A proposal for an efficient way to prevent spam by analysing SMTP and HTTP tar pits towards their efficiency in fighting spam and combining them

TOBIAS EGGENDORFER
ITIS e. V. Institut für Technik Intelligenter Systeme
Universität der Bundeswehr Neubiberg
Werner-Heisenberg-Weg 39
D-85579 Neubiberg
GERMANY
tobias.eggendorfer@unibw.de http://www.unibw.de/Tobias.Eggendorfer/

Abstract: - Unsolicited commercial email (UCE, spam) is the biggest threat to email communication. As of January 2005 more than 93% of all emails are spam. There are different attempts to fight spammers, one of those is to delay the collection of email addresses by setting up HTTP tar pits, another one is to delay the transmission of spam by setting up SMTP tar pits. This paper compares those two methods with a view to their efficiency in preventive spam fighting. It then analyses how SMTP and HTTP tar pits could be combined to increase their efficiency and provides results from a real world test.

Keywords: Spam, SMTP, HTTP, tar pit, proactive anti-spam-measures, combined tar pit

1 Introduction

According to different statistics, more than 80% of all emails sent are spam: In 2004, [1] reported 82% of all U.S. Emails to be spam. According to [2], in January 2005 93% of all emails in any user's inbox were spam.

1.1 Reactive methods

Most methods to reduce the amount of UCE in any user's inbox rely on some kind of filtering. One of the first approaches was blacklisting, i.e. each incoming request's IP-address is tested against a list of known spamming hosts. Although, when invented back in the late 1990s, it supported the demand of switching off so called open-relays, blacklisting often has heavy side-effects [3]: Almost all big email-providers have already been blacklisted on at least some of the widely available blacklists [4], [5].

The other common approach is to apply a content-filter to the header and / or the body of a mail message. This kind of filtering mainly is based on some kind of “bad-word-list” and other spam symptoms like references to external images and unusual encoding of the message body. Those filters need to be adjusted to individual needs and require maintenance: Spammers are reported to register mail accounts with online services known to have spamfiltering and to test their spam against their filters. This leads to a permanent “one-step-behind”-situation for filters, however advanced content-filtering becomes.

Another reactive way to reduce spam is greylisting [6], i.e. forcing the sending Mail Transfer Agent (MTA) to resend the mail after a few seconds. As of now, this solution is quite potent, because nowadays most spam is sent through so called zombies, usually Windows-PCs infected with some worms and backdoors. Those worms contain their own, usually quite simple SMTP-engine. Most of them are still unable to handle the temporary unavailable condition used in greylisting and therefore consider this condition as fatal error and stop delivery. Greylisting has two major disadvantages: It slows email communication down and it is likely to be useless as soon as those worms will implement better SMTP-engines, which is to be expected soon.

1.2 Modifying SMTP

Another approach is to fix the supposed real cause for spam: SMTP lacks authentication. So one of the key approaches is to implement some kind of authentication and authorisation. Beside side-effects seen on current implementations, like breaking 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. There are still 225.000 open-relays1 out there, although they are deprecated and blacklisted since at least ten years. Considering this, any change to SMTP is likely to take at least another ten years to be broadly available.

1.3 Preventing harvesters

Considering this, new solutions are required. The probably most promising is to prevent spammers from collecting email-addresses, because spammers currently only use two relevant ways to collect addresses: One is by installing worms and trojans on computers and have them read local addressbooks, emails or even all files, collect email-addresses found there and spam to them. There is an obvious and theoretically simple solution to this: Have users install safe operating systems, virus scanners and personal firewalls and protect their PCs with external firewalls and

1 See: http://www.ordb.org/statistics/relaycount/
application-level-malware-filters.

The other email-address-source for spammers is the internet, most notably the web and the usenet. There, they collect email-addresses using spidering technology known from search engines. The programmes doing this job are called "harvesters".  

Again there are some ways how to handle them: One is to obfuscate email-addresses, so they would not be recognised by harvesters. In [7] I suggested some different solutions, that are both compatible to any installed browser and barrier free, and proved their effectiveness in an ongoing real world experiment [8]. Later, in [9], an automated solution to dynamically obfuscate email-addresses published on the web was proposed, thereby solving the problem to modify or redo existing webpages.

2 HTTP tar pit

Another way to bar harvesters from collecting mail-addresses is to trap them in a tar pit. The basic concept is to create random webpages containing random links on the tar pit. This pollutes the list of webpages 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 it's resources are attracted to the tar pit, thereby preventing it to collect email-addresses on other webpages.

