Empirical Analysis of Denial of Service Attack Against SMTP Servers

Boldizsár Bencsáth, Miklós Aurél Rónai
Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics, Hungary
{*bencsath, ronai*}*@crysys.hu*

ABSTRACT

In this paper we show that the performance of the generic SMTP servers are more limited than we previously thought. We implemented a environment to test SMTP server performance focusing on Denial of Service (DoS) attacks. Our measurements show that a standard SMTP server can be easily overloaded by sending simple email messages and the overload can occur without consuming all network bandwidth. Our measurements also show that the usage of content filtering applications can harm the performance so much that the server become even more vulnerable to DoS attacks. In the paper we describe the problems of performance measurements in SMTP environment and we also give a detailed background about the performed measurements.

KEYWORDS: E-mail, SMTP, DoS attack, bechmark, throughput

1 INTRODUCTION

Denial of Service (DoS) attack is a hot topic in research, but no single solution can be developed to solve the general problem of DoS. The internet mail delivery service, which is provided by Mail Transfer Agents (MTA) that use the Simple Mail Transfer Protocol (SMTP) for exchanging e-mails, is often a target of intended or unintended DoS attacks. In this paper we investigate the behavior of various MTA server implementations under heavy load that can lead to denial of service. We will show that the SMTP servers are highly vulnerable to such attacks, as only a small number of emails (small bandwidth) is enough to fully load an SMTP server that can lead to a DoS condition. We do not intend to give information on countermeasures. There are a number of possible solutions proposed against DoS situations, e.g we proposed to use traffic level measurements in a DoS front-end ([1]). In the current paper our goal is to help the understanding why SMTP servers are so much affected with DoS related problems, and why such countermeasures are important. We also do not claim that measuring throughput of a server is something really novel, but in fact, we found no such usable information (e.g. scientific paper) to help or base our work in the field of DoS protection of SMTP servers.

*This work was performed in the frame of the EU project IST-026600 DESEREC, "Dependability and Security by Enhanced Reconfigurability". The authors thanks for the help and support of all project partners.

2 RELATED WORK

The available information of SMTP server benchmarking is limited. There are some comparisons on different MTAs carried out by independent persons (e.g., [8], [9], [10]). Some of these do not give enough detail to get information about possibility of DoS attacks and mainly performance is covered, while other papers do not give enough insight into the testing. Many of the information available today is also outdated.

There are also some tools to help measuring SMTP throughput, like [11] or the more sophisticated tools like smtp-source included in Postfix ([15]). These tools are helpful, but even if they are available, the analysis has still to be done.

There is also a number of performance analysis on proprietary software, e.g., on MailMarshal ([12]), but their main goal is to show how much their product is better, they are not focusing on the problems like DoS attack. Whole methodologies of performance testing is also available, like for Microsoft Exchange the MMB3 benchmarking standard ([2]) was developed. This methodology is very sophisticated but aims to gather data on the general behavior of the SMTP server rather than to focus only SMTP delivery. SMTP delivery is important, because incoming e-mails are under the control of a possible attacker. In our work we focus on SMTP performance regarding DoS attacks.

In case of distributed denial of service attacks (DDoS) the service on the server is hanged due to the high load several machines cause by requesting the same service from the

same server at the same time. Regarding DoS and DDoS attack several research papers are available (e.g. [3], [4], [5]), and many other papers mention the probability of application specific DoS attacks. Sometimes even SMTP DoS attacks due to e-mail flooding are directly mentioned, but they do not give enough information about the circumstances under such attacks can happen.

3 DOS OF SMTP SERVERS

Several reasons can result in denial of service in SMTP servers. SYN flood or network traffic flood or attacks on special programing errors are the most basic DoS attacks against servers. These are general attacks, not specifically designed to harm SMTP servers. In our paper we focus on the possibility of basic DoS attacks specifically designed against SMTP servers. An example for specific SMTP DoS attack is when a malicious attacker starts an intended attack by sending thousands of e-mails with several machines to one single SMTP server. SMTP DoS can happen also unintendedly, where the high e-mail load is not due to an attacker that wants to harm the SMTP server, but, for example, by non malicious users that want to send out too many e-mails at the same time (e.g., electronic newsletters). Sometimes infected computers (e.g., by virus or e-mail worm) in protected areas (company intranet) can flood the company SMTP server resulting the server to hang. Spammers can cause DoS unintendedly by Directory Harvest Attacks (DHA), where they are trying to collect valid e-mail addresses. We can also mention backscattering attacks, when the attacker forces the SMTP server to generate a huge number of non-delivery reports (NDA) [6].

