Budapest University of Technology and Economics

New Approaches to Mitigate Network Denial-of-Service Problems

PhD dissertation

Boldizsár BENCSÁTH
MSc in Technical Informatics

Supervised by
Prof. István VAJDA
Department of Telecommunications

2009

Alulírott Bencsáth Boldizsár kijelentem, hogy ezt a doktori értekezést magam készítettem és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

I, the undersigned **BENCSÁTH Boldizsár** *hereby declare, that this Ph.D. dissertation was made by myself, and I only used the sources given at the end. Every part that was quoted word-for-word, or was taken over with the same content, I noted explicitly by giving the reference of the source.*

Budapest, ...

....
Bencsáth Boldizsár

Contents

Summary

New Approaches to Mitigate Network Denial-of-Service Problems

Denial-of-service issues take a distinguished place among the open Internet problems. The problem of DoS has close relation to many other current Internet attacks. During my research I tried to find a comprehensive solution for the problem. Since there is no single general solution, I had to carry out in-depth analysis of the problem, like analyzing the protocol DoS-specific behavior and using the extra information of the protocol to enhance protection.

The results of my research can be divided into the following three claims:

In the first part I tried to deal with the general form of DoS problems, where I analyzed and enhanced the protection of application-level protocols against general DoS problems. My results show that the outcome of this approach is not sufficient to handle every type of the problem and it has certain limitations.

In the second claim I concentrated on the problem of Denial-of-Service attacks against Internet STMP (e-mail) servers. Not counting the very service-specific solutions there are only a few available methods against this threat. I proposed a method to protect against DoS attacks by observing and analyzing the traffic to a particular system.

No matter how excellent general solutions we manage to come up with, specific solutions against DoS problems utilizing special properties of the protected system or service can be more useful than general protection. In my third group of claims I focused on the prevention of DHA attacks, which also can cause DoS conditions. This section contains the analysis of Directory Harvest Attacks (DHA) (a type of attack to collect e-mail addresses). By giving countermeasures against DHA attacks, I also enhance the protection against other problems, like DoS.

Összefoglalás (Summary in Hungarian)

Új védekezési módszerek hálózati szolgáltatásmegtagadásos problémák ellen

(New Approaches to Mitigate Network Denial-of-Service Problems)

A megoldatlan internetes problémák között a szolgáltatásmegtagadásos támadások igen különleges helyet foglalnak el. Ez a terület ugyanis összefüggésben van rengeteg más támadásfajtával. Lehetőség szerint megoldást kerestem általános protokollok védelmére. Nem mindig adható azonban általános megoldás, így figyelembe kell venni azt is, hogy mélyebb ismeretek, pl. a protokol specifikálása mellett milyen többletinformáció áll rendelkezésre a védelem javítása érdekében.

Kutatásaimat három fő csoportba foglaltam.

Kezdetben a szolgáltatásmegtagadásos támadások általános sémájával foglalkoztam, amelyben általános alkalmazás szintű protokollok DoS védettségét elemeztem és növeltem. Kutatásaim azt igazolták, hogy az általános megoldások hatóköre szűk, és noha egyes esetekben megoldást jelenthetnek, alkalmazásuknak számos gátja van.

Második tézisemben a internetes e-mail szerverek szolgáltatásmegtagadásos támadásaival foglalkozom. A specifikus támadásoktól eltekintve itt is szűk a mozgástér: A vizsgálataimat a hálózati forgalom analízisén alapuló módszerekre koncentráltam. Ez az SMTP szerverek esetében jelenthet általánosabb megoldást a problémákra.

Bármilyen általános megoldással szemben sokkal hatásosabbak lehetnek (és néha csak ezek lehetnek hatásosak) azok a megoldások, ahol a konkrét támadást, támadó szándékot, célt és módszert is figyelembe vesszük a védekezés során. Harmadik tézisemben DoS támadások előkészítése ellen mutatok be speciális védekezési módszert. A fejezetben egy speciális problémával, az ún. Directory Harvest Attack (DHA), azaz SMTP szerverek címkigyűjtést célzó támadásaival foglalkozom. A probléma elleni védekezés segíti azt, hogy megelőzzünk intenretes támadásokat, köztük a DoS támadást.

Acknowledgements

First of all, I would like to gratefully thank my supervisor, Prof. István Vajda for his helpful guidance and continuous support. Without his help and inspiration the results presented in this thesis would have never been reached.

I am also grateful to Dr. Levente Buttyán for his valuable comments and suggestions and the inspiration during our joint work.

For our joint efforts and publications I thank the help of Miklós Aurél Rónai, Géza Szabó and Gábor Simon. I thank the technical help of Andrea Nagy, Zoltán Rajnai and Elek Máthé.

I would also like to thank to all the people who established the creative atmosphere in the Laboratory of Cryptography and Systems Security (especially István Berta, Gergely Ács, László Csík, László Dóra, Tamás Holczer, Péter Schaffer, Thong Ta Vinh, László Czap). I would like to thank to my colleagues at the Department of Telecommunications for providing a pleasant working environment.

The financial support of the OTKA grant T046664 of the Hungarian National Science Fund and the support of the 027748, 027795, 026600, 026820 IST FP6 projects are gratefully acknowledged.

Last but not least, I would like to thank the support and patience of my wife Anita, my sons Oliver and Miklós and my family throughout my work.

1 Introduction

The Internet and its protocols have been greatly enhanced in the recent years. During the planning of the Internet the main focus was set on to protect it against outside attacks by not relying on any central element in the Internet. The original plans did not cover the protection against internal attackers.

The Internet can be considered today as a constant battlefield. The Internet is full of continuous attacks, most of which cannot be prevented or properly handled. More and more specialists and researchers propose a complete clean slate redesign of the Internet with considerations of the dangers observed today.

Attacks on the Internet have also been enhanced in recent years. The first viruses and exploits were built to be just proof-of-concept tools or just for the sake of their own ego. Recently, viruses, worms, pieces of spyware and other malicious code have been introduced to enable spamming, phishing or collecting private data. The individual simple attacker was replaced by the commercial-scale attackers who scan thousands of computer. Their goal is to gain control over computers through their attacks and turn them into so-called zombies to help their future goals.

Denial-of-Service (DoS) attacks have been known for a long time. In recent years we have encountered many different forms of the attack. Two of the most basic and widespread versions are the attacks using software bugs to disable (freeze or slow down) the server (e.g. ping of death), and the simple flooding of the network to eat up all the available bandwidth. Meanwhile, protection against denial-of-service attacks did not improve substantially. The main reason behind this is that the best protection against DoS is the appropriate architecture that helps to deal with the problem. The Internet in its present form was not planned to focus on such problems. The DoS problem is therefore a long known problem, but definitely not solved, and to find a solution in the current architecture is difficult. My research is focused on this area, namely, on the possibility of the protection against DoS problems in the current Internet infrastructure.

There are a number of publications about the possibility of DoS attacks. This could mean that these attacks are really wide-spread, but that is not the case. DoS attacks could potentially be a much larger problem than we currently observe. The protection against DoS attacks is very difficult, but it is very easy to initiate a DoS attack. This shows that we have to take this issue very seriously. Although DoS problems are rare, all this could change soon and we are not prepared for that.

We can also say that most of the current DoS attacks are not detected, and users only discover the results through the collapse of the system or service. Administrators could also give excuses for the problems without considering the possibility of a direct attack. In a modern IT subsystem, the complicated chain of software and hardware elements make it very hard to find the real reason, specifically the DoS attack in the background of the system collapse. Therefore, detection of such attacks can be one of the hardest problems, and sometimes the protection against a detected attack is easier than the detection itself.

The problem of spam is a good example of the problem of detection. The biggest part of today's Internet traffic is unsolicited e-mails, spam. The current state-of-the-art method against spam is the usage of spam-filtering software, but they consume a lot of server resources. The larger number of e-mail messages received by the server means larger resource requirements that can lead to a denial-of-service problem. This way, the system could collapse. A DoS situation occurs, although the goal of the individual attacker was not a DoS attack and an individual attacker alone might not cause a DoS. The result of the enormous amount of e-mails is a system collapse, but the analysis of this collapse is a very hard problem because it is not a standard performance planning problem based on service capacities and client needs, but rather a problem based on the marketing processes of some attackers. The usual dependability and performance analysis might be useless in this case, because these tools focus on real service needs and not on completely different problems such as spam.

During my work I focused on problems that do not have sufficient coverage in the literature. Some may think that these problems are solved, others might not recognize the importance of these problems, or they tried to come up with solutions to this problems but did not solve them. These problems include the modelling of the DoS attacks, protection against the attacks, formal description of the attacks, analysis of the protection methods.

1.1 Research objectives

My preliminary research objective was to give new solutions and methods in the field of DoS that can help to make the Internet safer. My objective included giving models of the attacks, scientifically elaborated countermeasure methods, analysis, and important data for the handling of the problems.

Two different approaches can be distinguished during the analysis of Internet attacks. The academia tries to model the problems, to give formal methods, and to simplify the problems in order to handle them. This approach has limitations, sometimes the proposed method cannot be practically used because of simplifications. The other approach is the practical or engineering approach, where people try to give practical solutions to the problems, without an in-depth analysis, or scientific proof. They just think that the proposed method works, because their proposal is based on practical experience. In many cases their results were not or cannot be analyzed, there is no proof for the efficiency of the solutions, and the generalization of their work is problematic.

My goal was to incorporate the two approaches mentioned above. I wanted to handle practical problems with the approach of an engineer and meanwhile maintain the scientific method, therefore propose a scientific analysis and formalism. This can be very problematic in the field of DoS attack protection, as the attacks cannot be handled successfully using current scientific methods, and simultaneously, practical solutions can only be applied in some of these problems.

During my research I observed that Denial-of-Service takes a distinguished place among open Internet problems. The problem of DoS has close relation to many other current Internet attacks. In the course of my research I tried to find a comprehensive solution for the problem. Since there is no single general solution, I had to carry out in-depth analysis of the problem, as analyzing the protocol DoS-specific behavior and using the extra information of the protocol to enhance protection.

The main goal of my research was therefore to give new solutions to enhance the protection against DoS and to give models and methods to facilitate the scientific elaboration of these problems.

1.2 Research methodology

In the first phase of my research I tried to gain practical experience regarding Internet security problems. By my scientific and industrial expertise I encountered problems that could be solved using the current approaches and solutions. I started to model these problems and to plan and deploy protection methods against them. I tried to analyze the problem using both standard and novel methods too. I validated my analytical results with simulation to avoid errors within the formal methods of the research. After obtaining formal results I went back to the practical questions of the specific problem. I built a prototype application to show that the proposed method is workable and helps to solve the problem.

Definition of Denial of Service problems

I defined the denial-of-service condition (Denial-of-Service, or DoS) as follows: *If the functionality of the system or network service in the field of informatics is degraded so much that their clients cannot accept that degradation of the service (where the clients can be real persons or other entities such as software components), and the cause of the change is not a physical one, then we state that the system went into a denial-of-service condition.*

Meanwhile, Denial-of-Service attack (DoS) is defined in the following way: *If a system or network service in the field of informatics is harmed by a third party who intentionally tries to degrade the performance of the service by his or her wishes to a denial-of-service condition, then we talk about a Denial-of-Service attack*

The Denial-of-Service (DoS) attack always means that there is an attacker behind the results. Meanwhile, the DoS condition can also be the result of one of the several other circumstances: for example, if the attacker sends unsolicited e-mails to our system, and therefore he causes the mailing system to freeze (and unable to handle future requests), then we can say that a DoS problem occours, and we cannot really say that a a DoS attack happened. The problem of DoS attacks is very much related to the problem of dependability and resilience, but the goals of those areas of research differ considerably: in the area of DoS we investigate a problem of security, therefore the usable methods and solutions can differ significantly from the other solutions and results shown in the field of dependability.

During my research it was not really necessary to distinguish between a real DoS attack and an unintentional DoS situation. My proposed methods aim to handle DoS conditions whatever their real reasons are. Whenever it is really necessary to distinguish between a DoS condition and a DoS attack, I will indicate it appropriately.

2 Claims

Claim 1 - Protection method against DoS attacks with the usage of client-side puzzle techniques combined with game-theoretical methods

Supporting publications: [J1], [C2], [C3]

I propose a novel game theoretical approach for the analysis of the client-side puzzle technique against DoS attacks. I determined the optimum of the game depending on the cost parameters. I constructed a novel client-side puzzle challenge, which uses multiplications of prime numbers. The advantage of the proposed algorithms is that its computational complexity can be precisely computed.

I proposed a solution against DoS attacks with the use of a client-side puzzle technique combined with game-theoretical methods. I analyzed and described in detail how current protocols can be enhanced to be protected against DoS attacks with the proposed use of a client-side puzzle technique. I described in detail how the proposed technique enhances the protection against DoS attacks. I also defined methods to describe the behavior of the attacker and the server in a game theoretical way and I also showed how the usage of mixed strategies can enhance the protection against DoS attacks. The protection against DoS attacks can be enhanced with my proposed approach that combines game theoretical analysis and client-side puzzle technique. A further advantage of my work is that with the usage of both methods the analysis of DoS protection is easier and therefore better protection methods can be proposed.

I propose a scalable client-side puzzle challenge that uses the product of prime numbers. It is not easy to propose a client-side puzzle challenge with the necessary properties against DoS attacks. The generation of the challenge should be fast and also the verification process should be easy. The storage need of the puzzle at the server side should be low. The computational need at the client side should be given. My proposed client-side puzzle in this section carries a number of advantages and good properties, it is well understandable and the implementation can be also easy. The biggest advantage of my proposed puzzle against hash-based solutions is that it uses only basic operations such as multiplication and therefore the resource need of the puzzle is much more appropriate. To support the applicability of my proposed client-side puzzle, I give the necessary parameter computation through the analysis of the computational need of the puzzle.

Claim 2 – Protection based on traffic analysis against Denial-of-Service problems in Internet e-mail environment

Supporting publications: [C1], [C4], [C6]

I propose a new protection method against DoS attacks based on network traffic analysis. The proposed method does not need the modification of the network elements outside the victim server and minimizes the number of legitimate sources blocked by the server.

By performing measurements I show that it is possible to carry out successful Denial-of-Service attacks against SMTP servers with only a low bandwidth usage, and the possibility is even more feasible when using content filtering combined with the SMTP server.

My proposed new protection method is based on traffic analysis. In a front-end module we identify DoS attacks and filter out attackers. I show how the probability of false positive and false negative error event can be computed.

I give an upper bound to the error rate of the detection algorithm, and to the probability of false identification. These calculations help us to the design the real-life parameters of the protection system. Simulations confirm analytical results and help in further investigation of the sensitivity of the parameters

I designed an architecture for my system to be successfully inserted into a real-life SMTP scenario. Based on the proposed architecture I designed and developed a prototype of my proposed protection method.

Claim 3 – Protection against DoS preparation attacks

Supporting publications: [J2], [J3], [C7], [C8], [C5], [C9]

In this claim I propose new methods to counter some DoS preparation attacks, such as preventing the Directory Harvest Attack and identifying malware infected computers. Concerning Directory Harvest Attacks I propose a new architecture and method to protect against the attacks and I define an optimal algorithm of the DHA attack. For the problem of virus infected hosts I propose a novel method for the identification of virus infected computers and for the notification of the owners of such computers. In contrast to other methods it does not need active elements on the client side, as it is based on some information stored on the computer.

I analyzed e-mail Directory Harvest Attacks. DHA is typically used to collect e-mail addresses for spamming, and at the same time, it can lead to a DoS condition by itself. I designed an optimized DHA attack, where the optimization is based on the probability distribution of e-mail local parts and the expected number of valid addresses in the targeted domain. I give formula for the expected number of successfully collected valid e-mail addresses. I prove that in my model the attack is optimal, the expected number of the successfully collected e-mail addresses is maximal. I illustrated with simulation results that the optimal attack method is more efficient than the non-optimized attack. The simulation results based on real data supported the feasibility and efficiency of my proposed algorithm. I designed a new centralized, real-time blacklist (RBL) based architecture against DHA attacks. I also developed a prototype system based on the proposed architecture to show the applicability of my proposal.

3 Protection against DoS problems with client-side puzzle approach combined with game theoretical methods

Besides confidentiality and integrity, *availability* is one of the most important general security requirements in computer networks. Availability of a system means that it is accessible and usable upon demand by an authorized entity, according to performance specifications for the system [28]. In other words, a system is available if it provides services according to the system design whenever users request them.

If only accidental failures are considered, then replication methods can be used to ensure availability. Replication in itself, however, is not enough against malicious attacks that are specifically aimed at the loss of or reduction in availability. Such attacks are commonly called Denial of Service (DoS) attacks.

Roughly speaking, two types of DoS attacks against an on-line server can be distinguished: bandwidth consumption attacks and server resource consumption attacks. In a bandwidth consumption attack, the attacker floods the server with requests so that the server becomes overloaded and cannot accept requests from legitimate clients anymore. In a resource consumption attack, the attacker sends some requests to the server such that the server uses up all of its resources to process the requests and that is why it can no longer accept requests from legitimate clients. Of course, flooding the server may also lead to the exhaustion of all server resources (e.g., too many processes are launched to serve the requests), but depending on the type of the service provided by the server, a few, carefully constructed requests may have the same effect too. In this section, we are concerned with resource consumption attacks.

Interestingly, simply authenticating the clients may turn out to be not so useful against a (server) resource consumption attack. This is because the authentication procedure may involve expensive operations in terms of resource consumption at the server side, and therefore, engaging into multiple parallel instances of the authentication protocol itself may consume all server resources.

An approach that alleviates the problem is to use *client puzzles* [7]. The idea is that before engaging into any resource consuming operations, the server sends a puzzle to the client, who has to solve the puzzle and send the result back to the server. The server continues with processing the client's request (which may involve resource consuming operations) only if it received a correct response to the puzzle. This approach does not make resource consumption attacks impossible, but it makes them more expensive for the attacker in the sense that successful attacks require considerably more resources from the attacker. Moreover, by varying the complexity of the puzzle, the cost of an attack can be fine-tuned too; this allows one to adjust the system according to the assumptions he has about the strength of the attacker.

In this claim the client puzzle approach using game theory is analyzed. We model the situation faced by the DoS attacker and the server as a two-player strategic game. In this game, the server's strategy is characterized by the complexity of the puzzles it generates, whereas the DoS attacker's strategy is characterized by the amount of effort he invests in solving the received puzzles. Our analysis of the game gives useful insights into the client puzzle approach; we show, for instance, that under certain conditions (which mainly depend on the computational cost of the different steps of the protocol) the optimal strategy for the server is a mixed strategy where it generates puzzles with various complexity levels according to some probability distribution. Our main result is the derivation of the optimal strategy for the server in all conceivable cases.

3.1 Related work

Some DoS attacks on the Internet (Amazon, Ebay (2000), DNS root servers (2002)) are analyzed in [17]. Several methods have been proposed to alleviate the problem of DoS attacks in general (see e.g., [26, 5]), and to make cryptographic protocols resistant against DoS attacks in particular (see e.g., [20, 18, 4, 1]).