In [8], I gave some detailed explanations on how a HTTP tar pit needs to be set up. The major requirements were to protect "honest" spiders such as GoogleBot from being caught in the tar pit, to avoid the tar pit's server to break down under some kind of denial of service attack by catching possible multi-process and multi-homed harvesters and last, but not least, to be as invisible as possible. In [8] I provided details on how to implement a HTTP tar pit fulfilling those requirements. It's source code has been published in the proceedings.

The tar pit has meanwhile been enhanced a little to run parallel on different servers using different domains with a view to obfuscate it even more. It also changed its behaviour from constantly publishing 20 different links to itself to presenting a random link count, which results on an average of 15 links per generated page. Also the layout and "wording" of the tar pit has been modified to make it look even more like a real webpage.

3 SMTP tar pit

The basic concept of SMTP tar pits is to replace or proxy an email server and accepting email in his place. The tar pit attempts to bind the sender as long as possible to bar the sender from sending spam to other addresses on other email servers.

There are two main kinds of SMTP tar pits: Those adding tar pit functionality to an existing SMTP-server while serving real recipients, and those configured for a domain only set up to thwart spammers.

SMTP tar pits of the first kind are for example OpenBSD's spamd [10], the Sticky Proxy described in [11] and the tar pit wrapper by Lutz Donnerhacke [12]. Their main advantage is the ability to use them with an up and running mailserver. Slowdowns in regular mail operations are accepted with the main argument that regular users would accept a short delay if they receive less spam. This is basically the same argumentation as with greylisting.

Sticky Proxy uses spamassassin [13] to determine if an incoming message is spam. Spamassassin is highly configurable and might be optimised to the needs of a specific user or mail server, reducing the risk of false positives and thereby reducing the risk of slowing down non-spam MTAs.

Donnerhacke's tar pit wrapper uses a fixed list of spam IPs which needs to be updated manually. This is usually not a good idea, as spammers now use so called bot nets, systems infected with some worms and backdoors, to send spam. The IPs of those bots change constantly, because most of those zombies have a dial-in connection with dynamic IPs. Thus a hard coded lists of IPs is ineffective.

OpenBSD's spamd is implemented on the application level and answers every request by stuttering each second one character. It is somewhat self configuring by using greylisting to distinguish between good and spam mail servers. Based on the greylisting results, both a black and a white list are being populated [10].

Sticky Proxy delays traffic on the network layer. Its authors see the network layer approach as the preferable solution, because according to their logic, after having delivered a message, the spammer could disconnect from the tar pit. Certainly, a spammer would disconnect after his message has been delivered, but application layer delays are not limited to the notification of acceptance of a message.

The other kind of tar pit replaces the MTA for an entire (sub-)domain. Email messages delivered to it would be discarded and would not be forwarded to an existing user.

Those tar pits are usually advertised by publishing mail addresses on prepared web pages. Examples for those systems are Daniel Rehbein's "Mailvernichter" (German: mailtrasher) [14] and smtpart by Paul Grosse [15]. Both of them – and also Donnerhacke's tar pit wrapper mentioned above – use SMTP's continuation lines to delay the mail sending process.

Continuation lines are used to give further details on the SMTP status returned [16]. By sending continuation lines very slowly, the reply is delayed. Continuation lines are identified by a dash after the status code.

smtpart and mailtrasher require a MX record pointing to their server. Emails delivered to those systems will be trashed, as mailtrasher's name already suggests.

smtpart rejects email on each delivery attempt in the hope the bulkmailer would retry sending later. This concept is dangerous: If spam is not sent directly but through an open proxy, this behaviour would generate "mail delivery failure"-mails to the originating address which is almost never the email address of the spammer. In fact, abusing a real, existing address of someone other than the sender of a spam message is common practice and meanwhile a major source of annoyance to users [17]. The author of this paper...
receives an average of 450 spam bounces a day due to spammers abusing some of his email addresses as sender.

4 SMTP and HTTP tar pit compared

4.1 Effectiveness of a HTTP tar pit

A HTTP tar pit tries to intercept spammers' harvesters while collecting email addresses. It is a preventive approach: If spammers are unable to collect email addresses, they are unable to spam.

In [8] I presented results of a real world tar pit and how spammers accepted it: It was able to attract some harvesters to spend 20,000 visits to different pages on the tar pit resulting in each harvester being locked into the tar pit for one or more days and thus being stopped from collecting email addresses.

It is very likely that the harvesters caught in the tar pit were later stopped by their operators because most harvesters do indicate their progress by listing the email addresses found. The tested HTTP tar pit did not list any email addresses, a fact the operator of the harvester would then realise.