In all these cases after exceeding a certain load the SMTP server can not any more provide the same performance as before, its performance decreases, e.g., e-mails are delivered only after hours, or even the whole machine can crash. Looking at the situation a bit more deeper the following things happen with the SMTP server. If the size of the e-mail queue is big, which may be the case due to undelivered mails, then the periodically attempts to deliver the e-mails in the queue cost a lot of effort. For example in case of Exim ([13]) if we set the retry parameter to 5 minutes, then one retry attempt to deliver thousands of e-mails in the queue could take significant time and resources in every 5 minutes. This operation leaves only a low percentage of the normal performance to do something else, e.g., to handle incoming e-mails.

Under heavy load, most MTAs change their behavior and turn into a queueing-only mode. In this mode they receive e-mails from the network, they act as an SMTP server, but they put these e-mails into a temporary queue and do not start any processing or delivery action until the system load returns to the normal, or other events trigger the delivery function.

In case the load is high and the incoming e-mail has to be stored in the queue, then depending on the storage method this can mean creation of a file and a deletion of it after delivery, which cause additional load again, but this time on the filesystem. So in case the e-mail can be delivered immediately after receiving, then these filesystem operations can be avoided and the overall e-mail delivery throughput is higher than in case of using the mail queue.

So all in all overload has a bad effect on performance, since in an overload situation the server can consume more resources to deliver the same performance and it increases the probability of a DoS.

3.1 Definition of SMTP DoS

Generally, overloading SMTP servers does not cause a system crash. In fact, SMTP protocol contains countermeasures for DoS attacks. If the load is too high, the server can stop receiving mails with temporary errors or just by refusing connections. As SMTP is a delay tolerant service, the other party can send the particular e-mail later.

Definining a DoS condition as system crash or e-mails lost is useless. The most important thing is the user experience. If a message is received only after 3 hours, it can easily be understood as a DoS for that receiver party. So our definition of the SMTP DoS is the case, when the delivery process is harmed so much that the legitimate e-mails are affected with a non-tolerable delay.

In our investigation we do not want to identify the exact amount of delay or the actual point when the DoS situation is clearly visible. We tried to figure out the maximal throughput of the SMTP server. If the amount of messages exceeds this bandwidth, it surely causes delays. If the amount of the SMTP traffic exceeds much of the maximum sustainable traffic, then it can surely cause a bad user experience, or, as we defined sooner, an SMTP DoS.

3.2 Factors of SMTP performance and Problems of Benchmarking SMTP Servers

We identified the following factors that influence the performance of SMTP servers. Measuring all these performance factors individually is a very hard task and can be very complex. But if we don't measure individual factors, just the performance of the whole system (which is also very complicated), then we loose significant information about the reasons of the difference between different solutions.

3.2.1 Processor Performance

Processor performance is clearly an important aspect of SMTP performance. E-mail delivery does not seem to be

a processor consuming activity, but SMTP become quite complicated due to the appearance of additional tasks like sanity checking, re-writing, spam filtering. Taking all these mail delivery related functions into account processor performance can be a real bottleneck.

3.2.2 Memory Performance

Memory size and speed is not the biggest issue in e-mail transfer, since e-mail delivery is basically a data-transfer. However, in the past years the memory consumption of MTA solutions became higher than before, due to excessive using of Bayes databases.

3.2.3 Network Performance

Since SMTP is an internet service, network performance is important, but we show, that not only network bandwidth is important regarding DoS attacks.

3.2.4 Storage Performance

Storage (hard disk) performance is one of the key issues in SMTP delivery. Not because emails needs high storage capacity, but the real reason is that a typical mailbox and a typical SMTP working queue can contain thousands and tens of thousands of messages. If every e-mail message is stored in a distinct file, then simply working with that number of files in directories or accessing so much random data on the disk can seriously affect the performance of the system.

3.2.5 Hardware System and Architecture

Of course, the whole system architecture, the operating system and the I/O speeds also highly affects the performance. SMTP performance can clearly benefit from using multiple, low latency disk drives with large caches (or hybrid disk drives in the future), or multiple processor cores.

