LiNSEP een nieuw e-mail protocol

Dennis Brentjes

07-07-2010

Inhoudsopgave

1 Inleiding

Voor ons onderzoek voor het vak Research & Development hebben wij onderzocht of het mogelijk is om een spamvrij e-mailprotocol te schrijven. Ons onderzoek heeft betrekking op wat er precies moet gebeuren om dit te realiseren en wat de makkelijkste manier is om het ook daadwerkelijk te implementeren. Ook hebben we onderzocht of het nieuwe protocol niet te traag is en of het wel echt spamvrij is. Hierover staat in de methode beschrijving, resultaten en conclusie meer uitleg. Daarnaast staat er in het theoretisch kader nog meer uitleg en onderzoek naar rand onderwerpen die we gebruiken of aanstippen in de rest van ons verslag.

2 Theoretisch Kader

2.1 Inherente problemen van e-mail

Er zijn een aantal problemen met de huidige implementatie van e-mail. De oorzaak van deze problemen vindt men in een aantal factoren. Sommige van deze factoren zijn te verhelpen andere niet.

Het eerste probleem is de natuurlijke "slechtheid" van de mens om spam te versturen en daardoor de maatschappij op te zadelen met extra kosten en frustratie. Hier kan niets aan gedaan worden. Het is wel een bron van inspiratie voor een mogelijke aanpak. Namelijk om de spammers zelf aan te pakken in plaats van spam af te vangen. Hier gaan we het later uitgebreid over hebben.

Het tweede probleem is de e-mail infrastructuur of wel de servers die de e-mail versturen. Dit zijn in de meeste gevallen "relay stations"; dat wil zeggen er komt een e-mail bij een server aan en deze wordt vervolgens doorgestuurd naar een volgende server. Deze volgende server kan dan weer een relay- of ontvangende server zijn. Door deze aaneenschakelijng van servers ontstaat er een onveilige keten waarbij die net zo veilig is als de zwakste link in de keten, waardoor er een grotere kans is dat de Spam e-mail wel aankomt bij de persoon waar deze e-mail aan geadresserd was, terwijl deze persoon de e-mail niet wou ontvangen. Je zou denken dat het bovenstaande moeilijk te realiseren valt, maar niets is minder waar. Iedereen kan een SMTP (het huidige e-mail verzend protocol) server aanmaken, daarnaast kun je deze (als je een beetje handig bent) aanpassen om afwijkend gedrag te vertonen; bijvoorbeeld het toelaten van gespoofte e-mails of aangepast headers. Dit probleem is ook niet op te lossen. Om dit te realisren zou je mensen moeten dwingen om een eerlijke server te maken, maar dit is niet mogelijk omdat het SMTP-protocol te goed gedocumenteerd is.

Het derde probleem is de leeftijd van het protocol. Het SMTP-protocol stamt uit de jaren ‘60 en is onstaan in ARPAnet (de voorganger van het internet). Het is een samenraapsel van een aantal messenging protocollen die toen-ter-tijd door elkaar gebruikt werden om informatie te verzenden. Deze zijn gebundeld en eind jaren ‘80 opgenomen als officieel protocol. Het feit dat er geen spam bestond in de jaren dat het protocol tot stand kwam heeft ervoor gezorgd dat er geen faciliteiten zijn om spam te bevechten in het protocol zelf. In reactie hierop zijn er een aantal uitbreidingen op het SMTP-protocol gekomen in de loop van de jaren ‘90 en ‘00 (namelijk eSMTP en SMTP-auth). Deze zorgen voor betere identificatie en controle van e-mail verzenders, maar door de "relay" structuur en "backwards compatibility" voor SMTP(1.0) gaat het nog steeds mis. Nu de spammers botnets gebruiken is het identificeren van de verzender zelfs geen oplossing meer en bovendien biedt het geen oplossing voor het spoofen van e-mail adressen.

In ons project proberen we de bovenstaande problemen het hoofd te bieden door een beter e-mail protocol met ingebouwde spampreventie te bedenken.

