderzoek volgt om dit tegen te gaan, enzovoorts. Een bij-

zonder en kenmerkend verschijnsel hierbij is dat een groot gedeelte van het "onderzoek" niet voorhanden is: waar de beveiligingsbedrijven/universiteiten in de meeste gevallen redelijk veel publiseren is er in het tegenovergestelde kamp juist sprake van de noodzaak tot het achterhouden van informatie.

Er zal onderzoek gedaan moeten worden naar de wijze waarop het mobiele platform gebruikt wordt om botnets mee op te zetten, net zoals er onderzoek gedaan zal worden naar gelaagde encryptie van verkeer. Er moet worden onderzocht in welke mate patroonherkenning kan helpen bij het detecteren van botnetverkeer, en er zal meer onderzoek gedaan worden naar het ontsnappen aan gevirtualiseerde OS-en. Voorspellen wat er exact onderzocht gaat worden is onmogelijk; de evolutie van de botnets zal afhangen van de evolutie van de digitale techniek en de coevolutie van ons als mens.

6 Bronvermelding

1. http://www.pcworld.com/article/138694/
2. http://en.wikipedia.org/wiki/Storm_botnet/
3. http://www.neoseeker.com/news/7103-worm-storm-gathers-strength
4. http://www.youtube.com/watch?v=rIZS_zxHkHY('BlackHat2008:StormBotNetUpdatefromJoeStewart')
5. http://www.youtube.com/watch?v=_IHr9HvoEgM('24thChaosCommunicationCongress--lectureThorstenHolz')
6. http://www.informationweek.com/news/security/attacks/229400064
7. http://www.symantec.com/connect/blogs/bitcoin-botnet-mining
8. http://www.symantec.com/connect/blogs/new-peacomm-infection-techniques
9. "Analysis of HTTP2P Botnets; Case Study Waledac" (Dae-il Jang, Minsoo Kim, Hyun-chul Jung)
10. "Protocols and Encryption of the Storm Botnet" (Joe Stewart)
11. "Peacomm.C - Cracking the nutshell" (Frank Boldewin)
12. "Peer-to-Peer Botnets: Overview and Case Study" (Grizzard et al)
13. "Bootstrapping Peer-to-Peer Networks" (GauthierDickey, Grothoff)
14. "botnets the killer web app" (Craig Schiller, Jim Binkley, Gadi Evron and David Harley)
15. http://www.csoonline.com/article/681722/cross-platform-botnet-targets-both-windows-and-mac-users
16. https://tao.truststc.org/Members/vern/worms_and_botnets
17. http://en.wikipedia.org/wiki/IRC_daemon
18. http://tools.ietf.org/html/rfc1459
19. http://www.egghelp.org/tclhtml/3478-4-0-0-1.htm
20. http://www.webopedia.com/TERM/D/DDoS_attack.html
21. http://searchsecurity.techtarget.com/definition/smurfing
22. http://vxchaos.2hell.com/
23. http://ddos.arbornetworks.com/2009/08/twitter-based-botnet-command-channel/
24. http://en.wikipedia.org/wiki/Phoning_home

25. "Kademlia: A Peer-to-peer Information System Based on the XOR Metric" (Maymounko, Mazi)
26. http://www.eskimo.com/~nanook/tidbits/2007/04/what-is-shell-account.html
27. http://www.writersservices.com/www/t_worm.htm
28. http://en.wikipedia.org/wiki/netsplit
29. http://searchsecurity.techtarget.com/definition/zero-day-exploit
30. http://www.darkreading.com/security/perimeter-security/208804708/on-the-trail-of-fast-flux-botnets.html
31. http://en.wikipedia.org/wiki/XOR_cipher
32. http://www.networkforensics.com/2010/12/13/vm-detection-by-in-the-wild-malware/
33. http://en.wikipedia.org/wiki/Executable_compression
34. http://www.shokhirev.com/nikolai/programs/code/Cryptography/TeaSet.html
35. http://www.mywot.com/en/online-threats/browser-exploits
36. http://www.symantec.com/avcenter/reference/windows.rootkit.overview.pdf
37. http://vx.netlux.org/lib/vmd01.html
38. http://en.wikipedia.org/wiki/Virtual_machine
39. "The Art of Computer Virus Research and Defense" (P.Szor)
40. "Detecting Environment-Sensitive Malware" (M.Lindorfer)
41. "Attacks on Virtual Machine Emulators" (P.Ferrie)
42. "Proceedings of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats" (U.Bayer)
43. http://www.vmware.com/
44. http://en.wikipedia.org/wiki/Windows_registry
45. http://en.wikipedia.org/wiki/System_call
46. "A Measurement Study of Peer-to-Peer File Sharing Systems" (S.Saroiu)
47. http://www.amazon.co.uk/Rootkit-Arsenal-Escape-Blunden-B/dp/1598220616/ref=sr_1_1?s=books&ie=UTF8&qid=1325952276&sr=1-1
48. http://www.theregister.co.uk/2007/10/25/storm_worm_backlash/
49. http://en.wikipedia.org/wiki/Webcrawler
50. "Towards Next-Generation Botnets" (Hund, Hamann, Holz)
51. http://en.wikipedia.org/wiki/Pump-and-dump
52. http://www.scmagazine.com/android-botnet-infections-on-the-uptick/article/211670/
53. "Andbot: Towards Advanced Mobile Botnets" (Xiang et al)
54. http://www.theglobeandmail.com/news/national/time-to-lead/internet/the-untouchable-hackers-of-st-petersburg/article1795650/
55. http://en.wikipedia.org/Sandbox_(computer_security)