3.2.6 Software Performance

The right software should be applied to the right hardware to obtain optimal SMTP performance. This is true in general, but SMTP is not about the performance. SMTP is used because of the off-line properties, the reliability, the rich features amongst other things. It is not easy to compare different e-mail MTA solutions, as the goal of the software development could be very different. A software could be developed to be feature-rich and easy-to-administer or highly compatible, while other softwares could built to be robust, high-performance or maximum security. In our test the different MTAs show different properties, but the e-mail delivery performance is not the single property we should be aware when we select an MTA software for use.

4 BENCHMARK METHODOLOGY

It is not straightforward to provide a benchmarking methodology that can be fitted to all kind of SMTP server implementations, since each and every real life MTA configuration is different. For example, there are different hardware in use, the number and the type of the handled domains and the amount of users is also varying, and the e-mail traffic is also not the same with respect to the size and the number of e-mails.

In a real life situation during the attack also legitimate traffic exists, which varies among different domains, different time periods of the day, so it is impossible to identify when the system will go in another phase neither in which phase the system can be really found. The question, whether the system will receive more e-mails or the attack is over, can not be answered.

4.1 MTA software under test

For our performance measurements we selected some of the most widely used MTA software. We have to emphasize, that our measurements are not elaborated deeply enough to directly compare the performance of these software solutions with each other, as the exact settings, system parameters and system tweaking can make significant differences in their performance. Our intention is only to show some information on how general MTAs behave under heavy load and to investigate the possibility of DoS attacks against general real-life content-filtering SMTP servers.

Our selected MTA software solutions were the followings:

- Qmail [14]: We investigated *netqmail 1.05* with factory default settings.
- Exim [13]: The debian *exim4-daemon-heavy (4.50-8sarge2)* package was used in our tests with Maildir delivery format and in some tests we used the queue_only_load parameter. We used it as a standalone daemon with a ”q1m” queuing parameter.
- Postfix [15]: We took also *Postfix v2.3.6* under investigation.
- Sendmail [16]: *Sendmail v8.13.8* was used, *Queue_LA* was set to 8 to get similar behavior to the others. *Refuse_LA* parameter was set to 40 to avoid connection refused messages. We manually started delivery process after the queue-only phase. For local delivery (MDA) we used procmail and maildir format.
- Microsoft Exchange[17]: We checked also Microsoft Exchange 2003 on a Windows Server 2003.

4.2 Hardware environment

Our test hardware environment consisted of some basic hardware, which can be a model for a mail server of a small company or academic institute. Our test computers, both on client and server side are 2800 MHz equivalent 32bit, single core, standard PC, with 1 GB RAM and SATA hard drives. For OSS software we used Debian Linux with kernel 2.6, for commercial software we used Windows Server 2003. We applied only minimal tweaks in the default installations, so we do not expect that this causes a big impact to the validity of our measurements. In Linux environment we did not use any processor or memory consuming software, like X Windows, whereas in the Windows environment the GUI surely consumed some system resources. Our test network was a standard 100 Mbps ethernet network with 100 Mbps direct internet connectivity.

4.3 Botnet architecture for testing

For our tests we installed the so called *botnet* architecture. An example for botnet can be seen in Figure 1. The botnets usually work the following way: A virus infects plenty of computers which are connected to the internet, and it connects to an Internet Relay Chat (IRC) server as an IRC bot to a specific channel, which is known by the attacker. So an attacker needs to connect to the iRC network, join the specific channel and give out commands to the bots that will make the machines do what the attacker asks for. In this way a large number of computers (zombies) can be controlled in a resilient way, as the IRC network provides dependable services.

Our main goal to utilize botnets was to avoid attacking computers being a bottleneck and we chose the botnet solution to coordinate our tests and initiate the attack from multiple computers. We set up a botnet that is similar to the generally used technique described above. Our botnet architecture can also be easily extended by new hosts to carry out some large-scale test attacks. We use an IRC server and automatic programs (bots), but unlike the real-life simple trojan programs, our bots are general, feature rich and extendable internet bots, so called Eggdrops [18]. We also set up controllable small scripts in TCL which can perform tasks according to our needs, like initiating an attack by sending plenty of e-mails.