2.2 Huidige spam preventie

Er worden op dit moment twee typen spam preventie gebruikt. De een is "content-based" en de andere is "sender-based". De eerste is naar mijn mening de meest bekende. Dit zijn namelijk de spam filters die je zelf kan installeren of door je e-mail provider worden aangeleverd. Deze spam filters kijken naar de inhoud en werken meestal door middel van sleutelwoorden om te beslissen of een bericht spam is of niet. Dit was een goede oplossing in het verleden maar met de opkomst van betere internetverbindingen en de komst van reclame e-mail met plaatjes niet in alle situaties toepasbaar. Ook is deze aanpak vervelend voor de gebruiker omdat ze nu via de spambox van hun e-mail account toch nog in contact komen met deze e-mailtjes.

Het tweede type spam preventie is "sender-based", deze methode probeert om de spam te stoppen voordat het aankomt bij de persoon waar deze naar gestuurd wordt. Dit kan men op een aantal manieren doen. Zo is het mogelijk om notoire spam verstuurders te blacklisten zodat men hier geen spam meer van ontvangt, maar met de komst van botnets is dit niet meer van toepassing. De reden hiervoor is dat deze botnets bestaan uit duizenden pc's van "normale" mensen die geïnfecteerd zijn door een bepaald virus dat ervoor zorgt dat spammers via deze computers spam kunnen versturen. De eigenaar van de geïnfecteerde computer merkt meestal niet dat zijn computer onderdeel uitmaakt van een botnet, omdat pc's tegenwoordig veel rekenkracht hebben. Met als gevolg dat je erg veel adressen moet blacklisten om resultaat te krijgen. Daarnaast als een peroson met een geïnfecteerde computer een e-mail verzendt naar een persoon met "sender-based" spam preventie dan komt het bericht niet bij de ontvanger aan, waardoor beide gedupeerd worden.

Een andere mooie "sender-based" systeem dat we zijn tegen gekomen is heel erg gesofisticeerd. Hier is een klein stukje uit hun artikel.

> In particular, we believe our work offers three contributions: First, we produce a general approach for inferring e-mail generation templates in their entirety, subsuming prior work focused on particular features (e.g., e-mail header anomalies, subject lines, URLs). Second, we describe a concrete algorithm that can perform this inference task in near real-time (only a few seconds to generate an initial high-quality regular expression, and fractions of a second to update and refine it in response to subsequent samples), thereby making this approach feasible for deployment. Finally, we test this approach empirically against live botnet spam, demonstrate its effectiveness, and identify the requirements for practical use. [1]

Wat deze aanpak in principe doet is kijken hoe verschillende spam e-mailtjes van hetzelfde botnetwerk op elkaar lijken. Spam mail wordt namelijk gemaakt vanuit een template. Dit is een e-mail met variabele velden in bijvoorbeeld het From veld of een achternaam in de body van de e-mail. Zodat de spammers e-mail kunnen versturen met dezelfde inhoud maar deze telkens net iets anders maken waardoor de "content-based" systemen de e-mail niet als spam klasseren.

Dit anti-spam systeem maakt dan (bijna) realtime een reguliere expressie waar deze e-mailtjes allemaal op matchen en dus veilig kunnen worden gefilterd. Dit allemaal bijna zonder 'false positives'; niet-spam die als spam wordt geïdentificeerd. Je moet jezelf echter wel afvragen of je überhaupt 'false positives' wilt hebben, want stel dat je hierdoor een belangrijk e-mailtje mist, zoals een herrinering aan een tentamen of een kamernet reactie. Als dit systeem ooit geperfectioneerd wordt zou dit een van de betere oplossingen zijn voor het spam probleem. Daarnaast zeggen de schrijvers van bonvenstaand artikel zelf dat het systeem in zijn huidige staat beter gebruikt kan worden als aanvulling op de al bestaande oplossingen.

2.3 Onze visie op spam preventie.

We willen de problemen als volgt aanpakken. We willen zorgen dat e-mail die verstuurd wordt vanaf een botnet wordt tegenhouden. We doen dit door een Captcha ("Completely Automated Public Turing-test to tell Computers and Humans Apart") te gebruiken. Hierdoor kunnen botnets niet automatisch massaal e-mailtjes versturen vanaf pc's van nietsvermoedende thuisgebruikers. Hierdoor onstaat er een probleem met andere "goede" geautomatiseerde e-mail zoals nieuwsbrieven en validatie e-mailtjes. Hiervoor zal iemand die een server implementatie maakt het waarschijnlijk handig vinden om een concept van black- en whitelisting toe te passen. zodat mensen (en misschien de server zelf) bepaalde afzenders, zoals de RU e-mail accounts, toe kunnen laten zonder dat deze afzenders een Captcha moeten invullen. Iets dat onmogelijk is voor geautomatiseerde systemen, dit nemen we in ieder geval aan. Dit is een beetje extra werk voor de gebruiker, maar dit beetje extra werk weegt niet op tegen de voordelen die de afname van spam in het algemeen met zich meebrengt voorzien door de afname van spam in het algemeen.

