DYNAMICALLY BLOCKING ACCESS TO WEB PAGES FOR SPAMMERS' HARVESTERS

Tobias Eggendorfer
ITIS e. V. Institut für Technik Intelligenter Systeme
An-Insitut der Universität der Bundeswehr Neubiberg
Werner-Heisenberg-Weg 39
85579 Neubiberg, Germany
tobias.eggendorfer@unibw.de

Jörg Keller
FernUniversität in Hagen
Lehrgebiet Parallelität & VLSI
58084 Hagen, Germany
joerg.keller@fernuni-hagen.de

ABSTRACT
Almost all current anti spam measures are reactive, filtering being the most common. But to react means always to be one step behind. Reaction requires to predict the next action of the attacker. So the focus on fighting spam should rather be on prevention. Current proposals focus on fixing SMTP's lack of authentication, but introduce two new major problems: First, all current attempts break existing SMTP functionality and, second, it seems to be hardly possible to enforce a change of SMTP world wide. Therefore other preventive measures should be implemented. The most promising approach is to prevent spammers from collecting email addresses. Several proposals show ways to obfuscate addresses on web pages and to create HTTP tar pits in order to catch spammers' harvesters. In our previous work, we combined a HTTP tar pit with a SMTP tar pit and found it to be very effective in trapping harvesters.
Here, we extend the use of the combined tar pit to identify harvesters and to dynamically block access to web pages for harvesters, because of the combined tar pit's high efficiency. We present a test setup to validate the effectiveness of our tool. As the experiment is still running, we can only report on preliminary findings so far.

KEYWORDS
Spam, harvester, dynamic access control, SMTP, HTTP, tar pit, proactive protection

1. Introduction

Participants of email communication currently suffer from lots of unsolicited commercial email, commonly known as spam. As current anti-spam measures such as filtering, blacklisting, and greylisting are not effective enough, and efforts to correct spam-enabling deficiencies in the underlying SMTP protocol probably will not have any effects in the next years, we feel that one way to take immediate action is to prevent spammers from collecting email addresses, which they mostly do by visiting web pages with harvesters. In previous research [1] we provided methods to obfuscate email addresses on web pages so that harvesters cannot recognise them, and methods to trap harvesters in a tar pit [2] so that they cannot visit regular web pages. In this research we combine both approaches: we use a tar pit to identify harvesters, and consequently use this information to block access to regular web pages for those harvesters. We report on the implementation issues and on a test setup to validate the effectiveness of our approach. As the experiment is still running, we can only discuss some very preliminary findings. Yet we are confident to be able to present more results on the final version of this paper.
The remainder of this paper is organised as follows. In Section 2, we review the state of the art in anti-spam measures. In Section 3, we summarise information on a http tar pit in order to provide the information necessary to understand our combined approach. In Section 4 we present our approach to identify harvesters with a tar pit. In Section 5, we discuss how to block access to web pages based on information received from the tar pit. In Section 6 we present the test setup for validation. In Section 7 we conclude and give an outlook on future research.

2. Current anti spam measures

2.1. Reactive methods

Currently, most relevant methods to reduce the amount of unsolicited commercial email (UCE, spam) in any user's inbox rely on some kind of filtering. Different filtering technologies are in use: probably the best known, but simplest is blacklisting, i. e. each SMTP client's IP-address connecting to a SMTP server is tested against a list of known spamming hosts. When invented back in the late 1990s, this approach both helped filtering spam and supported the demand of switching off so called open relays. But this solution has also been known for having heavy side-effects: Almost all big email-providers have already been blacklisted on at least some of the widely available blacklists [3][4][5]. By now, the increasing usage of so called zombie PCs, i.e. mostly Windows computers infected by some worms, to send spam from, turned those black lists more and more useless: They either have to block entire subnets known to be used by dial-in providers to block potential abuse and thereby block thousands of legitimate mail users that run their MTAs on Unix machines at home, or their filtering becomes more and more ineffective, as spam is not relayed anymore through open relays.
Other solutions are content filters applied to the header and / or the body of a mail message. Filtering is mainly

based on a “bad-word-list”. This solution needs individual fine tuning, off-the-shelf products are often too imprecise, e.g. a bank clerk can not filter on “mortgage”.

To improve filtering, scoring-mechanisms to weight words and other signs that a message might be spam were implemented. Those filters also require lots of fine-tuning and maintenance: Spammers are reported to register mail accounts with online services known to have spam filtering and to first test their spam against those filters. This leads to a permanent “one-step-behind”-situation for filters, no matter how advanced content-filtering becomes [6].