The concept of cryptographic puzzles were originally introduced by Merkle [22]. Later, Dwork and Naor used them to combat against junk mails [3]. Juels and Brainard introduced the idea of client puzzles to prevent TCP SYN flooding [7], whereas Dean and Stubblefield applied the approach to the TLS protocol [2]. Time-lock puzzles that cannot be solved until a pre-determined amount of time has passed were introduced by Rivest, Shamir, and Wagner in [27]. Our proposal of client-side puzzles were first presented at Hajducrypt in July, 2002 [C3], and published in the proceedings of SoftCom 2003 [C2]. The original idea of using hash functions for a client-side puzzle were developed by A. Back [11] (first introduced in 1997 in a USENET newsgroup) and enhanced to an Internet draft [10] in 2003. Lately, memory-bound puzzles were in focus of the research as they are believed more appropriate in the Internet environment [8]. After several years of availability of research results, client-side puzzles, or proof-of-work solutions are still not widely deployed. Historical information and analysis on this can be found in [9].

Other papers also propose game theoretical approach in the field of network security. (e.g. [31, 32], or in [29, 30], where the game theoretical approach is used in sensor networks to protect against DoS attacks. Also, as follow-up to our work, several papers refer to our work ([33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45]).

A formal framework for the analysis of DoS attacks has been proposed by Meadows [21], but her approach is not based on game theory. Game theory has already been used to study network security problems [19, 6], but those papers do not address the problem of DoS attacks directly. Michiardi and Molva [23] used game theory to reason about a mechanism that would prevent a very special form of DoS attack (packet dropping) specific to wireless ad hoc networks. Our work is different as we focus on how to protect an on-line server from DoS attacks on the Internet.

3.2 Strategic game model

In this section, we construct a game based model of the client puzzle approach. For this, we first describe the client-side puzzle approach in an abstract way, and then we define a strategic game that is intended to model the situation the DoS attacker and the attacked server face when the client puzzle approach is used.

3.2.1 An abstract view of the client-side puzzle approach

Using the client-side puzzle approach means that before engaging in any resource consuming operations, the server first generates a puzzle and sends its description to the client that is requesting service from the server. The client has to solve the puzzle and send the result back to the server. The server continues with processing the request of the client, only if the client's response to the puzzle is correct. This is summarized in the following abstract protocol, where C and S denote the client and the server, respectively:

step 1	$C \rightarrow S$:	sending service request
step 2	S :	generation of a puzzle
step 3	$S \rightarrow C$:	sending description of the puzzle
step 4	C :	solving the puzzle
step 5	$C \rightarrow S$:	sending solution to the puzzle
step 6	S :	verification of the solution
if the solution is correct:		
step 7	S :	continue processing service request

One can view the first six steps of the protocol as a preamble preceding the provision of the service, which is subsumed in a single step (step 7) in the above abstract description. The preamble provides a sort of algorithmic protection against DoS attacks. The server can set the complexity level of the puzzle according to the estimated strength (computational resources) of the attacker. If the server manages to set an appropriate complexity level, then solving the puzzle slows down the DoS attacker who will eventually abandon his activity.

3.2.2 Defining the game: players, strategies, and payoffs

Given the above abstract protocol, the DoS attacker and the server are facing a situation in which both have several strategies to choose from. The strategies that they decide upon determine the outcome of their interaction, which is characterized by the amount of resources used by the server. The goal of the attacker is to maximize this amount, whereas the server's goal is to minimize its own resource consumption (in order to remain available for genuine clients). This situation can naturally be modelled using the apparatus of game theory [25].

Accordingly, we define a strategic game, where the players are the DoS attacker and the victim server. The attacker has three strategies:

- A_1: The attacker executes only step 1 and then quits the protocol. The attacker may choose this strategy when he does not want to waste resources to solve the puzzle or is not able to maintain two-way communication with the server (e.g., he is using a spoofed IP address).

- A_2: The attacker lets the protocol run until step 3, and then, instead of solving the puzzle, it generates some random sequence of bits and sends it to the server as the solution to the puzzle. By using this strategy, the attacker coerces the server to carry out the verification step (step 6) of the protocol without actually solving the puzzle. The probability that the garbage sent by the attacker is a correct solution to the puzzle is negligible, and therefore, the server will not process the service request any further.

- A_3: The attacker solves the puzzle and sends the solution to the server. Actually, this strategy corresponds to the correct behavior of a genuine client. As a result of following this strategy, the attacker coerces the server to process the service request, and thus, to consume considerable amount of resources. In fact, the role of the protocol preamble (steps 1 through 6) is to discourage the attacker from frequently selecting this strategy during the attack period.

The server has several strategies too, each of which corresponds to a particular complexity level of the puzzle. The selection of the complexity level can be based on attack indicators, which are continuously maintained by the server. It is important to carefully select the complexity level of the puzzle: if the complexity level is too low, then the puzzle will be ineffective in deterring the attacker from mounting the attack; on the other hand, the complexity level should not be too high either, because that would result in an unnecessarily complex verification step at the server. In order to make the presentation easier, we assume that the server can choose between two complexity levels only, which we call *low* and *high*. The corresponding strategies are denoted by S_1 and S_2, respectively. We note, however, that our results can easily be generalized for the case when the server can choose among more than two complexity levels.

We denote by $G(A_j, S_k)$ the outcome of the game when the attacker and the server choose strategy A_j ($j \in \{1, 2, 3\}$) and strategy S_k ($k \in \{1, 2\}$), respectively. In order to define the payoffs of the players in each of the possible outcomes of the game, we need to introduce some notation for the cost (in terms of resource consumption) of the various steps of the abstract protocol. This notation is summarized in Table 1, where k stands for the index of the strategy used by the server. Thus, $c_a(1)$, for instance, denotes the resource consumption cost of solving a low complexity puzzle, and $c_a(2)$ denotes the cost of solving a high complexity puzzle. Naturally, we assume that $c_g(1) \leq c_g(2)$, $c_a(1) < c_a(2)$, and $c_v(1) \leq c_v(2)$. (where c_g stands for the cost of the generation of the puzzle, c_a for the cost of answering and c_v for the verification of the puzzle.

Depending on the puzzle used, a typical implementation of the algorithms (puzzle generation, puzzle solution, puzzle verification), by careful design, can be very close to the optimality (solving the puzzle with the least CPU and memory consumption). For such implementations, the needed memory and CPU clock cycles can be precisely calculated or measured, if the system architecture is known. The architecture used by the attacker and by the server is typically not secret, or can be detected with different methods by the other party, therefore, we can assume that these cost items can be precisely estimated by both the attacker and at the server side.

In Table 1 the costs noted by $p1$ and $p2$ are belonging to the A_3 attacker strategy, when the client-side puzzle handshake is fully executed. It this case, the attacker successfully forces the server to carry out massive calculations, thus c_p should be a small cost at the attacker side, whereas c_e is a large resource consumption at the server side.

In Table 2 we summarized the total cost for each party if using pure strategies. A pure strategy means that the strategy of the appropriate party is constant. If the player selects it's strategy

Step	Cost at the attacker	Cost at the server
service request (step 1)	c_r	
generating and sending a puzzle with strength k		$c_c(k)$
sending garbage	c_g	
calculating and sending the right answer	$c_a(k)$	
checking the answer		$c_v(k)$
$p1$ protocol query during service provision	c_p	
$p2$ protocol answer		c_e

Table 1: Resource consumption costs

according to a probability distribution, we talk about mixed strategy.

We assign a positive real number R to the attacker that represents the amount of his resources available for the attack. The attacker generally knows the exact amount of these resources for small-scale attacks. For large botnets, the available resources for the attack can change rapidly in time (e.g. many zombie computers are turned off at the end of the day) and so, even the attacker cannot estimate the available resources accurately. The estimation of R for the server side depends on the actual attack scenario. In some cases, the owner of the server can clearly identify the attacker and knows its resources, in other cases only a rough estimate can be given.

	S_k	
	cost of the attacker	cost of the server
A_1	c_r	$c_c(k)$
A_2	c_r+c_g	$c_c(k)$+$c_v(k)$
A_3	c_r+$c_a(k)$+c_p	$c_c(k)$+$c_v(k)$+c_e

Table 2: Full cost of the protocol (k=1,2 server strategy)

The estimation of the server's resources (which is indirectly used in the decision of the puzzle level) can be more precise (for estimation by both the attacker and the server side), because the server side typically uses constant, targeted resources (e.g. one server with certain capacity, or a small cluster of servers), than the attacker. It is also harder to hide the available resources at the server side, than for the attacker.

The estimation of the resources clearly affects the results, therefore an adaptation mechanism can be used if the final measurements show that the resources were incorrectly estimated. In Subsection 3.3.1 a similar mechanism will be shown for adaptation if the resources of the server cannot be precisely estimated.

Based on the R resource constraint of the attacker we compute the number of protocol instances the attacker could run with the server simultaneously as $R/c_{int}(j,k)$ (assuming that the attacker follows strategy A_j in every instance and the $G(A_j, S_k)$ game costs $c_{int}(j,k)$ for the attacker.

Now, we are ready to determine the payoffs for the players in every possible outcome of the game. The payoff for the attacker in outcome $G(A_j, S_k)$ is equal to the resource consumption cost of the server when the attacker and the server follow strategies A_j and S_k, respectively, and the attacker uses all of his available resources R for the attack. The payoff for the server in the same outcome is the (additive) inverse of the attacker's payoff. In other words, the game is a *zero sum* game. For this reason, it is enough to specify only the payoffs for the attacker in the matrix of the game, which is given in Table 3.2.2. The matrix of the game is denoted by M, and the element in the j-th row and k-th column of M is denoted by M_{jk}.

		S_1	S_2
		x_1	x_2
A_1	y_1	$M_{11} = c_c(1) \cdot \frac{R}{c_r}$	$M_{12} = c_c(2) \cdot \frac{R}{c_r}$
A_2	y_2	$M_{21} = (c_c(1) + c_v(1)) \cdot \frac{R}{c_r + c_g}$	$M_{22} = (c_c(2) + c_v(2)) \cdot \frac{R}{c_r + c_g}$
A_3	y_3	$M_{31} = (c_c(1) + c_v(1) + c_e) \cdot \frac{R}{c_r + c_a(1) + c_p}$	$M_{32} = (c_c(2) + c_v(2) + c_e) \cdot \frac{R}{c_r + c_a(2) + c_p}$

Table 3: Matrix of the game

3.3 Solution of the game

In order to be general, we allow each player to select the strategy to be used according to a probability distribution over the set of strategies available for the player. This probability distribution is called *mixed strategy*, because it determines how the player mixes its pure strategies. We denote the mixed strategy of the server by $X = (x_1, x_2)$ and the mixed strategy of the attacker by $Y = (y_1, y_2, y_3)$. This means that the server plays according to strategy S_1 with probability x_1 and it follows strategy S_2 with probability x_2. Similarly, the attacker uses strategies A_1, A_2, and A_3 with probability y_1, y_2, and y_3, respectively.

Roughly speaking, the goal of the players is to maximize their payoffs. However, a more careful look at the problem reveals that there are several plausible ways to interpret this general goal (i.e., several utility functions could be considered). We assume that the server is cautious and it wants to minimize the maximum of its average resource consumption cost, where the maximum is taken over the attacker strategies. In other words, the server follows a *minimax* rule. Accordingly, the attacker wants to maximize the minimum of the server's average resource consumption cost.

Let us assume for a moment that the server uses the mixed strategy $X = (x_1, x_2)$ and the attacker follows his pure strategy A_j ($j \in \{1, 2, 3\}$) (i.e., he uses a mixed strategy where A_j is selected with probability 1). In this case, the average cost of the server is

$$M_{j1} \cdot x_1 + M_{j2} \cdot x_2$$

Similarly, if the attacker uses his mixed strategy $Y = (y_1, y_2, y_3)$ and the server applies its pure strategy S_k ($k \in \{1, 2\}$), then the average cost of the server is

$$M_{1k} \cdot y_1 + M_{2k} \cdot y_2 + M_{3k} \cdot y_3$$

We can obtain the solution of the game by solving the following two systems of inequalities:

For the server:

$$\begin{aligned} M_{11} \cdot x_1 + M_{12} \cdot x_2 &\leq v & (1)\\ M_{21} \cdot x_1 + M_{22} \cdot x_2 &\leq v & (2)\\ M_{31} \cdot x_1 + M_{32} \cdot x_2 &\leq v & (3)\\ x_1,\ x_2 &\geq 0 & (4)\\ x_1 + x_2 &= 1 & (5) \end{aligned}$$

For the attacker:

$$
\begin{aligned}
M_{11} \cdot y_1 + M_{21} \cdot y_2 + M_{31} \cdot y_3 &\geq v && (6)\\
M_{12} \cdot y_1 + M_{22} \cdot y_2 + M_{32} \cdot y_3 &\geq v && (7)\\
y_1,\ y_2,\ y_3 &\geq 0 && (8)\\
y_1 + y_2 + y_3 &= 1 && (9)
\end{aligned}
$$

A graphical representation of the system (1) – (5) is depicted in Figure 1, where the dashed lines correspond to the first three inequalities of the system. According to the minimax rule, the server selects the minimum average cost, which is the lowest point of the shaded region. The optimization step for the attacker is similar.

According to Neumann's classical theorem [24], there always exists a common optimum for the attacker and the server. The corresponding set of parameters $x_1^*, x_2^*, y_1^*, y_2^*, y_3^*, v^*$ are called the solution of the game.

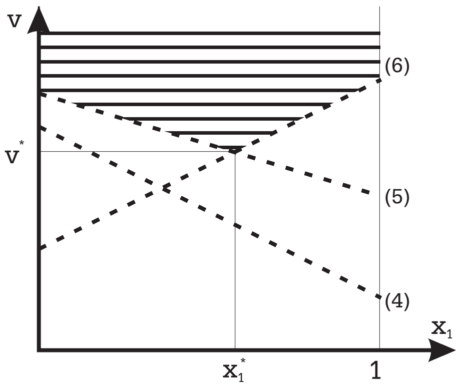


Figure 1: Searching the optimum strategy of the server

Note that if $M_{21} > M_{11}$ and $M_{22} > M_{12}$, then the second row of the matrix of the game dominates the first one. This dominance exists if

$$\frac{c_c(l) + c_v(l)}{c_r + c_g} > \frac{c_c(l)}{c_r}, \quad l = 1, 2. \tag{10}$$

which can be simplified to

$$\frac{c_v(l)}{c_g} > \frac{c_c(l)}{c_r}. \tag{11}$$

The cost c_r of sending the service request and the cost c_g of sending garbage as a result to the puzzle can be considered as the cost of a communication step (i.e., sending a packet). Therefore, $c_g = c_r$ seems to be a reasonable assumption. From here, the mentioned dominance follows if the cost $c_v(k)$ of verification of the solution to the puzzle is smaller than the cost $c_c(k)$ of generating the puzzle. The existence of a dominated strategy simplifies the problem of searching for the solution of the game, because the dominated strategy can be ignored.

Now let us assume that $c_v(k) < c_g(k)$, and hence A_2 dominates A_1. This is illustrated in Figure 1 by the line corresponding to inequality (1) being entirely below the line corresponding to inequality (2). This means that only the latter needs to be considered when looking for the optimum. The lines corresponding to inequalities (2) and (3) intersect at $x_1 = 1/(1+w)$, where

$$w = \frac{M_{21} - M_{31}}{M_{32} - M_{22}} \tag{12}$$

Therefore, we get the following result:

If $w > 0$, then the server's optimal strategy is a *mixed strategy* with probability distribution

$$X^* = \left(\frac{1}{1+w}, \frac{w}{1+w}\right) \tag{13}$$

If $w \leq 0$ then the server's optimal strategy is a *pure strategy* according to the following rules:

- if $-1 < w \leq 0$, then the server's optimal strategy is S_1,
- if $w \leq -1$, then the server's optimal strategy is S_2.

The cost at the server cannot exceed

$$v^* = \frac{M_{21} - M_{22} \cdot w}{1+w} \tag{14}$$

for any attacker strategy.

The different cases of the solution of the game are illustrated in Figure 2.

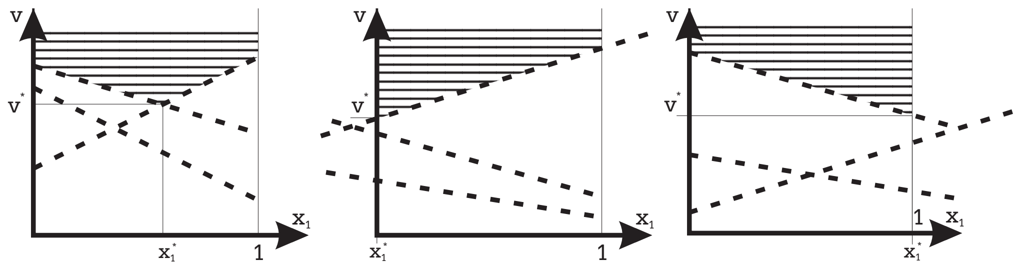


Figure 2: Illustration of the different cases of the game

If the server chooses a pure strategy, the attacker will also choose a pure strategy, in this special game the mixed strategy degenerates into pure strategy. When one of the inequalities in the system becomes strict ($<$ instead of $\leq$), when substituting X^* and v^*, then the probability weight of the attacker strategy's belonging to this becomes zero. If $x_1^* = 1$, $M_{21} = v^*$, $M_{31} < v^*$, then $Y^* = \{0, 1, 0\}$, and if $M_{21} < v^*$ and $M_{31} = v^*$, then $Y^* = \{0, 0, 1\}$. Similarly, if $x_1^* = 0$,

$M_{22} = v^*$ and $M_{32} < v^*$ then $Y^* = \{0, 1, 0\}$ and when $M_{22} < v^*$ and $M_{32} = v^*$, then $Y^* = \{0, 0, 1\}$.

When the server uses mixed strategy, the attacker will also choose a mixed strategy. In our case the attacker will choose randomly from the two strategies A_2 and A_3. y_1^* becomes 0, as the second inequality will dominate the first.

As an example, let us consider the following numerical values for the parameters: $c_r = 1$, $c_c(1) = 14$, $c_c(2) = 16$, $c_g = 1$, $c_a(1) = 40$, $c_a(2) = 400$, $c_v(1) = 10$, $c_v(2) = 12$, $c_p = 10$, $c_e = 3000$, and $R = 100000$. These values lead to a game the solution of which is the following: $v^* = 15679.8$, $x_1^* = 0.16$, $x_2^* = 0.84$, $y_1^* = 0.96$, $y_2^* = 0$, $y_3^* = 0.04$. Thus, the optimal strategies are mixed strategies.

In the two-strategy example a possibility for the strategies at the server side is that the server sometimes asks for a client-side puzzle, sometimes it skips this protection step of the protocol. If a scalable client-side puzzle is available, then the above mentioned multiple strategies can use the client-side puzzle with different levels. In most of the cases there is no need for the server to use too many strategies, only some (in our example 2) strategies might be enough to handle the situation and this also makes the analysis more easy.

However, if the server chooses to use only a low number of strategies, the client side puzzle should be set carefully to the appropriate levels to provide enough protection. In most cases this means, that a scalable client-side puzzle is needed, as the levels will be chosen depending on the resources of both the server and the client. The constraint in this sense is that (in the two-strategy version), the server should maintain it's services, thus the resources of the server cannot be exhausted.

As it was described above, if the attacker uses his mixed strategy $Y = (y_1, y_2, y_3)$ and the server applies its pure strategy S_k ($k \in \{1, 2\}$), then the average cost of the server is

$$M_{1k} \cdot y_1 + M_{2k} \cdot y_2 + M_{3k} \cdot y_3$$

.