4.1.1 Theoretical approach

The HTTP tar pit offers dozens of links back to itself – in the chosen implementation, an average of 15 links back to the tar pit using different domain names is offered to the harvester. The amount of back links the tar pit presents is randomly chosen between a minimum \( l_{\text{min}} \) and a maximum \( l_{\text{max}} \). This is done to obfuscate the tar pit and to reduce the risk it is being recognised by the spammer.

\[
I_{\text{tar pit}} = \left( \frac{l_{\text{max}} - l_{\text{min}}}{2} \right)^r
\]

**Formula 1: Total link count after \( r \) rounds**

Harvesters usually add each link they find to a list of links to visit and starting to contaminate it. For obvious reasons its efficiency increases with the amount of new links published on a web page and probably over estimates the number, as only a small part of the web has been crawled.

To become effective, the HTTP tar pit only needs to be linked from one web page, thereby being introduced into the harvester's list of pages to visit and starting to contaminate it. For obvious reasons its efficiency increases with the amount of links pointing to it.

To become effective, the HTTP tar pit only needs to be linked from one web page, thereby being introduced into the harvester's list of pages to visit and starting to contaminate it. For obvious reasons its efficiency increases with the amount of links pointing to it.

4.1.2 Results

Taking the increasing degree of contamination as a parameter for the efficiency of the tar pit, the tar pit should be effective. This is in good accordance with the results from real world experiments.

The HTTP tar pit is also capable of trapping a multi-threaded or even distributed harvester, as long as the list of pages to visit is shared among the harvester processes. A prerequisite that should be good practice to avoid visiting links more than once by any parallel instance. With a growing degree of contamination of the list of pages to visit, more and more processes of a parallel harvester will be trapped, although, to my knowledge, distributed harvesters do not exist yet.

To become effective, the HTTP tar pit only needs to be linked from one web page, thereby being introduced into the harvester's list of pages to visit and starting to contaminate it. For obvious reasons its efficiency increases with the amount of links pointing to it.

Due to the contamination of the harvester's list, the HTTP tar pit is considered an effective preventive anti spam measure.

\[
4 \text{ SMTP and HTTP tar pit compared}
\]
4.2 Effectiveness of a SMTP tar pit

A SMTP tar pit goes into effect later in the process of spamming: It tries to slow down a spam run. To do so, spammers need to have email addresses pointing to a tar pit. With tar pits set up in front of a real mail server, this is no problem because each account would receive spam. Standalone tar pits like smarpit and mailtrasher by contrast need mail addresses pointing to them published. Users those tar pits set up web pages and post into the usenet to advertise their addresses to harvesters and worms.

4.2.1 SMTP tar pit protecting the local MTA

If the SMTP tar pit is only set up to protect the local MTA, it has to be considered effective if less spam is delivered. Using the OpenBSD spamd as a representative example, a positive effect on the reduction of spam is visible. Due to its greylisting system, which is still effective, a certain percentage of spam is blocked. This effect however is not due to the tar pit, but only to its specific implementation.

But according to [10] most bulk mailers now have a SMTP tar pit detection and use very terse timeouts to avoid being trapped. This is realistic because SMTP tar pits are quite common. If bulk mailers are trapped in any SMTP tar pit protecting a MTA and they have their timeout set as described, they would not arrive to deliver their message. Therefore, the message is not sent to the MTA, thus reducing the amount of spam messages the mail server has to deal with. From this point of view, a SMTP tar pit is efficient.

4.2.2 Delaying the spam run

A protecting SMTP tar pit like OpenBSD's spamd, the teergrubbing wrapper or Sticky Proxy have no big impact on bulk mailers because they are set up to protect and not to fight. But mailtrasher and smarpit are intended to form a real obstacle to a bulk mailer. So their effectiveness is to be measured by the delay they mean to a spam run. Whether they are effective or not depends only on the implementation of the bulk mailer:

The simplest bulk mailer would try to deliver spam message after spam message to a MTA and would not use a timeouts. This – unfortunately rare – typus would suffer from a real slowdown.

If timeouts are implemented, the bulk mailer would also be delayed, but the effect would be less significant, because of the timeout interrupting the tar pit to early. Most SMTP engines found in modern worms use timeouts.

Modern bulk mailers span either multiple processes or are multithreaded. They try to deliver their spam to multiple mail servers at a time, each process connecting to one server. Only one connection at a time is slowed down.