Collaborative filtering is yet another approach to identify spam: To do so, large mail providers analyse mails their customers get and compare them to both mails directed to other customers and mails received on special honeypot addresses. In [7] some interesting statistics on this kind of filtering have been published: A spammer might wait between delivering two spam messages to different accounts at the same provider for some time. In only 92% of the cases, the two messages were received within 15 minutes of each other. This approach would require to store and delay each incoming message for at least 15 minutes to identify it with 92% probability as a part of a spam run. Storing messages on a MTA and comparing them requires huge amounts of both disk space and computing power and it rises important privacy questions. And again, spammers learnt their lessons: They already include random words and characters somewhere in the message to make it look different to collaborative filters.

Another still reactive way to reduce spam is greylisting, i. e. forcing the sending MTA of a message to resend it after a short time by issuing a temporary error during the SMTP connection. As of now, this solution is quite potent, as most spam is sent through zombies. Those worms contain their own SMTP engine, which is usually quite simple and only implements a subset of SMTP. Most of them are still unable to handle the temporary unavailable condition used in greylisting and therefore consider this condition as a fatal error and stop delivery. Greylisting has two major disadvantages: It slows down email communication and it is likely to be useless when those worms will implement better SMTP-engines, which is to be expected soon.

2.2. Modifying SMTP

The disadvantages of reactive anti-spam-methods as discussed above, initiated a discussion on fixing one of the supposed causes for spam: SMTP lacks authentication. So the key approach is to implement some kind of authentication and authorisation. Beside some side-effects seen on current methods, like breaking intended mail-forwarders, the real problem is to enforce the modified standards world-wide.

This is not only an organisational problem resulting from competing standards and companies trying to win their share of market by patenting their solutions, but also and mainly due to the broad, not centrally maintained base of billions of SMTP-clients and millions of servers in the Internet. Back at ARPANET times it was possible to change the standard to IP almost over night, but the Internet has grown. In [8] the amount of existing mail servers has been estimated to exceed 22.5 million machines world wide. By end of 2005 there were 938 million computers in use world wide [9], there were probably as many SMTP clients installed, as each PC implements at least one client.

There are still thousands of open relays out there, although open relays are deprecated and blacklisted since at least ten years. Considering this, any change to SMTP would need at least another ten years to be broadly available.

2.3. Preventing harvesters

Considering this, the search for new solutions has been opened. The probably most promising is to prevent spammers from collecting email addresses. Spammers collect email addresses both on places where they are publicly available (albeit not for the purpose of being harvested), the Web being the most prominent example, and places where the emails are not for public use. A popular example for the second case are worms and Trojans that get installed on computers and read local address books, emails or even all files to collect email addresses found there and send them to the spammer, so that he can spam to them.

There is an obvious solution to this: Have users install decent and safe operating systems, virus scanners and personal firewalls and protect their PCs with external firewalls and application level malware filters. Also, for all other non-public places where emails are stored, e.g. databases of companies' customers, appropriate protection is demanded to fulfil the appropriate privacy laws. Trading email addresses is not considered here, because those email addresses once were collected on one of the ways already described.

The other, public source of email addresses for spammers is the Internet, most notably the WWW and the usenet. There, they collect email addresses using spidering technology known from search engines. The programmes doing this job are called “harvesters”.

A harvester is a program that visits web pages, extracts all links in those pages and adds them its web pages to visit list. Besides extracting links its main goal is the collection of email addresses.

Again there are some ways how to handle them: One is to obfuscate email addresses, so they would not be recognised by harvesters. In [10] some different solutions are suggested, that are both compatible to any installed browser and barrier free, and proved their effectiveness in a still ongoing real world experiment. One of those solutions dynamically obfuscates email addresses published on the web [1], thereby solving the problem to modify or redo existing web pages to block out harvesters.

3. HTTP Tar pit

Our approach to bar harvesters from collecting mail addresses is to trap them in a tar pit. The basic concept is to create random web pages containing links to the same or other tar pits. This pollutes the list of web-pages-to-visit the harvester has, and keeps the harvester returning and finally staying in the tar pit. As soon as the harvester is caught, all of its resources are attracted to the tar pit, thereby preventing it to visit any other web pages and collect email addresses there.

3.1. HTTP Tar Pit Requirements

Setting up a functional and safe tar pit is not as easy as it might seem at first glance: First, "honest" spiders, such as GoogleBot, should not be trapped. Second: If the tar pit publishes links to itself, they need to be different. And last but not least the tar pit needs to make sure it is not hit by a denial of service condition if a harvester runs in circles through the site.