If $k = 2$ means the client-side puzzle with a higher strength, then this means an average cost of

$$c_c(2) \cdot \frac{R}{c_r} \cdot y_1 + (c_c(2) + c_v(2)) \cdot \frac{R}{c_r + c_g} \cdot y_2 + (c_c(2) + c_v(2) + c_e) \cdot \frac{R}{c_r + c_a(2) + c_p} \cdot y_3$$

If this is higher than the total amount of resources available at the server, then the server cannot survive an attack, even if it selects the pure strategy with the higher-level puzzle.

3.3.1 Adaptive technique for puzzle level setting

When no a priori information is available on the server's resources, or the available resources change in time (e.g. virtual servers etc.), an adaptive technique can be used. The steps of the adaptations is the following:

1. Set the level of the higher-level puzzle to some initial value L.
2. When there is new information on the resources of the server, recalculate the average cost at the server as described above.
3. If the calculation shows that the cost is higher than the available resources, increase L. Recalculate the game to set the appropriate strategies.
4. If the calculation shows that the cost is lower than the available resources, but the server is overloaded, then the model might contain false information, e.g. the value of R is incorrect. We need to refine the model, then recalculate the game to set the appropriate strategies.
5. If the calculation shows that the average cost is much lower than the available resources, the server might consider to lower the value of L, however, it should check the effects of this change on the game, and recalculate it to obtain the optimal strategy.

If the server can choose from more than two strategies, then a more complex system of inequalities have to be solved in order to determine the solution of the game. Fortunately, the solution can always be obtained by using linear programming tools.

In the following section we show a candidate for an easily implementable, scalable client-side puzzle.

3.4 Proposal for a client-side puzzle

We present two methods to generate puzzles. An important characteristic of a client puzzle is that the amount of computation needed to solve it can be estimated fairly well. Note that the puzzles used in the defense against DoS attacks do not require inherently sequential operations like the puzzles in [27], where it is important that an encrypted message cannot be decrypted by anyone until a pre-determined amount of time has passed. In case of DoS attacks, the amount of resources needed to solve the puzzle is more important than the time. Since, in general, parallelization does not reduce the amount of resources needed to perform a computation, the fact that a puzzle is parallelizable is not an issue here.

Let $T = \{p_1, p_2, \ldots, p_N\}$ be a publicly known, ordered set of different n bit prime numbers. A set S of k primes is selected randomly from T. The selection can be made with or without replacement. Below, we assume that it is done without replacement (i.e., S is a subset of T). The analysis can directly be extended to the case of selection with replacement. The elements of S are multiplied together, and the product is denoted by m:

$$m = p_{i_1} \cdot p_{i_2} \cdot \ldots \cdot p_{i_k}$$

Consider the following two puzzles:

- *Puzzle 1:* The puzzle is to find the factors of the product m.

k	4	8	16	32	64
$D(100, k)$	80.8	89.8	95.1	97.9	99.4

Table 4: Values of $D(N, k)$, $N = 100$

- *Puzzle 2:* Let m' be a modification of m, such that ℓ consecutive bits, $m_r, m_{r+1}, \ldots, m_{r+\ell-1}$ are replaced with zeros in the binary representation of m. The puzzle is to find the factors of the resulting number m', with the knowledge of position r and ℓ.

Puzzle 1 is used in this proposal as an introductionary example, Puzzle 2 is proposed for real-life usage.

For both puzzles, the task of the client is to find the prime factors of m (or m' in case of Puzzle 2), and respond with their corresponding indices in T (recall that T is ordered).

In case of Puzzle 1, the client calculates its response in the following natural way: Let μ be a variable that is updated in each step of the computation. Initially, $\mu = m$. In step i, the client checks if p_i is a factor of μ (and hence of m). If so, then μ is updated to take the value of $\frac{\mu}{p_i}$; otherwise μ does not change. This procedure is repeated until all factors of m is found (i.e., μ becomes 1).

During the above computation, at least $k - 1$ divisions are made[1], where the divisors are n bit size primes, and the size of the dividend decreases gradually from kn to $2n$. The average number of divisions is given by the following formula:

$$D(N, k) = \sum_{i=k}^{N} q_i \cdot (i - 1) \tag{15}$$

where q_i is the probability that the largest index in S is i, and it is computed as:

$$q_i = \frac{\begin{pmatrix} i-1 \\ k-1 \end{pmatrix}}{\begin{pmatrix} N \\ k \end{pmatrix}} \tag{16}$$

When k increases, $D(N, k)$ monotonically and quickly increases to N, and $D(N, k)$ is close to N even for relatively small values of k. In Table 4, numerical values of $D(N, k)$ are given for the instance of $N = 100$.

In case of Puzzle 2, the client is forced to do more calculations. The client may choose from the following two procedures:

- Solution A: The client tries possible substitutions for the missing bits. If a substitution is incorrect, then the client will likely get prime factors that do not belong to set T. In

[1]Note that the last prime factor need not be checked by division.

this case, the client continues with choosing another substitution. The average number of divisions required is approximately $N2^{\ell-1}$, thus the complexity of solving the puzzle is increased with a factor of $2^{\ell-1}$ on average. For instance, in case of parameters $N = 100$ and $\ell = 6$, the average number of divisions required is at least 3200.

- Solution B: The client directly calculates different products of k primes from set T until the tested product m' is obtained. The average number of multiplications required is

  $$\frac{1}{2}\begin{pmatrix} N \\ k \end{pmatrix}(k-1)$$

  For instance, if $N = 100$ and $k = 8$, then approximately $6.51 \cdot 10^{11}$ multiplications are needed.

It is possible that m' is obtained as the product of prime numbers different from those used by the server to generate the puzzle. It depends on the implementation, if the server accepts such collisions, bad solutions. It should be noted that the chance for such collisions, depending on the parameters, might be very low. The chance of such collisions can be set through the value of ℓ and the length of m. In the case of $N = 100$, $k = 8$, $\ell = 15$ and using about 9 bit numbers, m is about 72 bits long. The client will choose the second procedure as less computation is needed. Without the deleted bits, m still contains 57 bits. Therefore, a solution can be found with about approximately $6.51 \cdot 10^{11} \approx 2^{36}$ multiplications, whereas to find a collision, about 2^{57} different trials (with even more multiplications) would be needed.

The proposed solution uses a constant set of prime numbers which can be sent from the server during the protocol run, or can be pre-distributed. The pre-distribution is not very complicated, as the above mentioned case, where $N = 100$ might be sufficient, in this case of 8 and 9 bit integers the set contains 100 numbers between 257 and 887. Due to the size of this set, the usage of this technique can still be considered in low-resource environments, such as sensor networks.

Parameter setting for Puzzle 2

The exact parameters of the puzzle should be set during the implementation. Additional information, analysis and hints about the expected resource needs can also be found in [9]. In their work, the model is improving spam filtering by the client-side puzzle technique. The authors' model is to reduce the amount of false negatives in the spam filtering from 0.06 (found in commercial products) to 0.01. The result of the modeled calculations show that by limiting the capacity of sending e-mails to 75 e-mails/day for every client (Internet user), this target can be achieved. From this value, 75 e-mail/day, with an estimation on the capacity of an average computer and it's typical normal load, it is possible to plan the appropriate level of the puzzle.

In our case, let's consider that an attacker uses 10 computers for a distributed DoS attack and every computer has the same performance, as the protected server, let's say they have 2.4 GHz of CPU power. For $N = 100, k = 9$, to try a single solution of the puzzle, our naive implementation needs about 500 clock cycles. A goal can be defined to maintain only 70% CPU load at the server whenever the attacking computers are 100% loaded to solve the puzzle. Let assume that the generation of the puzzle costs about 400 CPU clock cycles at the server and the protected service needs 40 000 clock cycles to fulfill the request, delivering an e-mail. The rationale of our cost assumptions is based on the empirical performance comparison, which is described in the next section, and on the empirical performance analysis of the SMTP servers in Claim 3. Let's consider all the other cost items to be 0, and let's consider there is no legal traffic to the server.

$n \cdot 40000 + q \cdot 400 = 2400000 \cdot 70\%$, where n stands for the number of successfully solved puzzles by the attacker in each second and q is the number of challenge queries sent by the attackers. Let's consider $n = q$, the attacker does not want to cheat with false queries. In this case $n = 41.6$. The full capacity at the attack is about $2,4 \cdot 10^{10} cycles/s$, so the maximum number of trials is $48 \cdot 10^6/s$. In this case $n = 48 * 10^6/(N * 2^{\ell-1})$. This gives a result $\ell = 14.5$. This means, if e.g. $l = 15$, this system can be protected successfully during the attack within the above described model.

However, there is no reason to anticipate this attack scenario, therefore, the result just gives us some hint in parameter settings. E.g. under normal conditions, a slightly lower strength might be enough, e.g. set $\ell = 12$, in this case, to solve a puzzle, only $500 \cdot 100 \cdot 2^{10} = 51.2 \cdot 10^6$ clock cycles are needed,which makes only a slight, 21ms delay. On the other hand, to provide effective protection against more serious attacks, considering the results and overestimating the attack, the administrator can decide to use $\ell = 17$. In this case, for normal users $500 \cdot 100 \cdot 2^{16} = 3276.8 \cdot 10^6$ clock cycles are needed which causes about 1.4 s delay for legal users with the same processor speed, and the server can withstand an attack with significantly higher than 10 attacking hosts.

Comparison with other proposed puzzle schemes and implementational issues

The original client-side puzzle scheme from Dwork and Naor [3] considered a central authority to store a secret key. Our proposed method does not rely on a central point and applicable to real-life environment. As a comparison to the most widely known client-side puzzle scheme 'hashcash' [10], we can emphasize that our solution does not use hash functions as a central element. We think that this is an advantage as a hash function is a complicated primitive compared to our basic elements (division, multiplication).

The comparison of our proposed client-side puzzle and the above mentioned hash-based methods is not straightforward. First of all, it is hard to strictly define the optimal client-side puzzle. This is closely related to the model, where we expected that the attacker has a certain level of resources and we setup the client-side puzzle based on this information. From this basic model one could say that both our proposed method and the hash-based method is acceptable, as for both methods the resource need might be set appropriately. While this is true, we should mention that a hash function can be optimized and because their complicated structure, it is not evident how much optimization can still be done. The recent problems identified in hash functions ([12, 13]) might also uncover special attacks for puzzles based on hash functions. Therefore, it might be harder to set up the level of a hash-based puzzle, and special attacks and optimizations can help the attacker.

On the other hand, in our proposed method, multiplications of integer numbers is much more

	32-bit 850 MHz	32-bit 2400 MHz	32-bit 3000 MHz	32-bit 3330 MHz	64-bit 3000 MHz	64-bit 3330 MHz
hash/sec - MD5	349800	1399016	1717646	1902212	1962449	2175139
relative speed - MD5	100	399,947	491,0366	543,799	561,020	621,823
hash/sec - SHA1	294409	1303474	1599337	1785359	1874534	2084976
rel. speed - SHA-1	100	442,742	543,236	606,421	636,710	708,190
trial/sec	1329654	4727745	7430386	8154001	11737441	13004626
rel. speed - trials	100	355,562	558,821	613,242	882,744	978,045

Table 5: Benchmark results of the proposed and hash-based solutions

simple and further optimizations cannot seriously modify the resource need. However, the resource need of the multiplication depends on the hardware architecture, e.g. multiplying numbers larger than 32 (16) bits can be significantly faster on 64-bit architecture.

3.4.1 Empirical performance comparison of implementations

To show the difference how the proposed method and the hash based method depends on the speed and CPU architecture, we carried out a benchmark test. We used the OpenSSL 0.9.8i assembly based (optimized) hash function speed tests with 64-byte blocks and calculated the average speed of 10 runs on different CPUs. This was carried out for both SHA-1 and the MD5 hash function. For testing our proposal we made a basic implementation for $N = 100, k = 9$. The implementation uses the GMPlib3 library (version 4.2.1) for multiplying large integers, which is highly optimized. Both test programs were compiled with the same GCC compiler and statically linked. The results (hash/sec and trial/sec) is collected in Table 5.

The relative speed of a method on different CPUs are shown on Figure 3. The figure shows that our proposed method behaves only slightly different depending on the CPU speed than the hash-based solutions. On the other hand the usage of a 64-bit CPU highly affects the speed of multiplications, while it does not speed up the hash functions significantly. The evaluation of the 64-bit speed-up should be done during the implementation of the puzzle, as it can be a drawback in certain situations of the proposed algorithm, or, from another point of view, this behavior also means that the speed of our proposal can be more close to the real computing power of the processor.

3.4.2 Attack against the proposed puzzle and the protection against it

The proposed puzzle has a very simple structure and uses elementary operations (multiplications, etc.). However, no proof is given that there cannot be exist any algorithm that solves the puzzle faster than the mentioned solving algorithms.

However, for some small primes, special attack is possible against the proposed puzzle. First of all, if the set of primes contains the prime 2, a trivial attack is possible. To test a binary number for divisibility by 2, it is enough to test the last binary digit of the number. If it is 0, then the number is a divisible by 2, if it is 1, then it is not divisible.

For puzzle 2, this fact leads to the following possibilities:

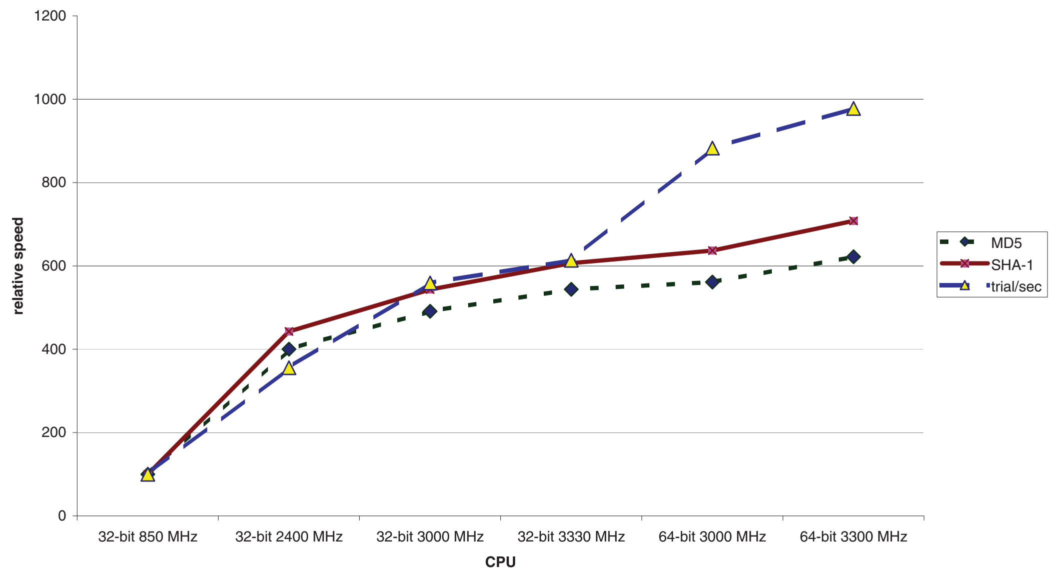


Figure 3: Relative speed

- If the T set of primes contains the prime 2, and the last digit of m' is not replaced with zero (deleted), and the last digit of m' is 1, then there is no need to test with 2, as it is surely not part of the primes producing m.
- If the T set of primes contains the prime 2, and the last digit of m' is not replaced with zero (deleted), and the last digit of m' is 0, then one of the factors of m' is surely 2, no matter what the deleted bits are, there is no need to test divisibility by 2.
- If the T set of primes contains the prime 2, and the last digit of m' is replaced with zero (deleted), then we have no additional knowledge.
- If the T set of primes does not contain the prime 2, and the last digit of m' is not replaced with zero (deleted), then the last binary digit is surely 1, but this does not cause any speed-up of solving the puzzle
- If the T set of primes does not contain the prime 2, and the last digit of m' is replaced with zero (deleted), then the last digit of m is surely 1.

Depending on the above mentioned possibilities the search for the right solution might be speed-up, although this speed-up depends on the size of T and other paramters. To avoid this simple attack, the prime 2 should be excluded from the T set and the last binary digit should not be deleted (replaced by 0) or, the prime 2 might be included in T and the last binary digit should be deleted when producing m'.

For primes 3,5,7 and other primes in the form of $2^n - 1$ or $2^n + 1$ ($n \in Z$), an other form of attack is possible.

Testing of divisibility with 3 in binary representation

The following algorithm can be used to test the divisibility by 3 for a binary representation of a number: Divide the binary representation of the number into 2 digit chunks beginning from the least significant bit.

Multiply the value of every chunk with the value $(2^2)^i \bmod 3 = 1^i \bmod 3 = 1$, where i is a step counter, for the least significant chunk it $i = 1$ and increased by 1 for every chunk. (the 2^2 comes from the size of the chunk, while 3 is the number for which we test the divisisability). In this case the multiplication with one can be omitted as it does not change the value (= 1 for all chunks). Sum the product of the numbers and check if the sum is divisible by 3. If it is divisible, then the original binary number is also divisible by 3.

A sample calculation is as follows. Let's check the divisability by 3 for the binary representation of the decimal 15057, which is 11101011010001.

$$
\begin{array}{cc}
\text{Binary string:} & 11\ 10\ 10\ 11\ 01\ 00\ 01 \\
\text{Decimal value of the bits:} & 3\quad 2\quad 2\quad 3\quad 1\quad 0\quad 1 \\
\text{Multiplied by:} & *\,1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1 \\
\hline
\text{Product:} & 3{+}2{+}2{+}3{+}1{+}0{+}\ 1 = 12
\end{array}
$$

As the result is 12, which is divisible by 3, the original number is also divisible by 3.

Testing of divisibility with 5 in binary representation

The same approach can be used to test the divisibility by 5. Let's divide the binary representation of the number into two bit chunks, beginning from the least significant bit. Assign multipliers to the chunks as $(2^2)^i \bmod 5 = -1^i \bmod 5$, where i is a step counter, for the least significant chunk it $i = 1$ and increased by 1 for every chunk. Calculated the product ofthe chunk multiplied by the assigned multiplier and calculate the sum of the results. If the resulting number is divisible by 5, then the original number is also divisible by 5, or the actual reminder is given.

For the aboved mentioned example, decimal 15057, which is 11101011010001 in binary form this gives the following computation:

$$
\begin{array}{cccccccc}
\text{Binary string:} & 11 & 10 & 10 & 11 & 01 & 00 & 01 \\
\text{Decimal value of the bits:} & 3 & 2 & 2 & 3 & 1 & 0 & 1 \\
\text{Multiplied by:} & *\,1 & -1 & 1 & -1 & 1 & -1 & 1 \\
\hline
\text{Product:} & \multicolumn{7}{c}{3{+}(-2){+}2{+}(-3){+}1{+}(-0){+}\ 1 = 2}
\end{array}
$$

The resulting value (= 2) shows that the original number is not divisible by 5, it gives the 2 as the value for $15057 \bmod 5$.

Similarly to the above mentioned examples, an algorithm can be constructed to test for divisibility by any arbitrary number, but the most practical form is when the number is in the form of 2^{n-1}, 2^{n+1}, for an arbitrary n, such as 3,5,7. The above shown algorithms not neccesarily speed up the solving of the puzzle, as depending on the processor and bit-size of m, the division by such numbers might be already an elementary or very fast operation.