Want zelfs als de Captcha's niet helemaal veilig zijn zullen de spammers meer moeite moeten doen om hun spam te laten aankomen. Hierdoor zullen de kosten oplopen en het gewin van spam verzender zal minder zijn, omdat het meer tijd kost om dezelfde hoeveelheid e-mail te versturen. Een verdubbeling van de tijd die het kost om een spam mailtje te versturen, zou er al voor zorgen dat de winst marge van de spammer halveert (dit is haalbaar door bijvoorbeeld een computer aan tekstherkenning te laten doen). Ook zou de door het botnet geïnfecteerde thuiscomputer meer last ondervinden van de infectie en zal de gebruiker eerder doorhebben dat zijn pc geïnfecteerd is en stappen ondernemen.

2.4 Ons protocol en de internet wereld.

Om ons protocol te laten werken hebben we een transfer protocol nodig, namelijk UDP of TCP. We hebben hebben naar de voordelen en nadelen van UDP en TCP gekeken, zodat we een goede keuze konden maken. Hier is een stukje van de specificatie van TCP.

> A connection is fully specified by the pair of sockets at the ends. A local socket may participate in many connections to different foreign

> sockets. A connection can be used to carry data in both directions, that is, it is "full duplex". [3]

Dit wil zeggen dat de verbinding blijft bestaan ook al zend je er geen data overheen en ben je dus constant bezig om die socket te bedienen. De enige manier om de connectie te verbreken is door expliciet te zeggen dat de verbinding klaar is (of natuurlijk een time out). Iets wat met veel connecties tegelijk best zwaar kan zijn voor de server. Aan de andere kant willen we wel de e-mail meteen afhandelen. In sommige gevallen moet de gebruiker namelijk een Captcha invullen. Als we dit met TCP doen weten we meteen waar we dit naar toe moeten sturen, zo niet moeten we bijhouden waar de gebruiker is en dan terug sturen wat natuurlijk ook de server weer extra belast.

als 2e keus hebben we UDP. UDP is een "loose connection" protocol. Hier volgt weer een stukje uit de specificatie.

> This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [4]

Dit is een protocol wat een pakketje opstuurt naar een doel computer en niet garandeert dat deze daar aankomt en ook geen uitspraak doet over de volgorde waarin meerdere pakketjes aankomen. Dit zijn beide problemen die op te lossen zijn met een software methode, maar dit is best moeilijk en erg latency afhankelijk. Bovendie voorzien wij een probleem als de client de verbinding verbreekt voordat de captcha is aangekomen, waardoor de client in de veronderstelling is dat de e-mail verstuurd is, maar dit echter niet het geval is. Daarom kiezen we voor het TCP protocol

3 Methode.

3.1 Is ons nieuw protocol niet een te grote last voor de gebruiker?

Het eerste dat vastgesteld moet worden is wat "een te grote last" precies is om dit te doen moeten we de vraag beantwoorden wat mensen irritant vinden als ze een computer gebruiken. Dit is heel belangrijk omdat ons systeem een "Sender-based" spam preventie is en heel erg veel plichten legt bij de verzenders. Verder zijn mensen ongeduldig. Als ze dus iets van de computer gedaan willen hebben moet dit snel gebeuren. Dit kun je makkelijk vast stellen door vragen te stellen aan vrienden, familie en studiegenoten die betrekking hebben op dingen die gebruikers irritant vinden aan hun computer of die van de universiteit. Vaak krijg je een antwoord in de vorm van "het ding doet niet wat ik wil" of "ik heb 20 minuten gedaan over een print taak". Dus nemen we in alle redelijkheid aan dat *tijd* een belangrijke factor zal zijn in ons onderzoek. Het grootste probleem is spam en het is begrijpelijk dat wij aan het belangrijkste probleem geen andere bronnen van ergernis willen toevoegen zelfs al voorkomen deze spam. Helaas is al gebleken dat we een vorm van Captcha's ("Completely Automated Public Turing-test to tell Computers and Humans Apart") moeten gebruiken. Deze zijn ook een bron van irritatie onder studenten en jongeren. Zij komen deze turing tests regelmatig tegen en als ze slecht gemaakt zijn kosten deze tests ook erg veel tijd.