4.4 Phases of the measured process

As we mentioned above, most SMTP servers have two basic modes of operation: Normal operation is instant delivery of received e-mail messages, whereas under heavy load they change to queuing only mode, when they put e-mails in a temporary queue and deliver them later. In our test we caused a high load on the server, and we could observe 3 phases:

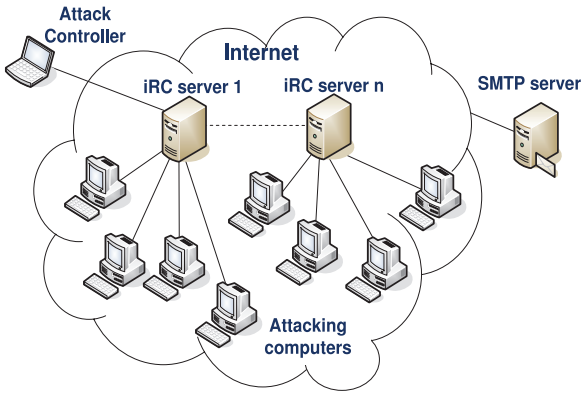

Figure 1: Botnet

- During **Phase 1** The server receives and delivers email until the load reaches the limit, and the server switches in queuing-only mode.
- During **Phase 2** The server receives messages and puts into a temporary queue, delivery is limited or does not happen at all.
- During **Phase 3** The server gets back to normal load that initiates (automatically by measurements or by timer or manually) a delivery process to deliver e-mails in the queue.

To give a better insight into the performance and activity of the system we divided our results by the different phases of the test. We combined Phase 1 & 2 because Phase 1 only takes some seconds therefore not easily distinguishable. For simplicity our tables we refer these phases as ”Phase 1 & 2” and ”Phase 3”.

5 RESULTS

5.1 Generic SMTP throughput

5.1.1 Comparison of different MTAs

We tested the maximum transfer rate (e-mail throughput) on different MTAs, under similar conditions. The number of attacking machines is 3, where each attacking machine was using 5 threads for e-mail sending. This gives a total number of 15 e-mail sending threads, which ensures that the tested SMTP server is fully loaded. The payload of the messages was 4096 bytes static text. During the test we sent 5000 messages from each attacking computer to a single mailbox, resulting 15000 new e-mails in the target mailbox. We used the *smtp-source* utility (included in Postfix) to send the messages. On every open-source MTA (and MDA) we

used Maildir as storage format. We ran the tests 5 times to avoid errors, and to measure the standard deviation of the measured properties.

Table 1 shows the basic behaviour of different MTAs during our test. The maximum e-mail throughput was measured with 10 seconds averages. We selected the "best" 10 second timeframe, where the throughput is calculated as e-mail deliveries per second. This information gives us an upper limit for the maximum sustainable throughput rate, e.g., under normal conditions in the current environment, no delivery rate above 145 e-mails/sec is expected at any time. That belongs to 580 kBytes/s (or $4640kb/s$) transfer rate.

While the throughput data in the phases gives some insight into the delivery process, the real interesting thing is the total delivery time. The throughput rate shows that the delivery was constant throughout the delivery process and the system was fully loaded throughout the test.

From Table 1 we can calculate the average throughput of the e-mail server. We show the calculated values in Table 2. The conclusion of the table is that an expected throughput rate (for long time calculations) in our non content-filtering environment can be between 17-64 e-mail messages per second, that gives about 68-256 kBytes/s ($544-2048kb/s$). This result shows that an SMTP server can be overloaded even within a typical Small Office Home Office (SOHO) system, because the SOHO's ADSL connection has a higher bandwidth than the bandwidth needed to carry out an overloading attack against the SMTP server.

In Table 3 we give some additional information about the queuing activity of the different MTA setups. During our test no e-mail message was lost, therefore the total number of delivered e-mail messages is constant. (To achieve this, we had to set the Refuse_LA parameter of the sendmail MTA very high to avoid connection drops which are not tolerated by our test tools.) However, some differences of workflow can be seen here. For example, Sendmail delivers a lot of messages during the queuing process, but therefore it takes more time to get every e-mail message into the queue.