3.2. Do not catch good spiders

The first requirement, safe guarding the good, is easily implemented: Any decent spider should obey the robots.txt standard [11], [12]. Excluding any spider from the page would do. As of now, harvesters ignore robots.txt. From the harvester developers' point of view this is a logical decision to find even more email addresses.
Practical experiments proved this assumption to be correct. Both downloaded harvesters and those visiting a test tar pit ignored this standard.
If in the future harvesters would learn not to ignore the robots.txt-standard, this would be a positive result of using tar pits: Hiding email addresses on web pages would then become as easy as hiding those pages from robots with a robots.txt file.

3.3. Generate different links pointing to the same file

The next step was to generate new pages containing links to the tar pit using different URLs. The different URLs are necessary, because otherwise the harvester will detect that it has already visited the tar pit, and will leave it. With different URLS, this is not possible and the harvester is kept busy by pages generated only for the purpose of catching the harvester. In the test setup, filenames might have between 5 and 30 characters each and there is a choice of different filename extensions like ".htm", ".html", ".shtml" or ".shtm".

3.4. Avoid Denial of Service

The most difficult task is to avoid a denial of service condition: If the tar pit publishes 20 links per page, the harvester will add those links to its list of pages to visit. On each of those links visited, it will receive yet another 20 links. Within a short time, the harvester has some millions of links in his list all pointing to the tar pit.
If the harvester supports parallel spidering and is running on multiple machines, it might have enough bandwidth to pull the whole server down. If the server is only serving the tar pit, this does not matter – but if the tar pit is run on a server also used for other purposes, the "real" pages become undeliverable.
To avoid those problems, the instances of the tar pit running should be limited to a maximum. The determination of the exact number of instances is quite a tricky task, because it depends both on the tar pit server and on the unknown capabilities of harvesters. We have done it iteratively by testing several different numbers, and chose the one that fitted best.

3.5. Combining SMTP and HTTP tar pits

Although in real-world experiments the tar pit described above proved to be efficient, tests with off-the-shelf harvesters available in the web gave some hints on how to modify the tar pit to be even more effective.
Most harvesters implement some kind of progressmeter by listing the last email addresses found. The first tar pit implementation did not deliver any email addresses. Therefore, a human operator could realise that his harvester got caught by a tar pit. He could even blacklist the tar pit and inform other spammers of its existence.
To have harvesters stick longer to the tar pit, the tar pit should offer some email addresses to the harvester. But those addresses need to be existent: Random addresses under random domains might easily contain existing email addresses belonging to someone else who then will receive spam.
The other downside to random addresses is the so called bounce spam. This is spam sent to a non-existent address seeming to originate from another domain or email address than the one the spammer has. For each undeliverable spam message an error message is created and sent to the supposed sender's address, and, if it is also non-existent, to the postmaster of his domain.
Considering this, email addresses published by the tar pit should be existent and a mail server should accept messages to them. To achieve this, the authors suggested in [2] to use a SMTP tar pit as pseudo-MTA for the HTTP tar pit.

3.6. SMTP tar pit

SMTP tar pits usually accept mails, but they answer incoming SMTP connections very slowly and thereby waste the time of the sender. There are two basic concepts in slowing down the connection: One is to slow down the connection on the TCP/IP-Level by using minimum framesizes, sending each frame on its own etc. [13], the other, more common, is to use application level slow downs. To do so, SMTP-continuation-lines [14] are used: Each request is answered with dozens of response lines. Those lines are usually sent with short delays in between, adding an extra slowdown.
Doing so, bulk mailers should be slowed down on each connection they have to a tar pit. But set up on their own, SMTP tar pits are quite ineffective: They only slow down one connection to a certain mail server at a time, which usually has almost no impact, as one server is able to accept many mails during one connection and most bulk mailers are capable of connecting to many mail servers in parallel [8].

3.7. Real world experiment

In the authors' real world experiment's configuration, a slightly modified version of smtarpit [15] has been used. This programme uses an application level slow-down technique with continuation lines.
The combined tar pit proved to be 20 times more effective than a standalone HTTP tar pit [2]. It was able to attract harvesters for several weeks. At least one harvester

continually connected to the tar pit until the machine running the harvester was disconnected from the internet by its provider due to spamming complaints.