Most authors (e.g. [18]) of those tar pits explain that their tar pit is effective, because it closes the spammer's ports. But each SMTP connection allows delivery of multiple messages to multiple recipients. So, only one connection to any mail server needs to be maintained by the bulk mailer. As one connection requires only one unprivileged port out of ~64,000 available ports, the effect is not relevant.

In ideal circumstances one bulk mailer might connect to 64,000 servers at a time, if the CPU and memory of it's host are capable enough. SMTP is neither a very complicated protocol nor does it require a lot of computing power nor a lot of memory. In fact, a sending process is waiting most of the time for the network. Nota bene: The tar pitted process idles even more - waiting for the tar pit's answer.

The intended impact to block the bulk mailer is not achieved if the bulk mailer is multithreaded.

The real bottleneck in communication is probably the bandwidth the spammer has. So, if the tar pit would consume a lot of bandwith, it might slow down the spam run. But the SMTP tar pit does not consume significantly more bandwidth than any other SMTP server, the continuation lines' overhead or even the lower level protocol overhead generated by TCP damping mechanisms is too small. On the bandwidth side, there is also no relevant effect.

To really slow down spammers, a lot of SMTP tar pit implementations would be needed internet wide: The more SMTP tar pits exist, the higher the probability of a spammer's bulkmailer attaching to more than one. To really delay connections at least 25% percent of all possible outgoing connections of a bulkmailer should be trapped. According to [19] in 2002 less then 1% of all mail servers were open relays. The percentage has decreased since. ORDB.org counted approximately 225,000 open relays in February 2006. If less than 1% of all mail servers are open relays, there should be a lot more than 22.5 million mail servers in the internet. Those numbers are estimated, probably underestimated. To trap 25% of all outgoing connections a bulkmailer has, roughly 25% of all mail servers should be a tar pit. As tar pits need to be added to the existing base of mail servers, 7.5 million SMTP tar pits should be installed to have an impact on spammers.

Due to the fact that more and more bulk mailers are tar pit aware [10] and would interrupt a connection to a SMTP tar pit, the installation of this massive amount of SMTP tar pits would become useless within a short time.

4.2.3 Results

Based on this analysis of a SMTP tar pit's behaviour the conclusion is that a SMTP tar pit is inefficient in delaying a spam run, but it might support in protecting one's mail server against spam.

5 SMTP and HTTP tar pits compared

Although SMTP tar pit's are promoted as a capable mean of fighting spam by delaying a spam run and thereby protecting others from receiving spam, their main positive effects are in protecting a single mail server from spam by turning bulk mailers down due to their timeouts.

HTTP tar pits by contrast are an efficient preventive anti spam measure, because they contaminate a harvester's list of links to visit thus preventing it to spider other web pages for email addresses. To stay as efficient as they are, there is still a lot of effort needed to obfuscate HTTP tar pits more efficiently.

1 See: http://www.orb.org/statistics/relaycount/
6 Combining SMTP and HTTP-tar pits

Although in real-world experiments the basic HTTP tar pit has proven its efficiency, tests with off-the-shelf harvesters 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. If no email addresses have been found for a long time, the harvester's operator could be alerted and realize his harvester has been tar pitted. He then would then stop his harvester and blacklist the tar pit.

To have harvesters stick longer to the tar pit, the tar pit should offer some email addresses to the harvester. Those addresses should 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 email address. 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. However, setting up a mail server to only accept spam would thwart the efforts in setting up the http tar pit.

Instead, it would be nice to again trap spammers in a tar pit when they try to deliver mail to the addresses collected from the webpage. To achieve this, a SMTP tar pit would be needed. There are different kinds of SMTP tar pits available in the net, as quite a lot of anti-spammers use them, although none of them has been used to my knowledge in combination with a HTTP tar pit. Teamed up with a HTTP tar pit, a SMTP tar pit could serve as MTA for the addresses published by the HTTP tar pit. Although the SMTP tar pit has virtually no effect on the performance of a spammer's bulk mailer, it would turn the HTTP tar pit even more effective by adding an additional level of invisibility to it.

6.1 Implementation

6.1.1 Modification of the HTTP tar pit

To combine the HTTP tar pit and a SMTP tar pit, the existing HTTP tar pit has been modified to deliver email addresses under certain domains. As those generated email addresses are shown to the harvester's operator, they should look like real addresses, made up from a first and a family name. To achieve this, a list of first and family names has been collected from the web. From each of those lists, 150 names were randomly selected. While running, the tar pit generates an email address by randomly selecting a first and a family name, joining them with a dot and appending one out of the configured domain names.