A method for speeding up the puzzle solution

The above defined algorithms can speed up the solution of puzzle 2 in case when the T set does not contain primes like 3,5 or 7. The algorithm is the following for prime 3:

As in the above mentioned algorithm, the binary representation of m' is divided into 2-bit chunks. Then, the values of the non-deleted chunks should be multiplied by 1 and summed together, without the deleted/unknown bits. If 3 is not part of the set T, then m is surely not divisibile by 3. Therefore the sum of the already known chunks and the sum of the deleted bits altogether should sum in a number that is not divisible by 3 if the T set does not contain 3.

A descriptive sample is as follows:

Binary string:	11 10 $XXXXXX$ 00 01
Decimal value of the bits:	3 2 X X X 0 1
Multiplied by:	$*$ 1 1 1 1 1 1 1
Product:	3+2+X+X+X+0+ $1 = 6 + Z$

This way, an efficient algorithm can be constructed that avoids any substitution of the deleted bits, when the result is divisible by 3 although the T set of primes does not contain 3. With a slight overhead, there is no need to test $1/3$ of the possible binary strings for the deleted part of m', as all those possible deleted representations can be skipped from the testing that would be result in a number that is divisible by 3 although 3 is not part of the possible factors of m.

The speed-up of this method is about less than $33\% = 1/3$ as an overhead is needed due to the algorithm. For the prime 5, this speed-up is less than $20\% = 1/5$, and it is related to the previous speed-up if the discarded possibility is both divisible by 3 and 5.

The previously defined speed-up algorithm can be coutinued altough every additional number results in a lower speed-up and still the overhead is not properly counted.

Protection method against divisibility algoirthm speed-up

The protection of the proposed algorithm against this kind of speed-up is very straightforward. If the set T of the possible prime factors contains all the most-probable primes for speed-up algorithms (3,5,7, or even 11,13,17 etc.), then the above mentioned speed-up algorithm is unusable. The algorithm might speed-up the divisibility-test, but we can also consider that modern testing algorithms consider these possibilities. The full-skip for some number is not workable: the attacker cannot be sure if the factors of m contains the affected number or not, therefore no additional information is learned. For higher prime factors, the attack will not result in a significant speed-up, therefore the effect might be discarded.

In summary, the protection against the above mentioned attack is to always include the low primes (e.g. 3,5,7,11,13,17,19) in the set T.

3.5 Summary

I proposed a solution against DoS attacks based on a client-side puzzle technique combined with game-theoretical methods. I analyzed and described in detail how current protocols can be enhanced to be protected against DoS attacks with the proposed use of a client-side puzzle technique. I described in detail how the proposed technique enhances the protection against DoS attacks. I also defined methods to describe the behavior of the attacker and the server in a game theoretical way and I also showed how the usage of mixed strategies can enhance the protection against DoS attacks.

In the first part of this section I studied the client-side puzzle approach to defend against resource consumption DoS attacks. I modeled the situation faced by the DoS attacker and the attacked server as a two-player strategic game. I analyzed this game and gave useful insights into the client-side puzzle approach. I also determined the optimum of the game depending on the cost parameters. The main contribution is the derivation of the optimal strategy for the server in all conceivable cases.

The protection against DoS attacks can be enhanced with my proposed approach that combines game theoretical analysis and client-side puzzle technique. A further advantage of my work is that with the usage of both standard analysis and game theoretical methods the analysis of the DoS protection is easier and therefore better protection methods can be proposed.

I constructed a novel client-side puzzle challenge. My proposed puzzle uses multiplications instead of complicated hash functions, therefore the resource needs of my solution can be much more accurately approximated. I've shown how the computational need can be calculated at the client side for my proposed client-side puzzle

4 Protection based on traffic analysis against Denial-of-Service problems in an Internet e-mail environment

In some cases, the modification of a network protocol is not possible, therefore solutions to the DoS problem such as client-side puzzle technique are not feasible. In this case a more general approach is needed to deal with the problem. In the first part of this section, I show the motivation and importance of the protection by showing how vulnerable is the SMTP environment against depleting attacks. After this introduction, I present my proposal for a protection method that uses traffic level measurements to identify the attack, and the attackers and to filter out the traffic of the attackers.

4.1 Motivation to protect SMTP servers against DoS attacks

Denial of Service (DoS) attack is a hot topic in research, but no single solution can be developed to solve the general problem. 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. 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 e-mails (small bandwidth) is enough to fully load an SMTP server that can lead to a DoS condition. There are a number of possible solutions proposed against DoS situations (e.g., we propose to use traffic level measurements in a DoS front-end). Our goal is to help the understanding why SMTP servers are so much affected with DoS related problems, and why such countermeasures are important.

4.2 Related work on SMTP performance measurements

The available information of SMTP server benchmarking is limited. There are some comparisons on different MTAs carried out by independent works (e.g., [53], [54], [55]). Some of these do not give enough detail to get information about the 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 with measuring SMTP throughput, like [56] or the more sophisticated tools like smtp-source included in Postfix ([60]). 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 [57]), but their main goal is to show how much their product is better, they do not focus on problems like DoS attack. Whole methodologies of performance testing are also available, like the MMB3 benchmarking standard ([47]) for Microsoft Exchange. This methodology is very sophisticated but aims at gathering data on the general behavior of the SMTP server rather than focusing only on SMTP delivery. SMTP delivery is important, because incoming e-mails are under the control of a possible attacker. In my work I focus on SMTP performance 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 requesting the same service from the same server at the same time. Regarding DoS and DDoS attacks, several research papers are available (e.g. [17], [49], [50]), and many other papers mention the hazard of 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 which such attacks can happen.

4.3 DoS of SMTP servers

Several reasons can result in a denial of service in SMTP servers. SYN flood, network traffic flood, and attacks on special programming errors are the most basic DoS attacks against servers. These are general attacks, not specifically designed to harm SMTP servers. 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 using 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 send out too many e-mails in parallel (e.g., electronic newsletters). Sometimes infected computers (e.g., by a virus or an 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 f orces the SMTP server to generate a huge number of non-delivery reports (NDA) [51].

In all these cases, after exceeding a certain load the SMTP server cannot anymore 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 can happen with the SMTP server. If the size of the e-mail queue is large, which may be the case due to undelivered mails, then the periodical attempts to deliver the e-mails from the queue cost a lot of effort (resources). For example in case of Exim ([58]) 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 queuing-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.

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 condition.

4.3.1 Definition of SMTP DoS

Generally, overloading SMTP servers does not necessarily cause a system crash. In fact, the SMTP protocol contains countermeasures against 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.

Definition of a DoS condition only as a list (e.g. with 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 the receiver party. So our definition of the SMTP DoS is as follows: when the delivery process is harmed so much that the legitimate e-mails are affected with a non-tolerable delay.

In my investigation I do not want to identify the exact amount of delay or the actual point when the DoS situation is clearly visible. I try 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 highly exceeds the maximum sustainable traffic, then it can surely cause a bad user experience, or, as we defined sooner, an SMTP DoS.

4.3.2 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. But if we do not 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. However, we do not intend to fully benchmark solutions, we just want to show how vulnerable are these systems against DoS attacks.

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.

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 have become higher than before, due to the excessive using of Bayes databases.

Network performance Since SMTP is an Internet service, network performance is important. Although we will show that a low bandwidth at the server does not mean that a DoS is only possible by depleting bandwidth.

Storage performance Storage (hard disk) performance is one of the highest performance factors in SMTP delivery. Not because single e-mails need 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.

Hardware system and architecture Of course, the whole system architecture, the operating system and the I/O speeds also highly affect the performance, e.g., 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.

Software performance The right software should be applied to the right hardware to obtain optimal SMTP performance. This is true in general, but the decision regarding of an SMTP deployment is generally not based on the performance. SMTP is used because of its off-line properties, its reliability, and its 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 of when we select an MTA software for use.

4.3.3 Benchmark methodology

It is not straightforward to provide a benchmarking methodology that can be fit to all kind of SMTP server implementations, since each and every real life MTA configuration is different. For example, there are different types of hardware in use, the number and the type of the handled domains and the amount of users are 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, neither when the system will go in another phase, nor 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.

4.3.4 MTA software under test

For our performance measurements we selected some of the most widely used MTA software. We must 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 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 are the followings:

- Qmail [59]: We investigate *netqmail 1.05* with factory default settings.
- Exim [58]: The debian *exim4-daemon-heavy (4.50-8sarge2)* package was used in our tests with Maildir delivery format and, in our tests we used the queue_only_load parameter. (as a technical detail: e used it as a stand-alone daemon with a "q1m" queuing parameter)
- Postfix [60]: We took also *Postfix v2.3.6* under investigation.
- Sendmail [61]: *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[62]: We checked also Microsoft Exchange 2003 on a Windows Server 2003.

4.3.5 Hardware environment

My test hardware environment consisted of some basic hardware, which can be a model for the mail server of a small company or academic institute. My 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 OS software I used Debian Linux with kernel 2.6, for commercial software we used Windows Server 2003. I applied only minimal tweaks in the default installations, so I do not expect that this causes a big impact to the validity of my measurements, as my main goal is to identify basic performance against DoS attacks, not a full-scale benchmark test. In Linux environment I did not use any processor or memory consuming software, like X Windows, whereas in the Windows environment the GUI surely consumed some system resources. My test network was a standard 100 Mbps ethernet network with 100 Mbps direct Internet connectivity.

4.3.6 Botnet architecture for testing

For my tests I installed the so called *botnet* architecture. An example for botnet can be seen in Figure 4. The botnets usually work in 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 to the attacker. So an attacker needs to connect to the iRC network, join the sourpecific 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.

My main goal to utilize botnets was to avoid attacking computers being a bottleneck and I chose the botnet solution to coordinate my tests and initiate the attack from multiple computers. I 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. I 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 [63]. I 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.

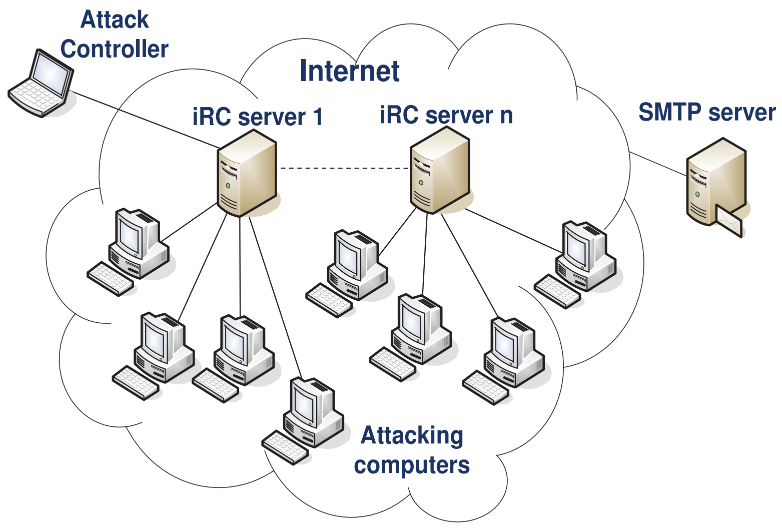


Figure 4: Botnet in general

4.3.7 Phases of the measured process

As I mentioned above, most SMTP servers have two basic modes of operation: Normal operation means 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 my test I caused a high load on the server, and we could observe 3 phases:

- During **Phase 1**, the server receives and delivers e-mails until the load reaches the limit, and the server switches to queuing-only mode.
- During **Phase 2**, the server receives messages and puts them 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 the test into two subgroups. We combined Phase 1 and 2 because Phase 1 only takes some seconds therefore not easily distinguishable. For simplicity in my tables I refer to these subgroups as "Phase 1 & 2" and "Phase 3".

4.4 Results

4.4.1 Generic SMTP throughput

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 6 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 total delivery time is the most interesting. The throughput rate shows that the delivery was constant throughout the delivery process and the system was fully loaded throughout the test.

From Table 6 we can calculate the average throughput of the e-mail server. We show the calculated values in Table 7. 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 8 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 to a very high value 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 6: 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. of average
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 7: Average Throughput - No Content Filtering

MTA	Avg. throughput (e-mail/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

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 9. 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

Table 8: Queuing Behaviour - No Content Filtering

MTA	Time for Phase 1 & 2	Std. dev.	Avg. e-mails delivered during Phase 1 & 2	Std. Dev.	No. of delivered e-mails
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

Table 9: Exim - With And Without Queue_only_load Parameter

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

throughput rates are very close to each other in these two cases, therefore we can see that the queuing properties do not modify the system performance significantly, even the overhead of the queuing process, which is caused by the higher file system activity, cannot be distinguished.

4.4.2 Content filtering

A large number of content filter solutions (virus and spam scanning) are available as commercial and open-source free software. As our resources to carry out investigations were limited, we decided to test only Amavisd-new ([64]), ClamAV ([65]) and Spamassassin ([66]) 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 neighter 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 permanently resides in the memory, client programs can communicate with the clamd daemon through a unix socket. Our results are shown in Table 10.

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 e-mail/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.

Table 10: Content Filtering - Clamscan vs. Clamd

setup	Total time to deliver (s)	Std. dev.	Avg. delivery rate (e-mail/s)	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

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 ([67]), Pyzor ([68]), bayesian filtering, DCC ([69]) and other plugins to distinguish spam and legitimate e-mail (ham).

Our results are shown in Table 11. 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/s. 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 11: Content Filtering - Spamassassin Scenarios

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

4.5 Analysis of results of SMTP performance measurements

We tested the SMTP throughput in our test environment, which can be a model for a basic real-life SMTP server. 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 is meaning, 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. Therefore, 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 cannot be not easily distinguished or filtered from the network. Of course, content analysis can help, but as perfect (no false positives, no false negatives) 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 e-mails/day and to accomplish this with the statistical properties of legitimate users, the attacker will not need too much more than several thousands servers.

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.

4.6 Conclusion on results of SMTP performance measurements

In real-life, SMTP DoS condition is one of the most general problems. Company administrators 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.

I 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 environments 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 attack an SMTP server seriously. An e-mail flow of 1.54 messages / second, that is about 50kb/s can fully load a server if it uses 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 (rule of thumb).

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

4.7 DoS front-end based on traffic measurements

During a Distributed Denial of Service (DDoS) attack the target is an Internet server and the attacker's aim is to exhaust the server's resources in order to prevent authorized access to services or degrade the quality of service. The attacker utilizes more attacking clients and acts organized to multiply the impact of the attack and to avoid being discovered and filtered. (see [17] for a comprehensive description of DDoS attacks).

It is a fairly easy task to deploy a DDoS attack. Various pre-written tools are available on the Internet.

Protection against DDoS attacks highly depends on the model of the network and the type of attack. Although several solutions and methods have been proposed, most of them have weaknesses and fail under certain circumstances.

Protocol reordering and *protocol enhancement* are workable methods to make security protocols more robust and less vulnerable to resource consumption attacks (a few examples are listed in [20] and [18]). This approach tries to make a protocol less vulnerable against a single source of attack, but it is not so effective against a distributed attack.

Stateless protocols eliminate problems that follow from memory overload. However, it is achieved at the cost of transforming the memory overload problem into a network load problem and at the same time the chance for message-replay attacks is also increased. So, stateless protocols can be useful in specific cases, but do not provide a general solution. (A study on the use of stateless protocols against DoS can be found in [70]).

A good starting point against any type of attack could be the one of the initiator identification. There are tools for *tracing the source* of an attack, for example *IP Traceback* (see [71] and [4]). Unfortunately, this approach in a sense contradicts one of the values of the Internet: anonymity. For example a network using onion routing technology provides complete anonymity. Traceback algorithms to identify attackers might turn out to be completely useless in such a network. The goal is to catch the 'real' initiator, not the attacking computer, but this tracing technique can only identify the latter. During a DDoS attack the number of attacking sources can be very high; Identifying and disarming all of them is not a simple task.

Several methods, like *ingress filtering[72], rate control, distributed rate control mechanisms and pushback[73]* make the attacker's work more difficult, but at the same time these tools open up new areas for attacks. If for example MULTOPS (a router technique, see [5]) were widely deployed, it would need too much resources. On the other hand, if it is not widely deployed, its positive effect is very restricted.

Client side puzzle and other *pricing algorithms* (see [74, 75]) are effective tools to make protocols less vulnerable to depletion attacks of processing power, but in case of distributed attacks their effectiveness is an open question.

We provide a simple and robust protection algorithm based on easily accessible information at the server. This is done in such a way that the server host cannot be disabled by an attacker and as soon as the overload disappears, the normal service quality resumes automatically (e.g. without restarting the server). Simultaneously we wanted to minimize the number of legitimate sources blocked by the server as it is typically needed in a real-life scenario. This criterion is not met by many of the current commercial solutions. (see [76] for a review of some commercial solutions) Our approach does not need to modify network elements outside the victim server.

4.8 DDoS front-end module

4.8.1 Traffic model and attack model

In our traffic model "packets" from the network mean small stand-alone queries to the server (e.g. a small HTTP query or an NTP query and answer). For simplicity we assume that every query causes the same workload on the server: by using the appropriate enhancements (protocol enhancements, crypto hardware, caching, ...) on the server the workload of the different queries can be very similar. Every query causes some workload (memory and processor consumption) and thus after a certain level, the server cannot handle the incoming traffic.

The attacker uses A hosts during the attack. When A=1 the attack originates from a single source, while the case of $A > 1$ corresponds to a distributed attack (DDoS) (see [17]). There are one or more human attackers behind the attacking sources. These attacking sources are machines on the net controlled (taken over) by the attacker for the purpose of the attack. We assume that the attacking machines use real addresses, consequently they can establish normal two way communication with the server, like a host of any legal client. The human attacker hides well behind the attacking machines in the network, which means that after carrying out the attack and after removal of all compromising traces of attack on the occupied machines, there is no way to find a trace leading to him.

Two types of sources are distinguished: legal sources and attacking sources. There are $N(t)$ legal sources and $A(t)$ attacking sources in time slot t. In our model the attacker can reach his aim only if the level of attacking traffic is high enough compared to the level under normal operation. It is assumed, that the attacker can control the extra traffic by changing the number of attacking machines and the traffic generated by these machines. In this respect we assume a powerful attacker; he can distribute the total attacking traffic among attacking machines at his choice. We assume that the reason for using several attacking machines is to make it more difficult for the server to identify and foil them all. Note, however, when more attacking machines are used by the attacker it becomes more difficult for him to hide.

Therefore, we assume that the attacker limits the number of attacking hosts ($A(t)$ is low), and thus, the traffic intensity from the attacking sources is higher than the normal.

In fact, a trade-off can be identified between the ability to hide and the efficiency of the attack.

On the other hand, we should clearly state the limits of the protection based on traffic level measurements. If the attacking source generates traffic that is indistinguishable by statistical means from the normal traffic, then there cannot exist any algorithm which successfully protects the target. That practically means that if the attacker generates packets that look the same as normal packets (in format and in payload), generates queries with the same probabilities - statistics (including seasonality, etc.) as legal users do, and every IP source in the attack uses the same (or lower) traffic intensity as the legal sources, then there is no reason to decide if a query is part of the attack and therefore any protection method that uses traffic analysis might give no better results than random guessing.

This is why the trade-off is important: Due to the trade-off, we can expect, that the traffic intensity from the attacker is higher, than the traffic intensity from a legal source.