4. Identifying harvesters with the combined tar pit

Thanks to previous real world experiments with HTTP and combined tar pits, our tar pits are heavily linked from many web pages in the Internet. They therefore attract harvesters that follow links on web pages. Humans that accidentally followed a link to the tar pit will soon notice that the page they came across is not intended for human visitors. Harvesters by contrast will stay in the tar pit, as field experiments proved.

This makes the tar pit a useful mean to tell apart humans from machines: As soon as a visitor stays for more than a few visits in the tar pit, it is very likely to be a machine.

If the visitor is a machine, it did not obey the robots.txt standard [11], [12], that protects good spiders from being trapped in the tar pit. Therefore, the visiting machine is likely to be a harvester.

To understand harvesters' behaviour, the log files of the authors' tar pits were analysed and evaluated. As expected, there were no time patterns to be identified – harvesters seem to wait for a random time between two visits and they also have different length lists of pages to visit that also influence when they will visit the next link to the tar pit in their list.

Accidental human visitors by contrast usually visit the tar pit for at most 15 minutes and have then left it. Those that stayed so long seemed to have analysed the tar pit's behaviour. This at least is made plausible by looking at entry points harvesters used when visiting the tar pit: Some webmasters who linked the tar pit understood well how it worked and crafted their own, specific links to it.

As harvesters' timing is unpredictable, but humans' is, the first piece of information to identify a harvester is that it is not visiting the tar pit for only fifteen minutes. Humans should not be blocked from visiting other web sites, but harvesters should.

If our assumption from the log file is true, that human visitors who click on more than one link in the tar pit are likely to be people who try to understand how the tar pit works, it might be safe to also assume that those people will understand why their access to other web pages has been blocked for a certain time.

So the real problem are “one link visitors” who by accident came across the harvester trap. To avoid them to be banned from web page access for too long, the ban is imposed depending of the amount of visits during a certain time period.

Another important piece of information derived from log file analysis was that a non negligible fraction of harvesters are operated from dynamic dial-in IPs, i.e. their IP address is changing at least every 24 hours. Therefore, it should be avoided to block IP addresses for more than 24 hours, if, after 24 hours after the first harvesting report from this address has occurred, harvesting suddenly ceased.

5. Blocking Harvesters

To block harvesters, the Apache[1] output filter presented at last years CNIS [1] that provided a solution to dynamically obfuscate email addresses, has been enhanced to look up in a database of known harvesters whether the IP address of the machine requesting a web page from the Apache server is listed in the database of known harvesters. This database is populated with data from the combined SMTP HTTP tar pit and regularly maintained in a way, that, if an IP is found in the database, the output filter knows, that this IP is to be blocked.

If a blocked IP has been identified, a preconfigured web page is delivered informing the visitor that his IP was blocked due to harvesting and giving him advice on what to do if he feels to have received this message in error. For obvious reasons we recommend to offer only obfuscated email addresses on this page as it will mostly be seen by harvesters and not humans. However, it is a good suggestion to politely explain why access has been denied, just in case the visitor uses a dynamic IP still blocked from its previous owner.

6. Test setup

6.1. Tar pits

To test the concept, a combined tar pit has been installed on three servers in the web. Each of those machines served at least three domains, where on each machine one domain was registered among the generic and country code top levels and the others were DynDNS[2] domains. Under each domain, subdomains could randomly be choosen – this was supported by both the webserver and the DNS configuration. This was done to allow a high degree of obfuscation and to trick harvesters that only allow a certain amount of links for each virtual server.

One of those virtual servers has been configured to be the mail exchange (MX) for all domains used for tar pitting. This seems reasonable, as in previous tests, the SMTP tar pit had much less requests to handle than the HTTP tar pit.

We decided on using three tar pits in three different networks to both have a bigger chance to catch harvesters and to reduce the risk of network failure.

6.2. Storing Harvesters' IPs

In our test setup we decided to keep the installation process as simple and therefore stable as possible. Thus, we decided to use a MySQL[3] database to store the IPs and access times on the tar pit. MySQL already implements the necessary serialisation required to handle parallel requests from different servers, it has an acceptable level of access control for our test setup, it is easily accessible both from Perl we used for the Apache output filter and PHP used for the tar pit and it offers a well documented interface to analyse and modify data stored in the database.

1 http://httpd.apache.org

2 http://www.dyndns.com Dynamic DNS allows a domain name such as example.org to be bound to frequently changing IP addresses. It also offers DNS services for free.