6.1.2 Selection and adaption of an existing SMTP tar pit

As SMTP tar pits are commonly used in the internet, available tar pits were evaluated. One of the key requirements was the ability to easily modify the source code to later store collected emails in a database. Other requirements included, but were not limited to, security considerations and compatibility to the existing platform without the need to install too much additional software.

The most promising candidates were [14] and [15]. [14] required the installation of Java on the tar pit and had only very few configuration options. [15] instead is written in Perl, for which an interpreter has already been installed on the tar pit server. And perl reduces the risk of memory leaks and buffer overflows by design. The tar pit is very configurable, offering almost all options needed. The well-documented source code has than been reviewed for potential security problems, where only a minor enhancement was to be made.

As described above, by default, smarthp tar pits refuses to accept mail with a 500-SMTP-error-code after having wasted the bulkmailer's time. This refusal might finally lead to bounce spam. To avoid this, the tar pit has been modified to either return a "temporarily unavailable" status or accept the message, with a probability of acceptance of 70%. So it is very likely that after some attempts a message could be delivered – after only ten attempts, the probability reaches 99,9994%. The standard configuration for sendmail e.g. would retry delivery for five days every thirty minutes resulting in a non-delivery probability of 3.2 * 10^-126. This is considered to be acceptable.

6.2 Real world experiment

To test the efficiency of the combined HTTP and SMTP tar pit, hidden links to the tar pit where published in the internet in cooperation with some well frequented web pages.

The classic HTTP tar pit proved already its efficiency: It kept some harvesters returning for more than 20,000 visits, blocking each of them up for one or more days. It is very likely that their operator interrupted them because no email addresses were delivered.

With the combined tar pit, this shortcoming has been resolved: four weeks after the installation of the tar pit, one harvester stayed for seven days and more than 401'000 visits. This is a daily average of 57'285 visits. This harvester was run on a rented server at a Germany based web service provider, who confirmed by phone that he cancelled the hosting contract due to too many spam complaints originating from this server's IP on the same day the harvesting stopped. By analysing the tar pit's log file by IP addresses, the next two of top three visiting IPs were identified: The second requested more than 94'400 pages within 24 hours, the third 32'197 pages. Both of them used dynamic IP addresses, i.e. IPs that will change after a maximum of 24 hours.

Looking at the top visiting harvesters' user agent, i.e. the identification string a browser sends, I found one of them to be unique and therefore supposed it to be used only by one installation of the harvester. It might however be that this user agent is specific for this harvester and not for one specific installation of it. So those numbers are less accurate than those based on the IP, but, due to the usage of dynamic IPs we had to identify something else to deter-
mine returning harvesters after they changed IP. Based on its user agent, the harvester that counted for 94'400 visits within 24 hours from one IP is supposed to have spent a total of more than 537'000 visits in 20 days, this is a daily average of 26'850 visits.

Those results went far beyond the expectations in a combined SMTP and HTTP tar pit efficiency’s increase: 20 times as many visits to the tar pit than on a standalone HTTP tar pit were counted. The SMTP tar pit also attracted some spammers: Within four weeks, we counted approximately 3000 connection attempts. Compared to the HTTP tar pit’s numbers, this looks rather small, but it is in good accordance to our theories: During one SMTP session, all messages from one client to this server might be transferred. Therefore much fewer connections are required for SMTP than for HTTP. The theory concerning tar pit aware bulkmailers [10][20] was also confirmed: Most connections lasted for less than a second. Considering those two facts, the SMTP tar pit performed well and served its purpose to further obfuscate the HTTP tar pit.

7 Conclusion

Combining a HTTP tar pit with a SMTP tar pit increases the efficiency of the tar pit because the operator of a harvester might watch his harvester sucessfully collecting email addresses. This increased the time the harvester stayed in the tar pit.

Due to the huge amount of tar pitted email addresses the spammer collected, a welcome side effect is the delay of a later spam run by the SMTP tar pit.

The combination of both tar pits had a clear impact on the efficiency of the HTTP tar pit. The SMTP tar pit does not serve as first line of defense, as most other SMTP tar pits currently do, but supports the HTTP tar pit. Thereby the disadvantages inherent to a standalone SMTP tar pit have been overcome.

Furthermore, both tar pits were programmed with a focus on security, therefore both should not increase the risk of break ins into the tar pit server. A well documented configuration and installation process offer the possibility to easily implement the combined tar pit by any system administrator.

Current research is into further obfuscation techniques for the tar pit.

References
[11] Li, Kang et al., Resisting Spam Delivery by TCP Damping in, , 8, University of Georgia, Athens, GA, o. A.