Table 1: Performance - No Content Filtering

MTA	Maximum throughput during Phase 1 & 2 (e-mail/sec)	Maximum throughput during Phase 3	Avg. Total time for delivery	Std. dev.
Qmail	7.4	145	412.8	15.50
Exim	30.8	60,2	499.0	4.85
Postfix	26.4	145.2	236.2	5.45
Sendmail	31.6	51.2	543.2	10.16
Microsoft Exchange	16.8	35.6	869.0	23.92

Table 2: Average Throughput - No Content Filtering

MTA	Avg. throughput (email/sec)	Std. Dev
Qmail	36.37	1.35
Exim	30.06	0.29
Postfix	63.53	1.46
Sendmail	27.62	0.53
MS Exchange	17.27	0.49

Table 3: Queuing Behaviour - No Content Filtering

MTA	Time for Phase 1 & 2	Std. dev.	Avg. emails delivered during Phase 1 & 2	Std. Dev.	No. of delivered emails
Qmail	133.2	4.9	930.4	105	15000
Exim	296.8	11.6	4639.2	287	15000
Postfix	100.6	0.9	1612.4	113	15000
Sendmail	417.0	7.3	9974.4	280	15000
MS Exchange	489.8	14.0	5734.2	147	15000

5.1.2 Load limiting - queue_only_load in EXIM

While using Exim MTA we can set the queue_only_load parameter to the value of the load average where the server should stop instant delivering and start queuing only mode. In default configuration this option is not used and delivery is constant even during high load situations.

We measured the performance of Exim with and without the queue_only_load parameter and summarized the results in Table 4. We can see that the flow of the process slightly differs (and of course the maximum load average and system response time also differs, but we made no exact measurements). The average throughput rates are very close to each other in this two cases, therefore we can see that the queuing properties does not modify the system performance too much, even the overhead of the queuing process, which is cause by the higher file system activity, cannot be distinguished.

Table 4: Exim - With And Without Queue_only_load Parameter

Limit for Load Average	Maximum throughput during Phase 1 & 2 (email/sec)	Std. dev.	Maximum throughput during Phase 3	Std. dev.	Total delivery time	Std. dev.	Average throughput (email/sec)	Std. dev.
No limit	N/A	N/A	33.8	0.83	505.6	2.1	29.67	0.12
queue_only_load = 8	30.8	1.92	60.2	1.09	499.0	4.8	30.06	0.29

5.2 Content filtering

A large number of content filter solutions (virus and spam scanning) are available in commercial and open-source free software. As our resources to carry out investigations were limited, we decided to test only Amavisd-new ([19]), ClamAV ([20]) and Spamassassin ([21]) under different settings. Amavisd-new is a daemonized middleware for content filtering in SMTP servers. It uses various utilities to manipulate e-mail messages, then it can invoke a number of virus scanning engines and utilize spamassassin for spam scanning purposes. Amavisd can be integrated in many MTAs, in our case we selected Exim with SMTP forwarding to invoke amavisd. During testing of content filtering solutions we still used 3 attacking computers and 5 threads for every computer, but we tested the delivery of only 1500 e-mails to save time. Our test e-mails did not contain mime attachments nor viruses.

We tested amavisd with the popular free ClamAV antivirus. We tested two possible methods, using clamscan, and using the daemonized version of ClamAV called clamd (in this case accessed by the amavisd internal client code for clamd). The clamscan is a stand-alone executable for virus scanning in individual files. To use it, the middleware should execute the clamscan process every time an e-mail is received, therefore it creates a huge overhead (loading the executable, loading virus signatures etc.). Daemonized virus scanning engines are expected to be more efficient in content filtering environment. Such a solution is Clamd, which is constantly loaded into the memory, client programs can communicate with the clamd daemon through a unix socket. Our results are shown in Table 5.

Table 5: Content Filtering - Clamscan vs. Clamd

setup	Total time to deliver (s)	Std. dev.	Avg. delivery rate (e-mail / sec)	Std. dev.
Exim, amavisd-new, clamscan	372.5	7.77	4.03	0.08
Exim, amavisd-new, clamd	220.5	6.36	6.81	0.19

As we can see, the throughput (4.03 e-mail/s in case of clamscan and 6.81 with clamd) is significantly lower than the original 30.06 email/s rate. As we expected, content filtering has a huge impact on performance. We can also see that clamd is much more efficient than clamscan.

For spam scanning, we set up different scenarios with amavisd, clamd and spamassassin. Spamassassin is a framework for spam scanning, it can use different tools, RBLs, Razor ([22]), Pyzor ([23]), bayesian filtering, DCC ([24]) and other plugins to distinguish spam and legitimate e-mail (ham).

Our results are shown in Table 6. In the first test case (first row) we ran spamassassin with razor and internal bayes routines, and fed the SMTP server with fixed messages (generated by the *smtp-source* utility). Then we used random generated e-mail messages for our test with our dictionary based random text generator. The random e-mail message was still 4096 bytes long. The random text generation somewhat slowed down the e-mail injection, but we found no significant difference in average throughput.