3 http://www.mysql.com

To satisfy our safety requirements, we installed MySQL on a seperate, hardened server guarded by some firewalls and used reverse SSH-tunnels from this server to the tar pit servers to encrypt and secure the connection to the database. By using a reverse SSH tunnel the database server was to initiate the connections and not the exposed tar pits, that are likely to become
the target of an attack when a spammer detects that one of the tar pit machines carries a tar pit that caught his harvester, and thinks of revenge. We restricted the access rights for the database user, which the tar pits use to connect to the database to a bare minimum: it was only allowed to perform insert statements on one table.

6.3. Imposing bans on IPs

To analyse tar pit access, we decided to use another Perl programme. This programme would determine which IPs have been banned long enough and delete those outdated IPs from the database.
We decided to keep the decision logic out of the Apache output filter to both keep its performance as high as possible and to have an easy opportunity to modify the logic according to our findings during testing. If we had implemented the logic into the output filter , any modification would have required to restart Apache, as mod_perl[1] precompiles output modules upon the webserver's start and would not notice changes made to the module while the server is running.
We found that the ban time should increase with the square of the number of visits. We decided a minimum ban time of fifteen minutes. Formula 1 describes the basic ban time algorithm we used.

$$t_{ban}(v) = v^2 \cdot 900\, sec$$

Formula 1: ban time (t_{ban}) in seconds depending on visit count (v)

The reason for using a quadratic growth of ban time was to reflect the increasing likelihood for a harvester by one IPs visit count. We choose the fifteen minute minimum ban by looking at our tar pits log files that indicated that most often harvesters would return within the first ten minutes of their first visit to the tar pit. We also took into account that accidental human visitors should not be blocked for too long. To compare and optimise ban time formulas is part of our still ongoing testing.
As described above, some harvesters use dial-in accounts with dynamic IP addresses that change after 24 hours. To avoid banning the next person who would inadvertly receive this IP, if a IP address has been reported for harvesting within the last 25 hours and no harvesting has occurred during the last hour, this IP is removed from the harvester list. This is only done once after 25 hours of the first harvesting report of a IP in the database.
As soon as the ban has expired, all entries from the database will be removed to allow the Apache output filter to just look up an IP in the database, allowing maximum search performance. So, any IP listed in the database is a harvester.

1 http://perl.apache.org

6.4. Apache output filter

We enhanced the Apache output filter presented in [1] to test if the remote IP is listed in our database of known harvesting hosts. If the IP is found, a special web page is sent and regular page data is omitted.
Although Apache allows filtering in almost any stage of its process handling [16], we chose not to create an additional filter residing for example in the input chain, but to stick to our output filter, as we wanted to use its email obfuscation functionality if the web page is being delivered to a not banned IP. And we wanted to keep things simple for testing purposes.
However, using an output filter to only deliver a static page forces the web server to do all processing of the web page before it is possibly thrown away by the filter. In a non-testing environment, it might be useful to use an input filter instead to reduce the workload on the server.
It is worth noticing that the position where the filter is implemented in the processing chain is only a question of performance but not on whether the concept is effective or not. In our test setup, we were able to accept small performance trade offs, as we supposed that most visits to our test page will be real people and harvester visits would make up for less than 1 percent of all visits. Furthermore, our test web site consisted only of static pages, so the webserver would not require to much processing power to first prepare each page.

6.5. Efficiency of this setup

The test setup was running for ten weeks. The website had an average of 899 hits per day.
In terms of delivery perfomance of the web page, our test setup is almost as fast as the output filter solution we presented in [1]. This additional filter's impact on the webserver's overall performance is therefore small and acceptable compared to its promised spam reduction.
Our test web server's log files documented, that some visitors were already denied access to the web site. On average 2% of all visits to the web page were blocked due to the remote IP being blacklisted. By comparing IPs, we did not find any IP that has been blocked to have later or earlier accessed the web page. This indicates that the choosen block time algorithm seems to fit.
Logfile analysis did not show any blocked access by legitimate users or spiders, like GoogleBot. This indicates, that the harvester detection by using our combined SMTP- and HTTP-tar pits is working efficiently.
It is however possible that harvesters visited the web page, that were never caught by our tar pits. This is likely because there were still only a few tar pits installed in the world wide web.
Neither our test email accounts nor real user accounts published on this page did receive any spam.
However, by blocking at least some of the harvesters – even in a simple test setup – and by not blocking legitimate users, from our point of view, using this dynamic approach to prevent spammers from collecting email addresses from web pages is efficient. A larger scale test implementation is planned and will be used to further verify our findings.