4.8.2 The module

A DDoS front-end module is attached to the server from the network side. (The front-end can be a software component of the server, a special hardware in the server or an autonomous hardware equipment attached to the server)

The front-end and the server together constitute a virtual server. The incoming traffic enters a FIFO buffer[2]. Discrete time model is assumed, i.e. the time is slotted. Traffic is modelled and processed per time slot. The server empties μ storage units per time slot from the buffer. Because the buffer is fed by random traffic, there is a positive probability of the event that overflow occurs. When a DDoS attack begins, the incoming traffic quickly increases and the buffer becomes full. Most of the incoming units (packets) will be dropped, so the attacker achieves his aim of significantly degrading the service quality, essentially cutting the connection to the server. However, the server host will not be disabled and remains intact, only its services cannot be used. The goal of the front-end module is to try to suppress the traffic from the attacking sources effectively. (Similar method is used in [77] against SYN attacks)

Assume that there are two states of the incoming channel: normal state when there is no DDoS attack and attack state when the server is under attack. The attack begins at time t^* (time is measured in time slots). When at time $t^* + \delta$ the front-end buffer becomes full, the transport protocols run by legal and attacking hosts detect that no (or very rare) acknowledgments arrive from the server and they stuck in repeated transmissions until the timeout of the connection. The first task is to detect the beginning of an attack, and estimate time t^* as close as possible.

The next task is to identify the sources of the attack and to suppress their traffic. The identification can be based on statistical properties of the traffic flow. The front-end can identify all active sources, can measure the traffic generated by these sources and can classify them into specific sets. Note that in order to get reliable measurements for traffic level, we have to carry out these measurements in time slots between t^* and $t^* + \delta$. Consequently, the effectiveness of such protection is strongly affected by the time duration δ, during which the traffic flow is "undistorted" between the server and the sources. Therefore we should try to lengthen time duration δ to gain more time for traffic measurements. The only way seems to be the use of a huge buffer. Assume therefore, that the buffer length is $L = L_1 + L_2$, where:

[2]In practice the buffer can be a set of httpd child processes dealing with incoming queries.

length L_1 is designed to serve the normal state, assumed to be chosen according to the service rate μ and the accepted probability of accepted loss (loss is an event when incoming units are dropped because the buffer is full),

length L_2 corresponds to the excess length, with a purpose to gain enough time for measurements during the start-up phase of the attack.

For simplicity, we assume, that the system has not been attacked for a long time, and the attack begins at time t^*, which means that all attacking sources start emitting packets from this time: the network is in normal state for $t < t^*$ and turns into attacked state in time t^*. Let $\hat{t}$ denote our estimate on t^*.

Furthermore we make the simplifying assumption, that the set of active sources is constant during the attack.

$T_n(t)$ denotes the aggregate traffic from the legal sources (normal traffic) and $T_a(t)$ denotes the aggregate of attacking traffic. Let the corresponding expected values (per time slot) be denoted by λ_n and λ_a, respectively, i.e.

$$E(T_n(t)) = \lambda_n, E(T_a(t)) = \lambda_a \tag{17}$$

Similarly let the corresponding standard deviations be denoted by σ_a and σ_n. Let Q denote the a priori unknown ratio between averages λ_a and λ_n, i.e $Q = \lambda_a/\lambda_n$.

Because typically the beginning of the attack (t^*) is earlier than the time of its detection ($\hat{t}$), we waste precious time from efficient traffic measurements. To minimize this loss, we estimate the aggregate traffic level continuously using sliding window estimates. The front-end handles two sliding time windows, a longer (with capacity of w_ℓ) and a shorter one (with capacity of w_s slots). In this way we measure both a long time average level, $\bar{\lambda}(t)$ and a short time average level $\hat{\lambda}(t)$, of incoming aggregate traffic per slot at time slot t.

4.8.3 Algorithms of the module

The algorithm of protection consists of the following sub-algorithms, which are executed consecutively:

A.) detection of the attack

B.) identification of the attacking sources

C.) suppression of the attacking traffic

D.) checking of the success of the suppression

A.) Detection Of The Attack

An early detection of the attack is a vital point of the protection. (In paper [78] wavelet methods are used for DDoS attack detection). In our approach, we define three simple and robust algorithms for the detection of the DDoS attacks:

Algorithm A1.

The beginning of the attack is decided to be time $\hat{t}$, which is the time, when the following event occurs:

Event 1: The buffer length exceeds L_1

Algorithm A2.

The beginning of the attack is decided to be time $\hat{t}$, which is the time, the following event occurs:
Event 2:

$$\hat{\lambda}(\hat{t}) > (1+r)\bar{\lambda}(\hat{t}) \tag{18}$$

where r , $r > 0$ is a design parameter.

Algorithm A3.

The beginning of the attack is decided to be time $\hat{t}$, which is the time, when the earlier of the two events Event 1 and Event 2 occurs.

B.) Identification Of The Attacking Sources

We would like to suppress the aggregate traffic selectively at the input (front-end) by suppressing the attacking traffic as effectively as we can. Here the problem arises of distinguishing the traffic coming from attacking sources. The assumed distinguishing characteristic of the attacking sources is the higher mean of their traffic level. It is also assumed that the front-end device can measure the traffic characteristics of all active sources distinguishing them by their network address.
Starting at time $\hat{t}$, we measure the aggregate and the individual traffic levels (i.e. traffic per source). If we correctly identified an attack, i.e. $t^* < \hat{t} < t^* + \delta$, we can make measurements over the resting time $[\hat{t}; t^* + \delta]$.
Let the output of this measurement be denoted by $\hat{\lambda}_r(t^* + \delta)$ and $\hat{\lambda}^{(i)}(t^* + \delta)$ for the aggregate level and for source i respectively.
As we cannot determine the exact traffic from the legal sources during the attack we use $\bar{\lambda}(\hat{t} - c)$ $(c > 0)$ as an estimate on the mean aggregate traffic level of legal sources in time interval $[t^*, t^* + \delta]$ and we get an estimate

$$\hat{\lambda}_a = \hat{\lambda}_r(t^* + \delta) - \bar{\lambda}(\hat{t} - c) \tag{19}$$

on the mean aggregate traffic level of attacking sources. The set Z of active sources is decomposed into

$$Z = Z_n \cup Z_a \quad (Z_n \cap Z_a = \emptyset) \tag{20}$$

where Z_n and Z_a are the sets of legal sources and attacking sources respectively. The identification algorithm outputs a subset Z_a^* of Z. This set should correspond to Z_a as closely as possible.
The identification of attacking sources is made by the following algorithm:

Algorithm B1.

Find the maximum-sized subset Z_a^*=$\{i_1, i_2, ..., i_v\}$ of Z, which corresponds to sources with the highest measured traffic levels such that

$$\sum_{j=1}^{v} \hat{\lambda}^{(i_j)}(t^* + \delta) \leq \hat{\lambda}_a \tag{21}$$

The base of this method in our model is the predicate described in the beginning of this section: The attacker tries to hide himself and therefore limits the number of attacking sources ($A(t)$) which is in trade-off with the volume of the attack. As a result of this trade-off the traffic volume from the attacking sources is higher than the traffic from the legitimate clients.

Algorithm B2.

Omit those sources from set Z which have been active at time $(\hat{t} - c)$, $c > 0$, and use Algorithm B1.

C.) Suppression Of The Attacking Traffic

Once we have successfully identified the attacking sources, the traffic suppression algorithm is straightforward:

Algorithm C.

Discard all incoming units with sources from the set Z_a^*.
First, we apply filter rules to discard any incoming packets from the identified sources (needs memory), and then, we have to delete any previously stored packets in the front-end buffer.

D.) Checking Of The Success Of Suppression

Algorithm D.

In case of successful intervention, by running *Algorithm C* the buffer length has to retire below length L_1 within a timeout *t_out*. If it does not occur we should discard packets from further active sources beginning with the one with the highest measured traffic level, followed by a new checking of success step. These steps are repeated until the desired decrease in the queue length is reached.
A conservative estimate on timeout *t_out* can be the following:

$$t\_out = d \cdot \frac{L_2}{\mu - \bar{\lambda}(\hat{t} - c)} \tag{22}$$

where d is a small positive integer (e.g. d=3).

The quality of protection is determined by the
1.) reliability of attack detection,
2.) probability of false identification of attacking and legal sources,
3.) time duration of the unavailability of the service caused by buffer overflow.

4.9 Analysis of the algorithms

4.9.1 Reliability Of Attack Detection

In time slot t we test the following hypothesis about the state of the system

H_0: state of no attack

H_1: state of attack

There are two types of error: Type I error is the event of a missed detection, when our server is under attack and we failed to detect it, while Type II error is the event of a false detection, when an attack is detected and there is no attack.

Consequently Type I and Type II errors are the following:

$$P_I = P(\{\textit{decision on } H_0 \textit{ at time } t\}|H_1)(=0), \tag{23}$$

$$P_{II} = P(\{\textit{decision on } H_1 \textit{ at time } t\}|H_0). \tag{24}$$

We assumed that the system has been in normal state for a long time, when a change of state occurs. Considering detection algorithm A3 we can give the following rough upper bound on P_{II}

$$\begin{aligned} P_{II,A3} &= P(\{\textit{queue length} \geq L_1\} \cup \\ &\qquad \{\hat{\lambda}(\hat{t}) > (1+r) \cdot \bar{\lambda}(\hat{t})\}|H_0) \qquad (25) \\ &\leq 2 \cdot max\{P_{II,A1}, P_{II,A2}\} \qquad (26) \end{aligned}$$

where the individual probabilities for the Algorithms A1 and A2 are the following:

$$P_{II,A1} = P(\{\textit{queue length} \geq L_1\}|H_0) \tag{27}$$

$$P_{II,A2} = P(\{\hat{\lambda}(\hat{t}) > (1+r) \cdot \bar{\lambda}(\hat{t})\}|H_0) \tag{28}$$

Probability $P_{II,A1}$ can be calculated (designed) using approaches of standard buffer design (any corresponding textbook covers the needed techniques, see e.g. [79]). Probability $P_{II,A2}$ is considered below, where we give an upper bound on it.

Theorem 1. *Assume that random variables $T_n(t)$, t=1,2,..., describing the aggregate traffic of legal sources are pairwise uncorrelated. The probability that the short time window measurement with window length w_s results in a value exceeding r times the mean traffic of normal state, assumed H_0 is true, can be upper bounded by*

$$P\{\hat{\lambda}(\hat{t}) > (1+r) \cdot \bar{\lambda}(\hat{t})\} \leq \frac{1}{w_s r^2} \left(\frac{\sigma_n}{\lambda_n}\right)^2 \tag{29}$$

Proof: Let

$$\eta = \frac{1}{w_s} \sum_{m=1}^{w_s} \xi(m) \tag{30}$$

where

$$\xi(m) = T_n(t_m), m = 1, ..., w_s \tag{31}$$

are nonnegative, pairwise uncorrelated random variables.
Applying Chebysev's inequality:

$$\begin{aligned} P[\eta > (1+r)E(\eta)] &= P[\eta - E(\eta) > rE(\eta)] \leq \\ &\leq P[|\eta - E(\eta)| > rE(\eta)] \\ &\leq \frac{1}{r^2}\left(\frac{\sigma_\eta}{E(\eta)}\right)^2 \end{aligned} \tag{32}$$

where

$$E(\eta) = E(\xi(m)) = \lambda_n, \quad \sigma_\eta^2 = \frac{1}{w_s}\sigma_\xi^2. \tag{33}$$

Bound (10) decreases with the inverse of the window length, w_s. In the case of Poisson distribution, the probability of the event that the short time window average exceeds the double of the normal traffic (i.e. $r = 1$) is upper bounded by $1/(w_s\lambda_n)$ $((\sigma_n)^2 = \lambda_n)$. ♣

The upper bound presented above can be used for setting the parameters of a real-life system. Although this upper bound can be very loose, it might be useful as not too much information is required on the traffic. If we design a system with $\lambda_n = 1$ (one expected event in each second), and we want to bound the false detection (type II) probability below $1/10^5$ (about 1 false detection per day), and we expect the normal traffic has a Poisson distribution, we can design the parameters such as $r = 10$,$w_s = 1000$. This estimate is not very useful in real-life environment, as the detection defined by $r = 10$ is far less valuable than we expect in this scenario, although it can be used as a basic constraint or a fallback parameter for our system.

A stronger upper bound can be found by using the Hoeffding bounding technique [80] if we assume the stronger condition of independence of the aggregate traffic of legal sources. In this case the corresponding bound decreases exponentially with the window length, w_s. Assume that the peak aggregate traffic level in the normal state can be bounded by a constant K. The probability that the short time window measurement results in a value exceeding r times the mean traffic of normal state, assumed H_0 is true, can be upper bounded.

The basic Hoeffding formula says that if $X_1, ..., X_{w_s}$ are independent random variables and

$$P(X_i \in [a_i, b_i]) = 1; \qquad 1 \leq i \leq w_s \tag{34}$$

then

$$P(S - E[S] \geq w_s \cdot t) \leq \exp\left(-\frac{2w_s^2 t^2}{\sum_{i=1}^{w_s}(b_i - a_i)^2}\right) \tag{35}$$

In our case $S = w_s * \eta = \sum_{i=1}^{w_s} \xi(i); E(\xi(i)) = \lambda_n$. The constant K bound means that $P(0 \leq \xi(i) \leq K$ for every $i) \leq 1 - \epsilon$.

Therefore in our case

$$\begin{aligned} P(\eta \geq (1+r) * E[\eta]) &= P(\eta - E(\eta) \geq r * E(\eta)) && (36)\\ &= P(w_s * \eta - E(w_s * \eta) \geq w_s * r * E(\eta)) && (37)\\ &\leq \exp(-\frac{2w_s^2(r * E(\eta))}{K^2}) && (38) \end{aligned}$$

We get the result

$$P\{\hat{\lambda}(\hat{t}) > (1+r) \cdot \bar{\lambda}(\hat{t})\} \leq \exp\left(-2w_s\left(\frac{r * \lambda_n}{K}\right)^2\right) \tag{39}$$

The upper bound based on Hoeffding technique can be useful for designing of the parameters of a real-life system. In this case with the previously mentioned $\lambda_n = 1$ parameter and a $K = 10$ bound, the setting $r = 2$ and $w_s = 150$ results in an upper limit of about 1/150000. This probability is better than in the case of the above described environment and still the windows size and the expected traffic jump is much closer to an expected result. In this way the parameters of the front-end module can be therefore more aligned to the system properties, and the results prove better properties than in the previous case. Although, this approach needs more information about the traffic, therefore it cannot be used in all cases. If a more appropriate design of parameters is necessary to a particular system, burn-in tests and simulations can be used to set the parameters.

4.9.2 The Probability Of False Identification And The Problem Of Hiding

The identification algorithm outputs a subset Z_a^* of set Z of all active sources. The case when $Z_a^* \subset Z_a$ means the attacking sources are not fully identified.

The attacker wishes to hide as effectively as he can. Assume the attacker deploys the same aggregate attacking traffic independently of the number of attacking sources, $A(t)$, i.e. the use of more attacking machines means the generation of less traffic per machine. The attack is more severe when the total attacking traffic is distributed among more attacking sources. There are two extreme cases in the number of attacking sources.

Consider first the case of $A = 1$. Identification error occurs if event:

$$\max_{i \in Z_n} \hat{\lambda}_n^i > \hat{\lambda}_a^j \tag{40}$$

where $Z_a = \{j\}$ (i.e. there will be at least one misidentified source).

Let $F_n^{(i)}(k)$, and $F_a(k)$, $k = 0, 1, 2, ...$ denote the probability distribution functions of the traffic corresponding to the legal source with index i and the attacking source, respectively. The probability of event 40 is

$$P(event(40)) = \sum_{k=0}^{\infty} \left(1 - \prod_{i=1}^{|Z_n|} F_n^{(i)}(k)\right) (F_a(k) - F_a(k-1)) \tag{41}$$

where $F_a(-1) = 0$.

The other extreme case is when the number of attackers and legal users are equal and all active sources emit the same traffic level. In this case, a legal user will be identified attacking with the same chance as an attacking source. The decision cannot be better than a random selection from set Z. For instance, the probability that there will be no misidentified source is practically zero.

Therefore the best strategy of the attacker is to spread uniformly the attacking traffic among as many sources as he can. If the attacker has a good estimate on parameters $\hat{\lambda}_n$, $\hat{\lambda}_n^{(i)}$ and μ, i.e. the aggregate level and the per source traffic level of legal users and the service rate of the server, then the most powerful attack deploys A attacking sources where

$$A = \frac{\mu - \hat{\lambda}_n}{\min\limits_i \hat{\lambda}_n^{(i)}} \tag{42}$$

Time Duration Of Unavailability Of The Service

If the identification of attacking sources is perfect, the suppression of attacking traffic frees the buffer from overload at once because all packets from attacking sources are discarded.

Therefore the time of unavailability of the service is determined by the eventual reiteration of the identification and suppression step. This complex mechanism does not seems tractable for an analysis, it is better suited for simulation.

4.10 Attack detection with predefined error probability

For the original attack detection algorithms (Algorithms A1,A2,A3), a possible extension is described in this section, which can be used in real-life environments, especially if the network traffic cannot be modeled properly.

Depending on the situation, the above defined attack detection algorithm (Algorithms $A1$ and $A2$) can generate a number of false-positive alarms, which is analyzed in Section 4.9 in general form and some insight was also given how the network traffic can be handled with Poisson model. While, for example, the Poisson model is generally considered as a workable model in the field of modeling Internet traffic [48], in real-life examples these mathematical models might not work.

In some cases (specifal form of the probability distribution of the traffic intensities), the error ratio, the number of false positives can be very high. In real-life situation this can mean the following: whenever our system identifies an attack, automatic (fast-reaction) and administrative (reconfiguration, slow reaction) countermeasures can be activated. To avoid problems caused by false positives, administrative personnel can furthermore analyze the situation after an alarm is received, and if the personnel identifies the traffic as normal (and the alarm as a false-positive), the countermeasure (filtering of the traffic) can be withdrawn.

This means, that a false alarm will use up significant amount of workforce to correctly handle the situation, or cause problems in the network because of the filtering of legitimate traffic. If the number of such false alarms is too high, then most likely the administrative personnel will simply ignore the alarm and will not handle the problem correctly. Any protection method that uses traffic level measurements will have a certain level of false detections, but it is very important to be able to set a limit on these false positives in practical applications, even if the number of false negatives is affected by the modifications.

When there is not enough information about the network traffic to be able to set the parameters to reach the given false-positive ratio for the attack detection algorithm, but we can consider that the network traffic intensity distribution is stationary, the following detection method can be proposed:

- The system manager sets a P_M limit for the probability of false positives. (e.g. in practice: 5 false alarms each week can be accepted)
- The DoS front-end collects and stores statistical information on the normal traffic.
- The traffic intensity distribution is calculated from the stored data
- A maximum traffic intensity M is calculated from the intensity distribution, where M is the value where the probability of the tail of the intensity distribution (above M) equals P_M.

After this initialization (self-learning) process, the Attack Detection A4 is defined as follows.
An attack is detected, if

$$\hat{\lambda}(\hat{t}) > M \tag{43}$$

In this case, due to the definition of the algorithm, the average probability of false positives will be exactly P_M, therefore from the system administration point of view it is easy to plan the necessary human resources.

On the other hand, this simple algorithm does not give information about false negative probability and success ratio. However, from the administration point of view, this information might be not so important. The main goal in the administration of the system is to maintain the dependability of the system, therefore, the administrator is mainly interested in such attacks, where the attacker is able to degrade system performance. This means, that if there is an attack (by some definition), but the attack is not powerful enough to degrade system performance, then it might be not a big problem, if the attack detection algorithm does not detect this attack.