Constant e-mail messages are sometimes cached by our content filtering software, therefore testing the same message multiple times will be significantly faster than testing independent messages. This can be seen in the second row, the average throughput is down to 1.59 from the 5.11 e-mail/s. Turning off network-based tests (RBL, razor, etc.) by setting local_tests_only, or turning off razor alone causes nearly the same speed-up. Interestingly, turning off bayes does not make a speed up, in fact, turning off bayes engine makes spamassassin to work slightly slower than with bayes. We could reproduce this behavior, but we do not know the real reason behind this. Maybe the system was not

under full load during these tests because of the I/O locks, so we guess that timings, processor and I/O scheduling has a big role in this effect. We also set up mysql based bayesian testing in spamassassin with a large bayes database, and found it is somewhat faster than the internal bayes-storage engine (1.9 vs. 1.59 e-mail/sec).

We can see, that an Exim MTA with amavisd-new, clamd and spamassassin can result throughput as low as 1.54 e-mail/c. As we used 4096 bytes payload in the messages, the bandwidth is therefore around 6,3 kBytes/sec (50 kb/s). In fact, this is only the size of the payload, so some additional 1000 bytes overhead in every message can be considered.

Table 6: Content Filtering - Spamassassin Scenarios

setup	Avg. total time to deliver	Std. dev.	Avg. delivery rate	Std. dev.
Spamassassin, razor, bayes and fix message	294.0	14.1	5.11	0.2
Spamassassin, razor, bayes and random message	945.0	7.0	1.59	0.1
Spamassassin, local_tests_only, random message	448.0	15.5	3.38	0.2
Spamassassin, no razor, bayes, random message	458.0	4.2	3.27	0.1
Spamassassin, razor, no bayes, random message	975.1	7.4	1.54	0.1
Spamassassin, razor, mysql-bayes, and random message	789.5	9.6	1.90	0.1

6 ANALYSIS OF RESULTS

We tested the SMTP throughput in our test environment, which can be a model for a basic real-life SMTP servers. We found that the e-mail throughput (total average) without content-filtering and without fine-tuning or tweaking can be as high as 64 messages/second, whereas using a medium-fast MTA (Exim - 30.06 e-mail/s) and turning on content filtering can cause a performance drop to 1.54 messages/s. These rates equal to 2.6 millions and 133 thousands of e-mails/day. This looks huge, but in bandwidth the two values are about 50 kb/s to 2000 kb/s + overhead. That means, a network traffic as low as 50 kb/s or 2000 kb/s respectively can cause full load on the system and create a DoS condition. That means, an SMTP server on a single ISDN line (64kbps) can be under a DoS attack without consuming all the network bandwidth.

Modern intrusion detection systems (IDS) can detect a wide range of attacks by signature matching or statistical analysis. Considering an attack with a number of random e-mails sent to our mail server we cannot easily detect such attack by signature matching. Statistical analysis can be therefore a more sophisticated tool to detect such attacks. Most of the state-of-the-art tools are designed to detect traffic jumps or high traffic where the traffic reaches near to the capacity of the network.

If we look at the results we can see that a 50 kb/s traffic is so low comparing to the typical full network bandwidth, that such an attack is not easily detectable with our current tools.

DoS attacks are generally created with a number of zombies. If a zombie computer creates network traffic with the same statistical properties as normal, legitimate users, it is not easily distinguished or filtered from the network. Of course, content analysis can help, but as a perfect spam filtering is not possible with our current tools, the perfect identification of attacking sources is also a hard problem. To send e-mail messages in an amount of about 130k emails/day and to accomplish this with the statistical properties of legitimate users, the attacker will not need too much more servers than several thousands.

We can say that to carry out successful DoS attack against SMTP server seems to be much easier than previously thought and that successfully protecting servers against such problems is a hard, but very important problem.

7 CONCLUSION

In real-life SMTP DoS condition is one of the most general problems. Company administrator continuously try to maintain the stability of their e-mail servers, but with intended and unintended attacks (spam, virus, and such) the SMTP servers often fail to perform properly.

We made empirical experiments on SMTP performance. Measuring SMTP performance is not a simple task, as the delivery process is complicated and different scenarios are not easily comparable. We analyzed a number of different MTAs in different environment regarding content-filtering. We summarized our results in tables showing the gathered

statistical analysis and provided some description to help its understanding.