7. Possible enhancements

The basic setup we used for testing offers some possibilities for further enhancement. As the test setup does not keep record on harvester IPs, harvesters could unlock their IP if they stopped harvesting every 24 hours for one hour. In this case, they would only be locked for one day.

However this would require spammers to know how a harvester is identified. As the setup we provide in this paper is highly modular and offers many ways to configure the length of the time ban imposed, each user could easily decide to alter our suggestions and thereby further obfuscating the decision criteria.

Another improvement would be an automatic report on the existence of a harvester to the provider of the IP address harvesting has been reported from with a view to shut down the malicious machine. This would increase the economical and legal risk a spammer takes and expose him at an earlier stage than sending spam. [17] e.g. reports on identifying harvester's users by feeding harvesters specially created, unique email addresses and thereby documented that harvesters are often run from within the spamming company, as spammers still do not see any need to hide their harvesting activity.

8. Conclusions

Trying to prevent harvesters to collect email addresses of possible spam victims has already been described to be effective [1]. The usage of combined SMTP and HTTP tar pits to trap harvesters is also powerful [2]. By using a combined SMTP and HTTP tar pit to identify harvesters, it is possible to lock out harvesters from a certain web page and thereby prevent them to collect email addresses and links to other web pages there. This combination helps protecting the Internet from harvesting activity and safeguards the mail addresses published on this web page.

Further research is aimed at establishing a secure and covert communication channel between tar pits to safely exchange data on their visitors, as the reverse ssh tunnel used in the described test setup is only suitable for testing purposes. We also plan to conduct research into a secure and covert information interchange between tar pits to give them the possibility to dynamically link to each other, which would help to create a broader and dynamically changing base of tar pits, which in turn would increase the probability of harvesters being caught in them and therefore help identifying harvesters earlier.

References

[1] Eggendorfer, Tobias; Keller, Jörg, Preventing Spam by Dynamically Obfuscating Email-Addresses, Proceedings of CNIS 2005, Phoenix, 2005

[2] Eggendorfer, Tobias; Keller, Jörg, Combining SMTP and HTTP tar pits to proactively reduce spam, Proceedings of SAM 2006, Las Vegas, Nevada, 2006

[3] Spam Daily News, Gmail servers blacklisted by SpamCop again, http://www.spamdailynews.com/publish/GMail_servers_blacklisted_by_SpamCop_again.asp, 2006

[4] McWilliams, Brian, SpamCop blocking some Gmail servers, http://spamkings.oreilly.com/archives/2006/01/, 2006

[5] McWilliams, Brian, AOL lands on spam blacklist, http://spamkings.oreilly.com/archives/2005/04/aol_lands_on_sp.html, 2005

[6] Gansterer, Wilfried et. al., Anti-spam methods - state of the art, Institute of Distributed and Multimedia Systems, University of Vienna, 2005

[7] Donelli, Giovanni, Email Interferometry, Proceedings of Spam Conference 2006, Cambridge, MA, 2006

[8] Eggendorfer, Tobias, Comparing SMTP and HTTP tar pits in their efficiency as an anti-spam-measure, Proceedings of Spam Conference 2006, Cambridge, MA, 2006

[9] ETForecasts, Computers in use. Forecast by Country, http://www.etforecasts.com/products/ES_cinusev2.htm, 2005

[10] Eggendorfer, Tobias, Methoden der präventiven Spambekämpfung im Internet (in German: Methods of preventive spam abatement in the Internet), Masterthesis at Fernuniversität in Hagen, München, Hagen, 2005

[11] Hemenway, Kevin, Calishain, Tara, Spidering Hacks. 100 Industrial-Strength Tips & Tools, O'Reilly, Sebastopol, 2003

[12] W3C, W3C Recommendations. Appendix B: Performance, Implementation and Design, http://w3.org/TR/REC-html40/appendix/notes.html

[13] Li, Kang et al., Resisting Spam Delivery by TCP Damping in Proceedings of CEAS 2004, Mountain View, CA, 2004

[14] Klensin, John (Editor), RFC2821: Simple Mail Transfer Protocol, http://www.ietf.org/rfc/rfc2821.txt, 2001

[15] Grosse, Paul, SMTarPit v0.6.0, http://www.fresh.files2.serveftp.net/smtarpit/index.html, 2006

[16] Stein, Lincoln D., MacEachern, Doug, Writing Apache Module with Perl and C, O'Reilly, Sebastopol, 1999

[17] Rehbein, Daniel A., Adressensammler identifizieren - Ein Beispiel (in German: Identify address collectors – an example), http://spamfang.rehbein.net