We can consider safe and unsafe systems from the dependability point of view:

- Safe system: If M, and/or the end of the tail of the normal intensity distribution is far less than the maximum intensity that the system can handle without performance degradation. In this case, false negatives also do not affect the dependability of the system.
- Unsafe system: If M is above the maximum intensity that the system can handle. In this case, the system is unable to handle the traffic load sometimes even under normal conditions, therefore false positives can also mean serious problems in the system, as the dependability is affected by both attackers and normal traffic.

Of course, not every attack has a constant intensity, and no matter if a system can handle the additional traffic caused by an attack, it surely creates delays for the legitimate e-mails. Therefore this categorization is only giving a basic view on the dependability of the system in case of attacks.

An important issue with the proposed method is the learning mechanism of the e-mail traffic characteristics. If this traffic is not stationary, or cannot be divided into stationary parts, then the proposed solution is infeasible. A basic analysis of real-life e-mail traffic characteristics can be found in Annex I.

4.11 Simulation results

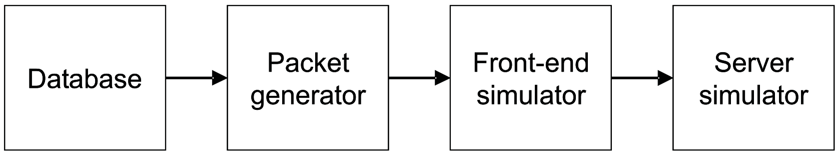


Figure 5: The block scheme of the simulation

The simulation consisted of the following elements shown in Figure 5.

In addition to scheduling and buffering arrived packets, the front-end simulator collects statistical data based on the traffic, detected attacks, identified attacking sources and suppressed attacking sources.

The simulation was written in C and carried out on a PC running Linux. A MySQL database was used for storing all the data, thus our tests are fully reproducible. The packets arrived in discrete time intervals, the time resolution was set to 10^{-6} seconds. Statistics (e.g. intensity of the traffic) were gathered every second. The simulation starts at 0 second, most of the calculations are carried out only once a second. The simulation begins with 100 seconds of normal traffic to initialize the DoS front end. The attack begins after the 100-th second and lasts until the 200-th second, the simulation ends with another 100 seconds of normal traffic to gain data about the recovery of the system.

4.11.1 Parameter setting

The arrivals of the various packets at the DDoS front-end are modelled by a Poisson process. These packets are stored in a buffer, and every packet is served by one virtual processor. The queue is type M/M/1, the inter-arrival times and the service times are assumed to have exponential distributions. The number of the sources in this simulation is constant in time.

The Poisson model might be judged as oversimplified for the typical Internet traffic. In this regard the aim of this first simulation effort was to see the algorithms in work and to get feedback on their basic strength and weaknesses, and the sensitivity of the parameters. At the same time, we believe that such a model might be suited to some Internet protocols if the parameters are correctly set. ICMP/Ping, NTP (Network Time Protocol) and DNS clients send many small uniform-sized packets with uniformly distributed inter-send times, which makes them more similar to our simulation than other protocols (e.g. HTTP or FTP).

The number of simultaneous legitimate clients can also vary in a broad range. Some examples for different hosts can be imagined: For a small corporate server the number of legal clients is low ($N(t)$=5) the server's capacity is high, the average load is low, and thus the number of attacking hosts should be relatively high: e.g. $A(t)$=40, so $N(t) << A(t)$. For a medium size server $N(t)$=50 and a successful attacker can deploy it's attack from a low number of hosts, e.g. $A(t)$=50, so it is easy to hide his attack ($N(t) \sim A(t)$). For a global portal there are lots of legal clients: $N(t)$=10000 and the attacker cannot easily estimate the needed number of attacking machines, so he can use $A(t)$=5000 attacking hosts with a high attacking rate (e.g. λ_a=λ_n·10) ($N(t) > A(t)$)

In our first simulation we show that our method is workable on a large number of hosts, so we decided to set the parameters as described in Table 12.

Table 12: Basic parameters of the simulation

Number of legal sources $N(t)$	10000
Number of attacker sources $A(t)$	5000
λ for legal sources ($\lambda_n^{(i)}$)	0.1
λ for attacking sources ($\lambda_a^{(i)}$)	0.4
Service rate (μ) (packets/sec)	1500

In our case, described in Table 12, it is trivial that the server's capacity should be more than 1000, but the attack is only successful when the service rate μ) is below 3000 ($\lambda_a^{(i)} \cdot A(t) + \lambda_n^{(i)} \cdot N(t)$). We consider $\mu = 1500$.

The average number of packets in the buffer during normal state is

$$E(X) = \frac{(\lambda_n/\mu)}{1 - (\lambda_n/\mu)} \tag{44}$$

which gives $E(X)$=2 for data from Table 12 (see [79] for details). We set the buffer size parameter L_1=40 (packets). The other parameter that affects the length of the buffers is the overflow traffic. As the normal traffic is about 1000 packet/second, length $L_2 = 30000$ (packets) seems a safe estimate. The parameters of the algorithm for detection of attack are set as shown in Table 13.

Table 13: Attacker detection parameters

Sliding window size (w$_s$)	10 sec
Tolerance for traffic jump (r)	0.6
Time frame to get last correct value of λ	45 sec

The available time for traffic measurements depends on the value of δ. This a priori unknown parameter depends on how long it takes for the buffers to fill up.

In our simulation we set a constant limit ($\hat{\delta} \leq \delta$) for traffic measurements. Suppose we know that the total traffic (with attackers) is $T_n + T_a = 3000$, and the service rate is $\mu = 1500$. We can expect the buffer (L_1) to be full after $40/(3000-1500) \approx 0.3$ seconds, and the whole buffer (L) after $30040/(3000-1500) \approx 20$ seconds (so $E(\delta) \approx 20$). It is a very safe estimation that $\delta = 10$. In real-world situations we have no preliminary knowledge about the attack and so δ (which is coherent with the L_2 buffer length) is the result of some estimation or adaptation. For simplicity, we set the short time moving average window size as $w_s = \delta$. We consider that the normal traffic is restored if the buffer length decreased under L_1.

Algorithm B1 was used during attacker identification.

4.11.2 Simulation 1

In this simulation our main goal was to gain data about the viability of our approach. Table 14 shows the most interesting results of the simulation. After running the simulation with $\hat{\delta} = 10$ we repeated the simulation with different window sizes. Table 14 shows clearly that a longer window size gives more accurate identification: the number of filtered legal clients decreased to 71 from 601 as the windows size increased from 5 to 40 seconds.

Table 14: First simulation results

$\hat{\delta}(\hat{\delta} = w_s)$	5	**10**	20	30	40
Correctly identified attackers	2963 (59,26%)	**3737 (74,74%)**	4497 (89,94%)	4750 (95%)	4875 (97,5%)
Filtered legal clients (type II error)	601	**562**	285	148	71
Dropped packets	0	**0**	0	14361	29326
Maximum buffer level (and corresponding time frame)	29713 (200s)	**14871 (110s)**	29654 (119s)	30040 (120s)	30040 (120s)
Time to restore (after t^*)	156	**111**	79	77	83

On the other hand a large window size endangers the system with the possibility of a buffer overflow due to the slower detection. During the simulated attack the buffer could only allow traffic measures for 20 seconds, after this the buffer filled up and after this the identification algorithm produced less accurate results.

The numbers in the maximum buffer level row show that when the time window is too short, our algorithms cannot determine the correct level of the attack. Therefore a too low number of hosts were filtered at $w_s = 5$ and in this way the maximum buffer level reached a very high level at a later time. The simulation results show that our method can successfully protect the system with a good estimation of the parameters, and when enough buffers are available for measuring the traffic levels.

4.11.3 Simulation 2

During the second simulation we simulated a smaller system: The sample system consists of $N = 50$ legal and $A = 50$ attacker clients. Using $\lambda_n = 0.1$ and $\lambda_a = 0.2$ the task of the identification algorithm is hardened. The service rate $\mu = 8$ while the buffer lengths are $L_1 = 40$ and $L_2 = 160$. The window size parameter was considered as $w_s = \delta = 10$. Other parameters and used algorithms are the same. During Simulation 2 we repeated our tests on 500 different sets of input data to gain statistical information about the properties.

After running the simulation 500 times on different data sets we found that the attack was firstly detected by Algorithm A1 in 4 cases, Algorithm A2 was faster in 454 cases while the attack has been detected by both Algorithm A1 and A2 in 42 cases. The simulation results are summarized in Table 15.

The mean time of attack detection is $E(\hat{t} - t^*) = 3$ seconds. After this time the DDoS front-end can start to suppress the attacking traffic. The minimum detection time is $(\hat{t} - t^*)_{min} = 0$ seconds, the maximum

Table 15: Simulation results

	minimum	average	confidence interval (95%)
traffic restoration time (after t^*)	51	116.624	1.8671
packets dropped	0	0.708	0.312
normal user filtered (error type II)	1	7.228	0.223
number of attackers filtered	24	34.102	0.286
attack detection time (after t^*)	0	3.07	0.09

is $(\hat{t} - t^*)_{max} = 6$ seconds.

4.11.4 Selection of parameter r

We carried out investigations on how the number of successful detections by different algorithms depends on the value of r (i.e. tolerance for traffic jump).

Table 16: The effect of parameter r on the first detection – aggregate no. of events from 500 tests

r	Alg. A1	Alg. A2	Alg. A1+A2 same time	A1 error type II	A2 error type II
0.2	0	500	0	0	1117
0.3	0	499	1	0	234
0.4	0	495	5	0	37
0.5	0	485	15	1	9
0.6	4	454	42	0	1
0.7	18	375	107	0	2
0.8	54	273	173	0	0
0.9	138	176	186	0	1
1.0	227	106	167	1	0

The first 3 columns of Table 16 show which algorithm detected the attack firstly. Other columns show Type II error events in case of *Algorithm A1* and *A2*. The length of L_1 was set to fit to the normal traffic, so the number of type II errors by *Algorithm A1* are very low. In the range of $r \leq 0.4$ the number of false detections in case of *Algorithm A2* is too high.

The high number of attack detections by *Algorithm A2* in this area shows that the algorithm detects the attack correctly, but signals the attack in a too early stage. In this stage the success checking algorithm

(*Alg. D*) might report the end of the attack because of the low number of packets in the buffers. Due to the lack of attacker identifications the attack detection algorithm detects the attack again. Meanwhile the traffic measurements do not reflect the correct values, because the timeframe measures contain more data from the (not fully detected) attack stage too. This makes our solution ineffective. The optimal detection would detect every attack once and identify all the attacking sources.

When r is between 0.5 and 0.9 then we can see that the number of attack detection by *Algorithm A2* decreases and for *Algorithm A1* slowly rises. For *Algorithm A2* high values of r cause that we are not detecting the attack by a traffic jump, but rather by a buffer overrun.

4.12 Prototype application

To show the applicability of our proposal we designed an architecture that uses our proposed DoS front-end module to protect a real-life environment. Our candidate is the SMTP system with real-time virus scanning.

The proposed architecture is shown in Figure 6. The basic SMTP system is extended with a general TCP wrapper, a DoS front-end client and a DoS front-end engine. The TCP wrapper is a simple wrapper that first asks the DoS front-end if the sender is permitted to send e-mails. After that it simply forwards the data through the connection towards the mail transport agent (MTA). The functionality of the DoS front-end is divided into two parts. The DoS front-end client is a simple application, it's duty is to communicate with the DoS front-end engine and depending on the decision of the engine, send the result back to the wrapper application. The DoS front-end engine consists of a statistical engine that measures the traffic and maintains the state-variables of the DoS front-end and a decision engine that decides if a sender should be filtered out preventing to send any e-mails.

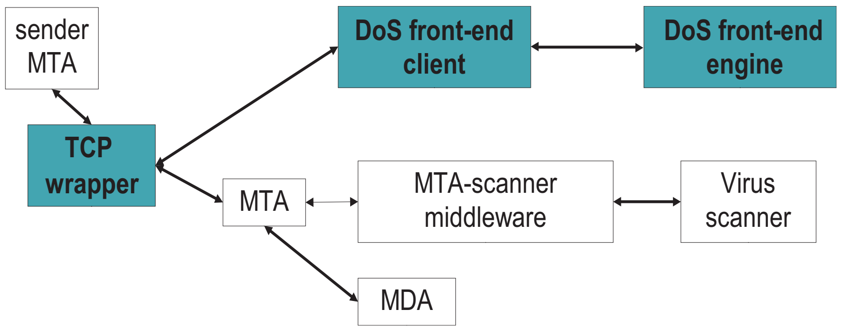


Figure 6: Prototype topology

We used the proposed architecture to design and develop a prototype system. Our prototype system consists of an exim MTA, an Amavis scanning engine, and a virus scanner. If our system considers a state as an attack state, it simply sends back a regular SMTP temporary error to the sender. If the system decision is a false detection (type II error), then the sender is not able to send the message immediately, but due to the way SMTP works, after some time-out the sender's SMTP server will try to send the message again. This way we can set an efficient protection against the attackers without harming the regular traffic (as we do not filter all the traffic only the attacking traffic), and we do not cause intolerable problems even if the decision is a false-positive (type II error). Our prototype is currently fully functional and can be used to protect an STMP server in a real-life environment.

4.13 Summary

I made performance measurements of SMTP content filters to analyze the possibility of DoS attacks. The results of empirical experiments on SMTP performance is presented in the first part of this section. Measuring SMTP performance is not a simple task, as the delivery process is complicated and different scenarios are not easily comparable.

The 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, which is about 50kb/s can fully load a server if it is using content filtering software. The results also emphasize the vulnerability against sophisticated DoS attacks, where the usage of statistical analysis against attacks is limited.

I developed a model to handle some special cases of DDoS attacks at the victim server. I identified a trade-off at the attacker which makes the protection by traffic analysis possible. I presented a simplified, but applicable model of this problem, and described all the methods and algorithms needed to successfully deploy such protection. The proposed approach does not need to modify network elements outside the victim server and minimizes the number of legitimate sources blocked by the server.

I gave analysis on the proposed method based on traffic analysis and I also show how the possibility of false positive and false negative errors can be computed. These results can be used to design the parameters of the system. The effectiveness of my proposed method is also supported by simulation, and this also helped to further investigate the effect of the parameters of the proposed method.

I also designed an architecture to show how my proposed system can be inserted in an SMTP environment. Based on this architecture I designed and developed a prototype of my proposed protection method, which is based on traffic level measurements. The developed software components were tested in real-life systems and confirm that the protection method is workable.

5 Protection against DHA attacks

As it can be seen from the previous section, the protection against a DoS attack is very difficult. The best countermeasure against any type of attack is to prevent the attack. For instance, the problem of spam can be alleviated by preventing spammers from collecting e-mail addresses on a large scale using automated tools such as a Directory Harvest Attack. DHA is typically used to collect e-mail addresses for spamming, and at the same time, it can lead to a DoS condition by itself. In the first part of this section I give analyse the DHA problem and give countermeasure against it. This gives us some prevention against spam, and spam related problems, but also protects against the DoS condition caused by the DHA attack itself.

The most typical DoS attack today is a distributed attack from a number of infected computers, zombies. If we can help the identification of the owners of such computers, and the removal of the malicious code installed onto those computers, we can also prevent such DoS attacks or at least lower the number of the possible attacking sources.

5.1 Motivation

The e-mail Directory Harvest Attack (DHA) is a special form of brute force attack. The attacker's goal is to gain information about the e-mail addresses used in a domain name. The attack itself is not harmful to the attacked domain (except when the volume of the attack is high enough to be considered as a Denial of Service attack). The secondary effect of a successful DHA is that the e-mail address gets inserted into a bulk e-mail address list and spam starts flooding the identified user.

The DHA problem is not elaborated well publicly, although most commercial spam solutions state that they deal with the threat. We analyze some interesting parts of the problem area and meanwhile we also present an efficient countermeasure. The first part of the section analyzes the attacker's possibilities and the economy of the attack. Our observations show that the attackers generally use a dictionary (wordlist) of a fixed size to attack different domains during a DHA. We will show that the optimal wordlist should be different to every domain according to the number of users expected in that domain. We also present proof that our algorithm is optimal. For countermeasure we recommend a centralized protection method, where a DHA RBL (real-time blacklist) server gets information about the hosts sending e-mails to unknown addresses. Beside the method we also present information about our prototype for this protection method.

5.2 Related work

On-line brute force attacks where the goal is to identify a valid username/password pair have been known for a long time. The oldest attacks were aimed at telnet and ftp accounts. Lately, attacks are deployed against SSH, POP3, RAS, Samba, and various HTTP based services.

These attacks show similarity to DHA attacks, as the attacker's goal is to identify valid data and filter out the rest. Active countermeasure is possible for both problems if the attack is on-line: the invalid access / harvest trials can be detected and therefore the attackers can be rejected.

The typical countermeasure for brute force authorization trials is locking of the affected accounts, but this can result in a Denial of Service (DoS) attack against the users of a particular system (see [85] for some details). The solution therefore can be extended e.g. by captcha-based unlocking mechanism: The user cannot log into the locked account, but the account can be unlocked automatically if the user proves that he is human. Captcha is an automated Turing test, a question that a human can answer, but an automatic program cannot. (e.g. recognition of characters in a picture) ([81])

Other possible protection is to insert some delay into the authentication process after a number of unsuccessful trials. Although this can deny the attack from a single host it cannot solve the problem with a distributed attack, meanwhile, the slowdown due to the delay might be unacceptable by the users.

Protection against DHA is similar to the password brute-force attacks, but an attempt cannot be bound to a particular user. One possible protection against a DHA is to falsify or deny information to the sender if an e-mail is sent to an unknown user (SMTP error 550 according to [46]). We think that this 'hack' is harmful as this information is very important to the users of the Internet. Many other solutions protect against DHA by filtering out the hosts sending a larger volume of e-mails to unknown addresses. As for the brute force password attacks, this method does not protect the system from a highly distributive attack. Commercial solutions also provide countermeasures for harvest attacks. Kerio MailServer ([82]) detects e-mails to unknown users and after a threshold the server begins to filter out possible attackers.

A commercial tool provides managed e-mail services with DHA protection. Their white paper ([83]) gives details about the protection method and about the DHA problem as well. Postini website also has important statistical data about current DHA activity they detected.

Secluda Inboxmaster offers customizable SMTP error message: If a spam is detected during the mail delivery (by other parts of the system), a bounce message is sent back to the sender so the sender might think that the address is not valid. The problem with this method that the e-mail message during the DHA might not be distinguished from legitime e-mails, as its goal is only to identify valid addresses, while the false positives can be misguiding to users.

Latest antispam projects, like ProjectHoneypot ([84]) have not yet integrated protection against DHA. The Project Honeypot system tries to identify spammers by trap e-mail addresses. This can be extended by the identification of mass e-mail senders with many messages to unknown recipients as we propose it later.

5.3 Attack model: The optimal DHA attack

To design effective countermeasure against DHA attacks, it is vital to understand what is the best possible way to perform the attack, for the attacker. If a defense mechanism is good against the optimal form of the attack, generally it is also valid in a real-life scenario. In contrast, whenever we design a protection against suboptimal, empirical attacks, the defense method might fail under an optimal attack. One could also expect that the attackers will soon identify optimized versions of the attack and change the attack behavior to the optimal one.