Our results show that even a relatively small number of messages, small amount of bandwidth can harm an SMTP server significantly. An e-mail flow of 1.54 messages / second, that is about 50kb/s can fully load a server if it is using content filtering software.

Our analysis was carried out by using generally available hardware and software components and we used only small tuning in the OS. Another problem is that the number of software components and systems in our analysis was limited, but this can be extended in the future. Our results are limited in direct applicability on real-life servers, but give basis for engineering estimates (thumb rules).

The results also emphasize the vulnerability against sophisticated DoS attacks, where the usage of statistical analysis against attacks is limited.

As for future work, an extension of our measurements is possible. We did not check the handling of attachments, compressed files and viruses. We also did not investigate the performance of the systems under different parameters of the load generators (number of threads, number of e-mails, size of e-mails). We are confident that these extensions of the measurements can not harm the validity of our current results, but can furthermore help to get an insight into SMTP server performance factors and attack resilience.

REFERENCES

[1] B. Bencsath, I. Vajda ”Protection Against DDoS Attacks Based On Traffic Level Measurements” 2004 International Symposium on Collaborative Technologies and Systems, 2004, pp. 22-28., Simulation series vol 36. no. 1..

[2] Microsoft Corp., ”Exchange Server 2003 MAPI Messaging Benchmark 3 (MMB3)” http://www.microsoft.com/technet/prodtechnol/ exchange/2003/mmb3.mspx

[3] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajovic. ”Distributed denial of service attacks” In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 2275–2280, October 2000.

[4] J. Mirkovic, J. Martin, P. Reiher, ”A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms” Los Angeles, CA, University of California Computer Science Department, 2001. http://citeseer.ist.psu.edu/article/mirkovic04taxonomy.html

[5] J. Mirkovic, P. Reiher, S. Fahmy, R. Thomas, A. Hussein, S. Schwab, C. Ko, ”Measuring Denial-of-Service” Proceedings of 2006 Quality of Protection Workshop, October 2006.

[6] Musashi, Y., Matsuba, R., Sugitani, K. ”DNS Query Access and Backscattering SMTP Distributed Denial-of-Service Attack” IPSJ Symposium Series, Vol. 2004, No. 16, pp. 45-49 (2004)

[7] K. J. Houle, G. M. Weaver, N. Long, R. Thomas ”Trends in Denial of Service Attack Technology” http://www.cert.org/archive/pdf/DoS_trends.pdf

[8] M. Andree ”Postfix vs. qmail - Performance” http://www.dt.e-technik.uni-dortmund.de/ ma/postfix/vsqmail.html

[9] M. Andree ”MTA Benchmark” http://www.dt.e-technik.uni-dortmund.de/ ma/postfix/bench2.html

[10] ”Mail Call - Testing the Axigen, Kerio, and Merak commercial mail servers” http://www.linux-magazine.com/issue/73/Commercial_Mail_Servers_Review.pdf , December 2006.

[11] Smtp-benchmark - a benchmarking suite to measure the performance of SMTP gateways, M. Balmer, http://freshmeat.net/projects/smtp-benchmark/

[12] ”Performance Benchmarking - MailMarshal 6.0 SMTP” http://www.nwtechusa.com/pdf/mm_performance.pdf

[13] Exim - MTA software” University of Cambridge, http://www.exim.org/

[14] Qmail, Netqmail - MTA software, D. J. Bernstein and Qmail community, http://www.qmail.org/

[15] Postfix - MTA software, W. Venema, http://www.postfix.org/

[16] Sendmail - MTA software, The Sendmail Consortium, http://www.sendmail.org/

[17] Microsoft Exchange - MTA software, Microsoft Corp., http://www.microsoft.com/exchange/

[18] Eggdrop, a popular Open Source iRC bot, http://www.eggheads.org/

[19] Amavisd-new - content checking middleware, http://www.ijs.si/software/amavisd/

[20] Clam Antivirus - GPL virus scanner, http://www.clamav.net/

[21] Apache SpamAssassin - Open-source spam filter, http://spamassassin.apache.org/

[22] Vipul’s razor - distributed, collaborative, spam detection and filtering network, http://razor.sourceforge.net/

[23] Pyzor - a collaborative, networked system to detect and block spam using identifying digests of messages, http://pyzor.sourceforge.net/

[24] Distributed Checksum Clearinghouse (DCC) - cooperative, distributed system to detect bulk mail, http://www.rhyolite.com/anti-spam/dcc/