Wij willen dus graag een spambot maken voor ons protocol en deze vervolgens e-mailtjes te laten sturen en kijken hoelang deze hier gemiddeld over doet. Hiervoor moeten we wel, het unieke aan ons protocol, de Captcha laten varen en 1 vaste Captcha meegeven in plaats van een willekeurige aangeleverd door een server. Omdat we e-mailtjes willen sturen met een geautomatiseerd systeem en geen black- of whitelist hebben geïmplementeerd.

3.2 Werkt ons protocol wel?

De tweede zaak waar we naar moeten kijken, als ons protocol te onhandig is wordt deze toch niet gebruikt, is of het protocol alle spam tegen kan houden. Dit ligt aan de definitie van spam en de manieren waarop deze worden verzonden. De definitie voor spam die wij gebruiken luid als volgt "e-mail berichten die naar veel meer mensen gestuurd wordt dan een mogelijk geïnteresseerde groep." Dit wil zeggen dat de e-mail verstuurd wordt door botnets en dus geautomatiseerde e-mail verstuurders. We moeten dus kijken of het wel onmogelijk is voor computers om een e-mail te versturen tenzij de ontvangende partij deze geautomatiseerde verstuurder kent en toelaat. Dit doen we door onze methode op een formele manier (in natuurlijke taal) te verwoorden. Hierdoor kunnen we dan een conclusie trekken.

4 Resultaten.

4.1 Stress test.

Hier zijn de resultaten van de "stress test" van onze test implementatie.

Testnummer	aantal seconden / 100 e-mailtjes.
1	52.43 s
2	49.88 s
3	51.79 s
4	48.72 s
5	54. 75 s
6	50.54 s
7	51.23 s
8	48.15 s
9	50.90 s
10	52.04 s

4.2 Korte bewijsvoering.

Als er een Captcha moet worden ingevuld

- De Captcha kan niet door een ander persoon opgelost worden
  - De Captcha kan niet doorgestuurd door zelf een e-mail te ontvangen. Het antwoord zelf wordt namelijk niet doorgestuurd, alleen een bewijs (md5sum) dat de oplosser ook de oplossing heeft
  - Via high traffic sites bezoekers laten oplossen. Dit is echter onwaarschijnlijk/zeer weinig [5]

Als de zender op een whitelist van de gebruiker of server staat

- Dan heeft de gebruiker zelf toestemming gegeven, om alle e-mail van deze zender in de inbox te bezorgen
- Dan is het zijn eigen domein en zegt van zich zelf dat hij geen spam verstuurd.

Als de zender op een blacklist van de gebruiker of server staat

- Dan heeft deze zender blijkbaar al eerder geprobeerd te spammen of heeft een slechte reputatie, dan is er geen probleem.

5 Analyse

5.1 De responsie tijd van ons protocol

Als we de verwerkingstijd nemen van 100 e-mailtjes dus het gemiddelde van de gevonden waardes namelijk 51.04 seconden. Vervolgens delen we dit gemiddelde door 100 dit levert ons een verwerkingssnelheid op van 0.510 seconden per e-mailtje. Ons e-mailtje was ~1,5 MB groot; een 1.4 MB Captcha en nog wat "plain text". Dit in combinatie met een gemiddelde netwerk snelheid van ongeveer 3.5 MB/s levert dit ons een minimum verwerkingstijd van 0.42 seconden op. onze implementatie, die dus puur en alleen data verwerkt en in een bepaalde volgorde op een rijtje zet, is verrassend inefficiënt. Verder zijn deze resultaten niet te vergelijken met die van SMTP omdat daar nog een encryptie overheen gaat en deze over relay stations meestal pas na 2 of 3 stappen op de plaats is waar deze moet zijn.

Als we dit toch even vergelijken met SMTP (deze doet er onder goede omstandigheden 2.1 seconden over). Geeft dat aan hoe veel meer SMTP doet ten opzichte van ons protocol, want zelfs met "DNS lookups" en beperkte bandbreedte over het internet zou je geen verhouding van 1 staat tot 4 kunnen recht rekken. Er zijn dus die andere factoren die er voor zorgen dat ons protocol snel is maar daardoor onveilig of zelfs niet functioneel genoeg. [6]

5.2 Houden wij spam tegen?

Hier is een korte analyse van de in 4.2 gepresenteerde resultaten van ons onderzoek naar captcha's en onze visie op het gebruik van deze "turing test".