First of all, we define the Directory Harvest Attacks. The Directory Harvest Attack can be categorized into two main categories:

- The attacker tries all possible valid character combinations with 1...N characters. This can be enhanced that only wordlike strings are used.
- The attacker uses a wordlist (or dictionary) of possible (frequent) user names (e-mail address local-parts). The wordlist is typically based on dictionary words or generated using previously known e-mail address lists.

The typical attacker cannot use up infinite resources, e.g. ten-millions of e-mail trials to a single host, therefore we only analyze the problem of the more efficient wordlist based attacks.

Description of DHA attack method

The actions of a DHA attacker can be described by the following steps:

1. The attacker selects a number of destination domains. The selection can be based on public information available about the domains. (E.g. number of expected users on the system)
2. The attacker gains control over innocent victim computers and turns them into zombies. This can be done using a trojan program or e-mail worm, etc.
3. The attacker carries out the attack by controlling the zombies.
4. The attacker gathers the results from the zombies and analyzes it.

A network attack with a similar method is presented in the case study [86].

Investigated systems we have access to we experienced most harvest attacks aim hosts with immediate SMTP error reporting if a mail is coming to an unknown address. Some of our hosts simply accept all incoming e-mails and pass them to another (protected) host (e.g. a firewall). On these systems a DHA attack is still possible, but the attacker should use and maintain a valid return address to get notified about the unknown users. We found that these hosts are almost never attacked by DHA.

During the last 3 months the attackers used about 100.000 different words altogether. Most of the words show similarity to common e-mail address local parts (like 'cpark', 'jsamson', with typical length of 5-7 characters). Sometimes the words seem to be artificially generated from syllables (eg. 'kizu', 'pugeriu'), when the typical length is 4-6 letters. Most of the words are tried multiple times even on the same domain, but we cannot distinguish if it is tried multiple times by the same attacker, or multiple attackers tried the same word. The attacks were highly distributed, but many times the single trial messages contained 20-30 different recipients.

5.3.1 The DHA attack algorithms

Let's introduce a few notations. An attacker controls $A = \{A_1, A_2, ..., A_{N_A}\}$ zombie computers, where N_A denotes the number of the zombies.

The target of the attack is the following set of domains: $D = \{D_1, D_2, ..., D_{N_D}\}$.

Within domain j in $(1..N_D)$ we have users $U^j = \{U_1^j, U_2^j, ..., U_{N_U^j}^j\}$.

The dictionary (wordlist) of the known e-mail address local parts (i.e. e-mail user names) is denoted by the sequence
$W = W_1, W_2, ..., W_{N_W}$ that is the size of the list is N_W.

For a geographic (cultural) region we can assume that the distribution of the local parts is very similar for every domain.

We assume that the probability distribution of the words $\{P(W_i)\}$ is known by the attacker. The wordlist is sorted according to the descending order of probabilities. ($P(W_i) \geq P(W_{i+1})$ for every i.)

Using 10 million e-mail addresses gathered from the Internet we analyzed the distribution of local parts. Figure 7 shows the success probability for a DHA trial on systems with various user counts. The figure shows an exponential form, a small fraction of the local parts is very common, then many words show similar frequencies.

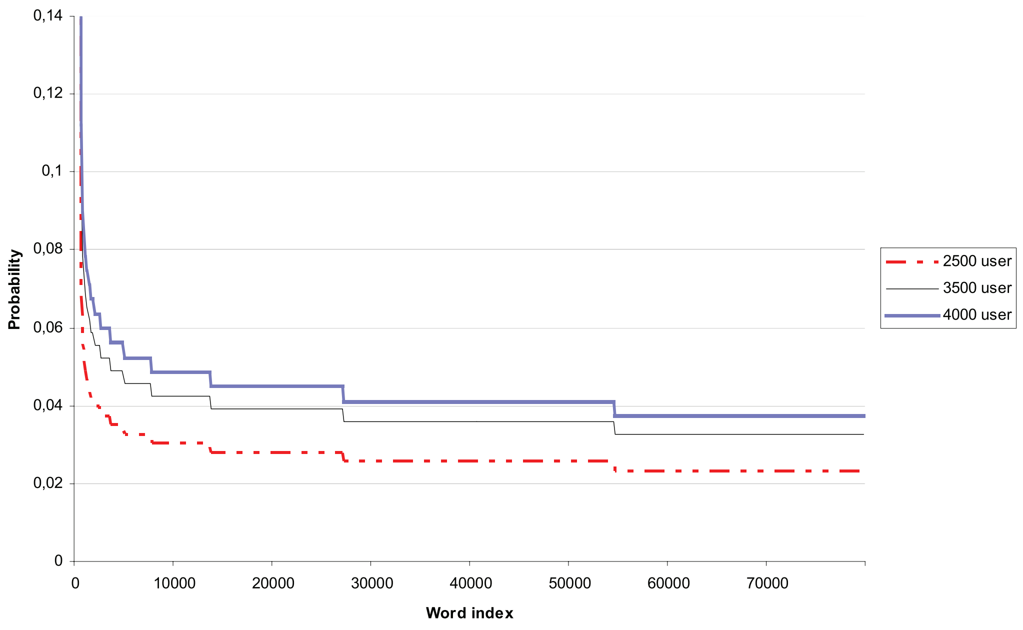


Figure 7: Probability of success for the wordlist elements in systems with different user numbers

We also define the targets/capabilities of the attacker. The number of attack trials is limited by integer t and the number of attacked domains is N_D.

The attacking algorithms are defined as follows: Algorithm 1 is the observed behavior of the DHA attackers. Algorithm 2 is our proposed (optimized) algorithm.

Algorithm 1: The attacker tries the first $T = t/N_D$ items of the wordlist for each domain, which are the words with the highest probability.

Algorithm 2: For every trial the attacker scans all the target domains and selects the domain where the probability of success is the greatest. The trial is carried out against the selected domain.

In pseudocode our proposed Algorithm 2 can be described as follows

```
**for all** $d \leq N_D$ **do**
  $i[d] \Leftarrow 1$
**end for**
$trial \Leftarrow 1$
**while** $trial \leq t$ **do**
  $d \Leftarrow 1, p_{max} \Leftarrow 0, target \Leftarrow 0$
  **while** $d \leq N_D$ **do**
    **if** $p(W_{i[d]} \in U^d) > p_{max}$ **then**
      $p_{max} \Leftarrow p(W_{i[d]}), target \Leftarrow d$
    **end if**
  **end while**
  Try(word $W_{i[target]}$, on domain $target$)
  $i[target] \Leftarrow i[target] + 1$
  $trial \Leftarrow trial + 1$
**end while**
```

The wordlist elements are tried one after the other, $i[d]$ denotes the index of the next word in the wordlist for a given d domain.

5.3.2 Reasons to attack: The economy of the DHA

The attacker might earn profit from selling e-mail addresses for spamming purposes.

The attacker tries to maximize the profit from the attack. The net profit of the attacker is calculated by the following formula $V = -(C_0 - f_c(t)) + I$ where $I = \sum_{j=1}^{t} p_s(w_j) * \mu$.

Here C_0 is the initial cost of the attack, $f_c(t)$ if the cost depending on the number of trials, furthermore I is the income decomposed into sum, where $p_s(w_j)$ is the success probability for the word tried in the step j, while μ is value for an identified address.

We do not know the exact cost function, therefore in this section our goal is to optimize (maximize) the value of the income (I). This corresponds to the maximization of the successfully identified e-mail addresses.

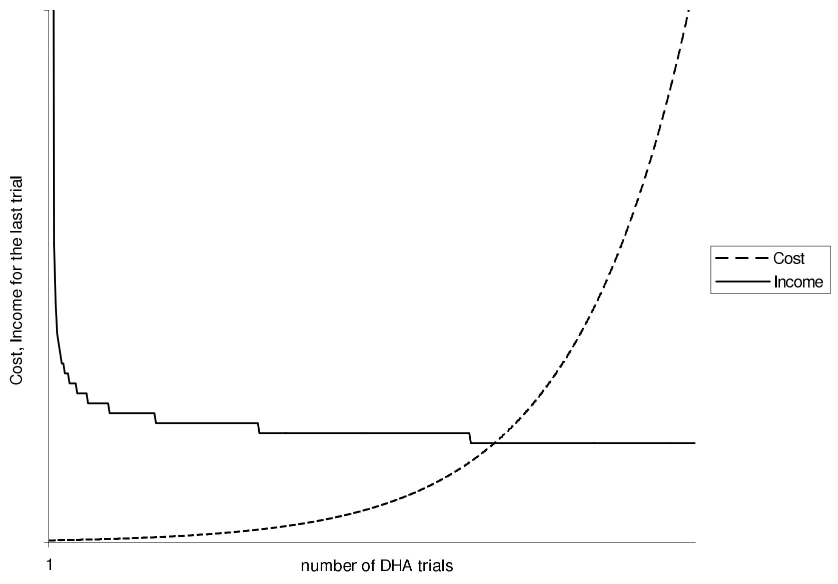


Figure 8: Cost and Income functions of the attacker

Figure 8 illustrates the cost and income curves: The intersection of them corresponds to the maximal profit of the attackers.

5.4 Analysis of the DHA attack Algorithms

Now we give formulae for the expected number of successfully found e-mail addresses.

Let random variable S_i denote the number of e-mail addresses found by the attacker when he attacks domain D_i, $i = 1, ..., N_D$. Random variable S denotes the total number of successes, i.e. $S = \sum_{i=1}^{N_D} S_i$. Below we give an analysis for the expected value $E(S)$ of the successful trials for the Algorithm 1 and

Algorithm 2 First consider Algorithm 1:

$$E(S_i) = E\left[\sum_{j=1}^{T} \chi_{\{W_j \in U^i\}}\right] = \sum_{j=1}^{T} E\left(\chi_{\{W_j \in U^i\}}\right)$$

$$= \sum_{j=1}^{T} N_U^i P(W_j) = N_U^i \sum_{j=1}^{T} P(W_j)$$

where $T = t/N_D$ and χ_A is the indicator function for set A. Hence we get

$$E(S) = \sum_{i=1}^{N_D} N_U^i \sum_{j=1}^{T} P(W_j)$$

In case of Algorithm 2 let t_i denote the number of trials against domain D_i, $i = 1, ..., N_D$. Note, these numbers are determined by the probability distribution $P(W_j) = h(j)$,$j = 1, 2...$, and by the sizes N_U^i, $i = 1, ..., N_D$, as it will be detailed below. Assuming the knowledge of t_i, $i = 1, ..., N_D$, for the expected number e-mail addresses found by the attacker applying Algorithm 2 we get:

$$E(S) = \sum_{i=1}^{N_D} N_U^i \sum_{j=1}^{t_i} P(W_j)$$

For the calculation of the number of trials, we use the Lagrange's multiplicator technique. We have to maximize function

$$H(\underline{t}) = n_1 h(t_1) + ... + n_k h(t_k)$$

under conditions $g(t) = t_1 + ... + t_k - t = 0, t_1 \geq 0, ..., t_k \geq 0$ where $\underline{t} = (t_1, ..., t_k)$, $k = N_D$, $n_i = N_U^i$. We start from function

$$G(\underline{t}) = H(\underline{t}) + \lambda g(\underline{t})$$

and we solve the following system of $k + 1$ equations:

$$\frac{\partial G}{\partial t_i} = n_i h'(t_i) + \lambda = 0,\ i = 1, ..., k$$

$$\frac{\partial G}{\partial \lambda} = t_1 + ... + t_k - t = 0$$

where h' denotes the derivative of function h. Eliminating λ we arrive to the following system of k equation in unknowns t_i, $(i = 1, ..., N_D)$:

$$n_1 h'(t_1) - n_{i+1} h'(t_{i+1}) = 0,\ i = 1, ..., k - 1$$

$$t_1 + ... + t_k - t = 0$$

For instance, we show how to solve this system, when function h has the following form $h(x) = a/x^b$, $h'(x) = c/x^d,\ c = -ab,\ d = b + 1$. After some straightforward algebra we arrive to the following system of linear equations:

$$m_1 t_{i+1} - m_{i+1} t_1 = 0,\ i = 1, ..., k - 1$$

$$t_1 + ... + t_k - t = 0$$

where $m_i = n_i^{1/d}$. Whence we get the following result

$$t_i = t \cdot n_i^{1/d} / \sum_{i=1}^{N_D} n_i^{1/d},\ i = 1, ..., N_D$$

It is easy to check that $t_1 \geq t_2 \geq ... \geq t_{N_D}$ and $t_1 + t_2 + ... + t_{N_D} = t$.

Algorithm 2 is optimal in the sense that the expected number of successfully identified e-mail addresses by the attacker is maximal.

Sketch of proof: Assume that there exists a certain set of trials (X) that is better than the set (Y) produced by Algorithm 2. (both sets have the same size t) Therefore there should exist an element (word, domain pair) $x \in X$ for which $x \notin Y$. Because Algorithm 2 selects elements with highest possible success probabilities, therefore element x must have success probability less than or equal to the probability for any element from Y. Let's substitute the element x in the set X by an element y, $y \notin X$ and $y \in Y$. Let the modified set be denoted with X^*. The sum of success probabilities for X^* compared to the same of X will be higher or it remains the same. As we have shown above this sum of probabilities is actually the expected number of successfully identified e-mail addresses. In case when the summed probabilities over X^* is greater than that of over X we arrive to a contradiction. (X is not better than Y) In the other case when the two sums of probabilities are equal, we repeat the step of substitution. Exhausting all possible substitutions we arrive to a contradiction or we will see that the result of Algorithm 2 is not inferior to the optimal set X.

5.5 Simulation of the attack algorithms

To support our results we carried out simulations. Two possible scenarios with different number of target domains and number of users were investigated. The results are summarized in Table 17.

Table 17: Simulation results

N_D	Total user	succ. Alg. 1.	succ. Alg 2.
11	1730	87	180
6	23000	5395	6754

The heading "Total user" denotes the total number of users in D, furthermore "succ. Alg i" denotes the average of successfully identified addresses by Alg. i. The simulation clearly shows the superiority of Algorithm 2.

5.6 Countermeasure Against DHA

We can separate the possible countermeasures in two categories:

- Host based protection: An autonomous system has its own protection method, without relying on other parties.

- Network based protection: The system is cooperating with other parties to protect itself from the DHA. This method can be centralized, when a server coordinates the protection.

5.6.1 Host based protection

A DHA attacker uses information gathered about unknown users to identify valid users. Some protection methods try to achieve protection against DHA by falsifying or denying information about unknown users. As we mentioned before, these protection methods rise new problems for the legitimate human users, therefore any protection method based on this behavior should be avoided.

Filtering based on error reports: When a computer is sending e-mail for an unknown user, we insert the address of the computer to a list and filter out all requests coming from computers on the list. To deal with the problem of false positives we only insert computers into the list if the number of e-mails to unknown recipients exceeds a threshold. The problem with such host-based filtering algorithm is that it is not resistant against distributive attacks. Experience shows that DHA attacks are highly distributive, the maximum of trials coming from an IP address can be as low as 1-2. Although the filtering algorithm will filter out thousands of computers, the attacker will continue the attack from thousands of other, not filtered zombies.

The simple host based filtering can be extended:

- We can examine the address field of attacking e-mails. Some attackers sort the wordlist in alphabetical order. Similar suspicious properties can serve a starting point of a protection method.
- The attackers generally use the most frequent e-mail addresses during the attack. This list can be reconstructed by observing the trials of the attackers. An enhancement to the basic protection method can filter out all the IP addresses trying to send e-mails to unknown users where the particular username tried exists in this wordlist.

Although these enhancements could be successful for a short time period, the attackers can also adapt to the protection and the trials will not be statistically distinguishable from legitimate e-mails. Even if this protection would be successful to protect a single host, this approach will not help any other servers and the Internet to efficiently fight against DHA.

The filtering approach can be real-time or log-based. If the protection is based on log analysis and is not real-time then the attacker has wide time windows to carry out attack enabling him to test lots of addresses from a single attacking host.

The behavior of the server to the offending computers (computers on the list) can also be different: Some protection methods deny any IP traffic coming from those computers, while other servers only reject receiving e-mails from the attackers. It is also possible that the server only reports a temporary error (like in greylisting [87]) to the attacker thus the attacker won't know if he is filtered or the delivery should be retried later.

5.6.2 Network based protection method

Our proposed solution in this section is also based on filtering but with a centralized approach. Parties of the protection are the following: the attacking computer, the attacked server host and the centralized DHA

RBL (Real-time blacklist) server. (check [88] for general some information about anti-spam techniques like RBL)

If an attacker sends an e-mail to an unknown address on the attacked server, the attacked server will send an error report to the central DHA RBL server. The error report contains the offending IP address, the recipient address tried, and the time of the attack. The centralized server collects the reports from the protected servers. If the number of trials exceeds a limit, the server inserts the address of the attacker in the blacklist. The server also stores the time when the last e-mail observed from the given attacker with unknown recipient.

As usual, the RBL list can be queried by sending an address as a request to the server questioning if an address exists in the list. The server does not publish addresses on the list, instead it simply answers with yes or no.

5.6.3 Aging

For every blacklist-based method it is very important to decide how the addresses are cleared from the list. After removing the address from the blacklist the attacker can continue the attacks, but if an address is remained too long on the list, then legitimate traffic might be blocked for a long time.

Several methods for clearing the blacklist:

Administrator-driven aging: The administrator of the RBL server manually decides about the removal of the address. The owner of the attacking zombie computer can also ask the RBL administrator to remove the computer from the list. (e.g. the backdoor used to carry out the attack is removed from the computer)

Simple aging: After a given amount of time elapsed from the insertion of the address into the RBL the address is removed automatically. The problem with simple aging is that an attacker can easily estimate when the address is cleared from the RBL, therefore he can immediately restart attacking hosts.

Multi-phase aging: The central entity might store historical data on the attackers, so when a recurring attack is detected, the aging time, or the detection threshold might be different from the first case of attack detection. It is also possible to use the attack reports that are received during the blacklist period to lengthen the aging for recurring attackers or to shorten the period when no illegal activity is detected during the blacklist period.

5.7 Simulations on the proposed blacklist algorithms

Simulations based on the different algorithms and parameters were carried out to show the behavior of the protection.

5.7.1 Simulation 1: Different blacklist thresholds

The following basic properties were used during the simulations:
The number of attacking computers is 200, every attacking computer tries to send an e-mail from time 0 in a rate of 1 e-mail/sec. 25 computers are attacked, every computer has 200 distinct users, the attacking

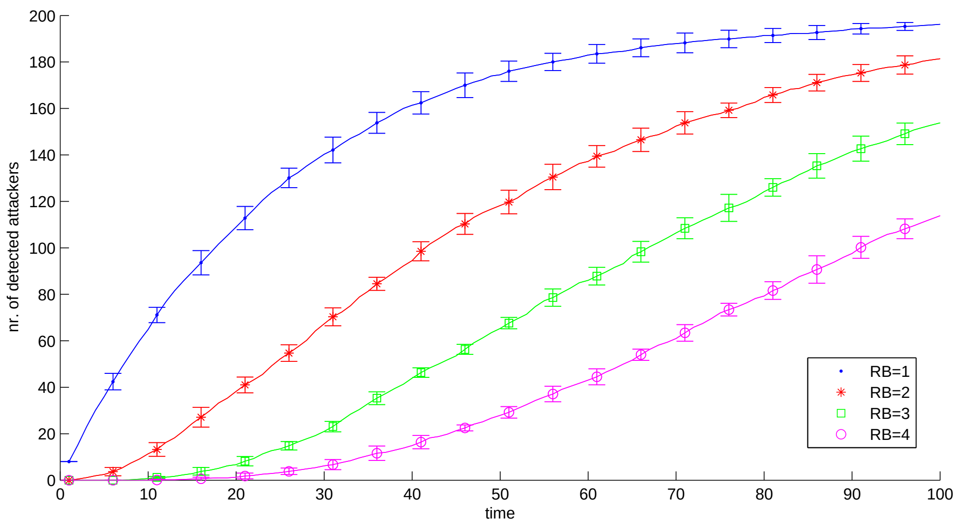


Figure 9: DHA protection with different blacklist thresholds

computer always selects a target based on the optimized attack algorithm. The protection algorithm also runs from time 0.