- Als we uit gaan van de definitie van Captcha's dan is het mogelijk om computers en mensen uit elkaar te houden in internetcommunicatie
- Als we van een gebruiker kunnen bepalen of deze een computer of een mens is kunnen we computers weren uit ons systeem.
- Als we alle computers (als in computers zonder beheerders toestemning) weren uit ons systeem is het niet mogelijk om met botnets ons systeem te benaderen.
- Als botnets ons systeem niet kunnen benaderen is het niet mogelijk om spam te versturen volgens de door onze aangegeven definitie van spam (zie theoretisch kader).

6 Conclusie.

6.1 Hoe is de responsie tijd van ons protocol.

Als we onze resultaten erbij pakken is er geen probleem om de conclusie te trekken dat ons protocol snel genoeg is, maar ons onderzoek blijkt niet representatief te zijn. Er zijn namelijk een aantal problemen. ten eerste hebben we geen gebruik gemaakt van een variabele Captcha's iets wat de hoofdzaak is van ons protocol. Ten tweede is er geen encryptie bij ons protocol en dit gebeurt ook niet in onze client. Iets wat nu wel gebeurt bij e-mail en dus bij ons minder tijd kost dan bij het oude manier van e-mail versturen. Ook is er niet getest in een internet omgeving maar een netwerk omgeving. Hier zijn de latency ook veel lager en zijn er geen vertragingen van bijvoorbeeld DNS servers of internet hardware van mensen thuis. Ook hebben we onze Black- en Whitelisting niet kunnen testen omdat we dan meerdere computers een client moesten laten draaien en we hebben dit thuis getest en niet op de universiteit, omdat er geen verbinding kon gemaakt worden op het universiteitsnetwerk.

Hierdoor kunnen we dus geen conclusies trekken over de last voor de gebruiker en dus verzender van dit protocol. We verwachten dat, de oplos tijd van de Captcha niet meegerekend, de last voor de gebruiker best wel meevalt. Hierdoor wordt het dus de vraag of je het erg vindt om mensen op te zadelen met Captcha's en dit is weer een ander onderzoek waar wij niet aan toegekomen zijn.

6.2 Zijn Captcha's genoeg?

In principe wel, als we maar de eerste aanname kunnen hard maken. Dit is echter niet het geval. Er zijn computers die Captcha's kunnen lezen en dus vervolgens oplossen. Dit duurt wel lang en is geen foutvrij systeem. Dus dit is wel te detecteren. Als dan een client meerdere keren achter elkaar de Captcha niet correct invult kun je deze ook uitsluiten voor het systeem door deze toe te voegen aan de blacklist. Het is dus, zoals in de theoretisch kader al aangestipt, voldoende dat we de clients genoeg afremmen zodat ze in ieder geval minder schade aanrichten en minder winstgevend zijn.

Het is dus niet te concluderen dat ons protocol helemaal spam vrij zal zijn, maar het zal wel een goede stap in die richting zijn, maar er zijn ook andere systemen zoals Judo die op een betere manier deze spambot problematiek tegenhouden zonder het ongemak van een ander protocol.

Referenties

[1] Andreas Pitsillidis, Kirill Levchenko, Christian Kreibichy, Chris Kanich, Geoffrey M Voelker, Vern Paxsony, Nicholas Weavery, Stefan Savage; Botnet Judo: Fighting Spam with Itself; onderzoeksrapport; International Computer Science Institute Dept. of Computer Science and Engineering Berkeley, USA University of California, San Diego, USA; 2009

[2] Sara Robinson; Human or Computer? Take This Test; New York Times, Science; 10 December 2002; 2 pagina's

[3] Jon Postel; Transmission control protocol; documentatie/specificatie; Information Sciences Institute University of Southern California; 1981

[4] J. Postel; User Datagram Protocol; documentatie/specificatie; IETF; 1980

[5] Vikas Bajaj; Spammers pay others to answer security tests; New York Times, Technology; 25 April 2010; 1pagina

[6] Joachim Charzinski; Observations in e-mail Performance; onderzoeksrapport; Siemens Information and Communication Networks; 2002