All simulations were averaged from 10 runs and the appropriate values of standard deviation are shown on the figures.

In Figure 9 I compare the attack detection when only a single computer uses individual DHA detection. The RB parameter gives the number of (unsuccessful) attack trials, after the detecting computer identifies the host as an attacker. This figure shows that a higher RB results in a slower detection of the attacking sources.

5.7.2 Simulation 2: Different number of detectors

Similarly, the number of detecting computers in the set of the attacked hosts also affects the speed of the detection. If a higher number of hosts report DHA trials to a central black-list engine, then there is a higher chance to detect the attacker soon after the beginning of the attack. Corresponding simulations are shown in Figure 10. As above, the set of the attacked computers consists of 25 hosts, whereas the number of detecting hosts (D) varies from 1 to 5. $D = 1$ also shows what would be the detection speed without a central entity (local detection). $RB = 4$ and $RB = 2$ again corresponds to the threshold of the attack identification, after RB detected attack trials we consider a remote server as an attacker.

5.7.3 Simulation 3: False positives with different number of detectors

The simulations shown above do not give information about false identification. There is a trade-off between the speed and chance of detection of real attacker in contrast to the false positives, where the system identify legitimate users (e.g. mistyping addresses or using old, invalid data) as attackers. In

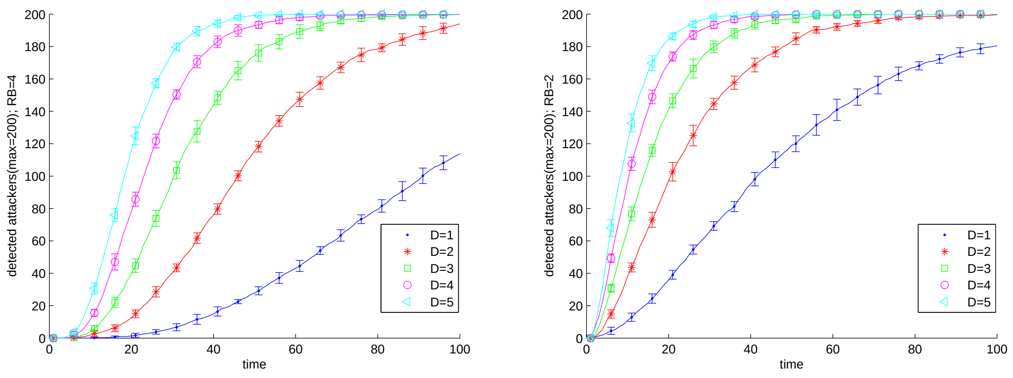


Figure 10: DHA protection with different number of reporter computers (D); RB=4 and RB=2

Figure 11 I show how the speed of the attack detection in this case affects the false positives. Three cases are shown, for different number of detectors ($D = 1, 5, 8$). An attack is considered after two identified trials ($RB = 2$).

The non-attacking legitimate users are simulated in the following way: There are 200 hosts, servers that send out false e-mails. These legitimate hosts send out e-mails in a lower rate than the real attackers, only 0.1 e-mails in each time slot, to a random target host. The target hosts cannot identify legal and illegal activity, therefore all such e-mail is also reported to the central engine for processing.

The figure shows that the red line is the fastest for detecting an attack ($D = 8$), but in that case the false positives (detected and filtered legal users) is also growing rapidly. In contrast, $D = 1$ belongs to the slowest detection of the attacking hosts, but also produces lower number of false positives.

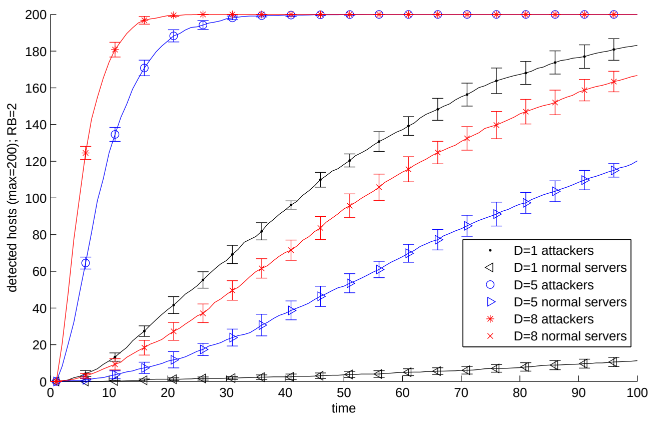


Figure 11: Detection of attackers vs. legal servers, different number of detectors RB=2

Intuitively, the higher number of detecting hosts gives a more clear view on the network activity, the attackers can be identified more accurately by obtaining more information at more places. Therefore, instead of lowering the detecting computers to avoid false positives, it is a better idea to have a high number of detectors and use additional tricks to avoid false alerts. A simple way is to modify the RB threshold to lower the number of false positives.

5.7.4 Simulation 4: False positives with different RBL threshold

In Figure 12 I show how the RB parameter can affect the number of false positives if the number of detectors is relatively high ($D = 8$). With a lower value of RB, the number of false positives is higher. Again, without aging the number of false positives converges to the number of all innocent hosts. The aging algorithms will be simulated later.

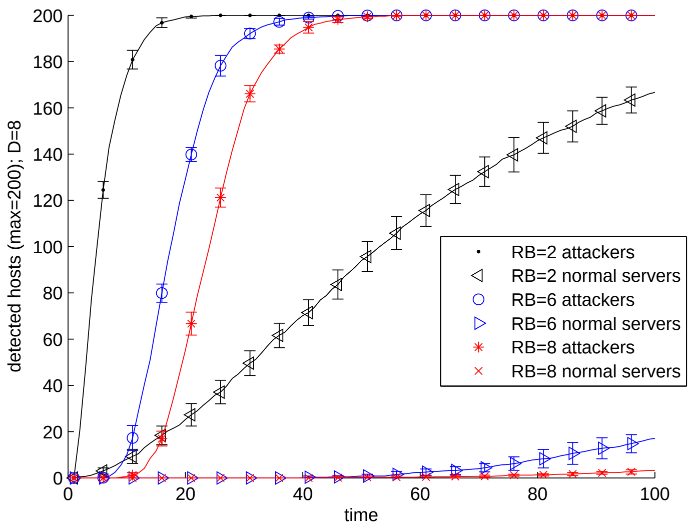


Figure 12: Detection of attackers vs. legal servers, different RB values D=8

5.7.5 Simulation 5: Lowering communication overhead with local detection threshold

Communication overhead can be reduced by implementing a local threshold at every detector computer. The computers do not send alarm messages to the central RBL after each received attack trials (or legal e-mails with bad address), just when the number of received e-mails from a single server reaches a threshhold (LRB). For threshold $LRB = 2$ I made simulations with the same method that is used on the previous figure. The results are shown in Figure 13. Not surprisingly the detection is slower compared to the same level of RB in the last example, but the number of false positives is also lower. $RB = 2$ in Figure 13 gives about the same detection behavior in this scenario that we have seen with $RB = 6$ in the case of Figure 12, meanwhile, only the communication overhead, the number of messages between the detectors and the central element is much lower, only about the $1/3$ of the messages were needed to achieve this performance.

Using this threshold makes it easier for the attacker to hide, if their servers send messages in a lower number to the detectors than the local threshold, the attacker will be invisible on the global level, therefore the local threshold might not be too high.

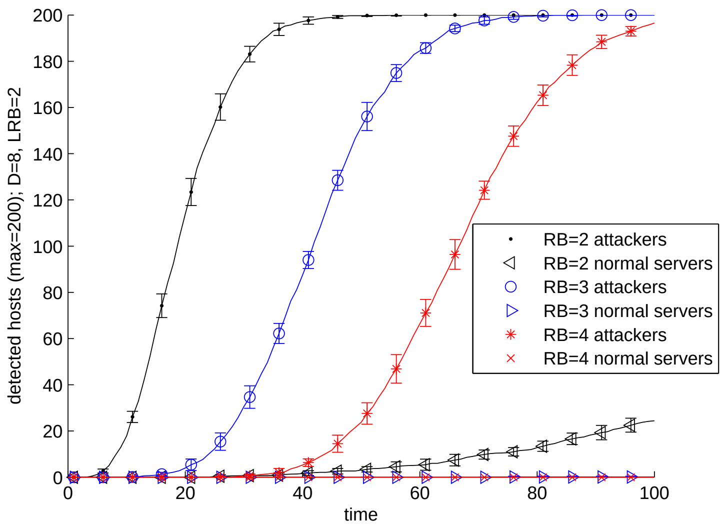


Figure 13: Detection of attackers vs. legal servers, different RB values D=8, local BL threshold=2

5.7.6 Simulation 6: Simple aging

The simulation results of the simple aging process is shown in Figure 14 and Figure 15. In Figure 14 $RB = 2$ was used under the same situation as in Simulation 4, no local threshold was used (every detected local event was directly sent to the server). In that case (Figure 12), the number of false positives rose rapidly to about 160 at the end of the simulation, and without aging it would rise until every legal sender (200 servers) would be listed. In Figure 14 the simple aging method was used in with a parameter MA. If an attacker is identified by the central RBL entity, it does not remain in the database forever, after MA seconds it will be removed from the list with all of the previous records. The potential attacker will be relisted if the number of trials received by the detectors after the removal from the blacklist rises above the RB level again. Different MA values were used (10,20,40), with $RB = 2$ and $D = 8$. It can be seen in the figure, that the number of false positives not rises so rapidly and converges to about 80 in the case of $MA = 40$ instead or even lower to 25 ($MA = 10$). The aging process helps to limit the number of false positives, but does not eliminate the problem. If $MA = 10$, for example, then the number of false positives is relatively low, but the number of detected attackers converges to about 130, so about $1/3$ of the attackers can carry out attacks continuously. Because of this trade-off, the system is very sensible to the MA parameter in a real-life scenario.

The reason of the periodicity (depending on MA) of the number of detected hosts in the figure is that the attack is started at the same time ($t = 0$) simultaneously from every host. In Figure 15 the time frame of the simulation was modified from 100 to 200 to show more insight on this periodicity and convergence.

Again, if no aging is used, then the number of false positives converges to 200, the total number of legal servers.

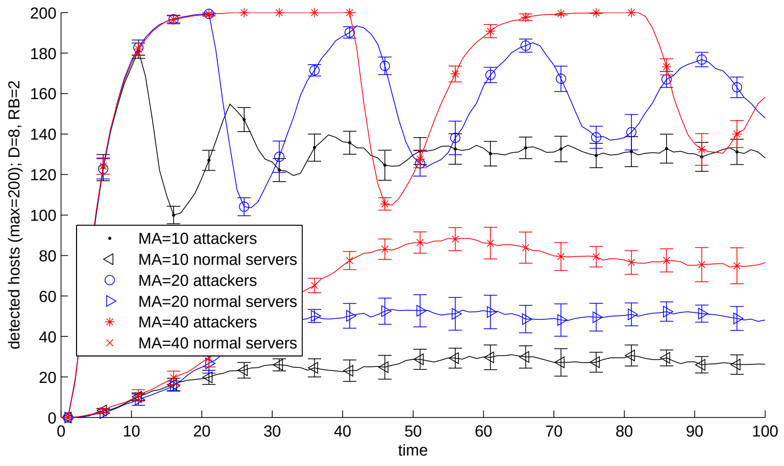


Figure 14: Aging. Detection of attackers vs. legal servers, RB=2, D=8, different MA values

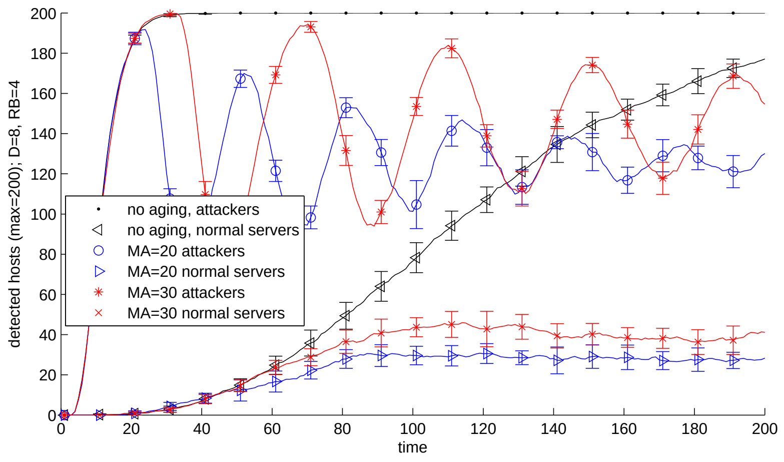


Figure 15: Aging. Detection of attackers vs. legal servers, RB=2, D=8, different MA values

5.7.7 Simulation 7: Simple and A1 aging algorithm

In Simulation 6 the simple aging algorithm worked in the following way: After a possible attacker was removed from the blacklist, the number of registered attack trials was reset to 0. Therefore, from that time, RB new trials were needed to put the host again into the blacklist.

In the current simulation, a modification of the aging algorithm is used. In this new algorithm (denoted by algorithm A1), a second RB parameter, $RB2$ ($RB2 < RB$) is used, after trials are detected from a previously blacklisted host, the threshold is lowered to $RB2$, so repeated attacks are faster identified. The results are shown in Figure 16.

The results show that the number of identified attackers converges to a higher number (about 160), but the number of identified legal senders also rose. In long term, aging algorithm A1 behaves the same way as the simple aging algorithm with a lower RB value, this can be seen by comparing the A1 algorithm curve in Figure 16 to the $MA = 20$ curve in Figure 14. In fact, this kind of adaptation technique would only help to avoid false positives, when the legal server sends out e-mails with bad recipients in a burst, then later they send out only legitimate e-mails. The identification of the burst would be avoided by a high RB value, but continuous attackers can still be identified by a low $RB2$ value.

A possible modification of the above mentioned two aging techniques is that if the central element uses aging not just on the blacklisting of a suspected attacker, but also for the recorded attack trials are also removed when they are too old. This can cause a lower number of false positives, but also enables real attackers to hide if they attack in low intensity.

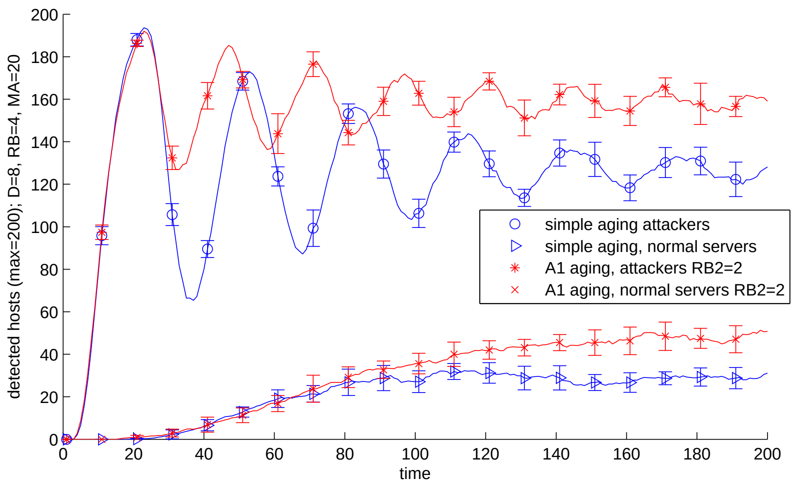


Figure 16: Aging. Detection of attackers vs. legal servers, RB=2, D=8, MA=20, simple and A1 aging

5.8 Adaptive detection

In the previous simulations I investigated the proposed algorithms for both positive properties (identified hosts, speed) and drawbacks (false positives). As a conclusion, it is a hard task to select the appropriate algorithm and parameters in real-life, because the optimal behavior, parameter values depend on the actual network, attack properties. It is evident that the proposed algorithm can successfully protect against the DHA attack, but the number of false positives might be very high and this causes problems in real-life environments.

In some cases, an adaptive detection technique might be proposed to lower the number of false positives. The main idea of the algorithm is to have different detection parameters during an attack and during normal conditions. At global level, attacks are carried out continuously, therefore this adaptive detection technique might not work, however, in smaller environments, where only a lower number of servers should be protected, this method might be efficiently used. A possible environment could be a company with multiple branches, multiple e-mail server that should be protected against DHA attacks.

The technique itself is very similar to the detection technique used in claim 2 (protection based on traffic analysis against Denial-of-Service problems).

Attack detection

The main tool of the adaptive detection mechanism is the attack detection method. A very simple attack detection can be considered, the central entity (RBL) server obtains alerts of possible attack trials, and measures the intensity of the reports. If the intensity goes over a limit, an attack situation can be considered. Later, based on the intensity measurements, the RBL server can also recognize that the attack is over and change the parameters back to the normal situation.

Parameter setting

The main goal of the adaptive technique is to avoid false positives, when the system is not under attack. Therefore, during the normal traffic the RB threshold might be very high to avoid false detections. Also, the aging in this case might be much more permissive, deleting attacking hosts from the RBL after a shorter time period.

When an attack is detected, the RB parameter should be lowered to identify attackers more rapidly. In this phase, the number of false negatives might also rise because of the change of the parameters.

5.8.1 Simulation 8: Adaptive detection

I give insight how the adaptive detection might work by a simulation. The simulation compares the normal detection scheme with the proposed adaptive algorithm. In general, this simulation uses the same parameters and models as Simulation 1-7. In this simulation the time-frame was 300 seconds, and the attacking hosts only begin their attack at $t = 150$. However, the legal servers still send e-mails from $t = 0$. Simple aging is being used, with parameter $MA = 40$ (even after the attack is detected). The blacklist threshold is $RB = 2$ for the normal attack detection scheme, for the new, adaptive detection scheme, two parameters are distinguished, instead of RB, a new variable $RBN = 10$ is used during the normal situation, and $RBA = 2$, when an attack is detected, instead of RB.

The attack detection works in the following way: the central entity receives attack reports in every second and calculates the number of reports received in that second. If the value is above 20, then an attack is detected. This behavior should be refined in real-life environments, but simplifies the simulation.

The results are shown in Figure 17. Comparing the false positives, not surprisingly, the proposed adaptive mechanism can significantly lower the false identifications. When the attack began at $t = 150$, both algorithms rapidly identified the attackers, as the simulated attack detection can respond to the attack instantly. After $t = 150$ the adaptive algorithm behaves like the normal detection scheme, therefore in the long run, the number of false positives are the same.

An overshooting of the false positives might be recognized in the figure (at about $t = 190$ the curve is significantly higher than the curve for the normal detection). The causes of this is that even if the $RBN = 10$ and only a low number of false positives present before $t = 150$, the reports are carefully collected by the RBL server, and when the attack is detected and $RBA = 2$ is used, the legitimate servers are blacklisted much faster because of their history at the central entity.

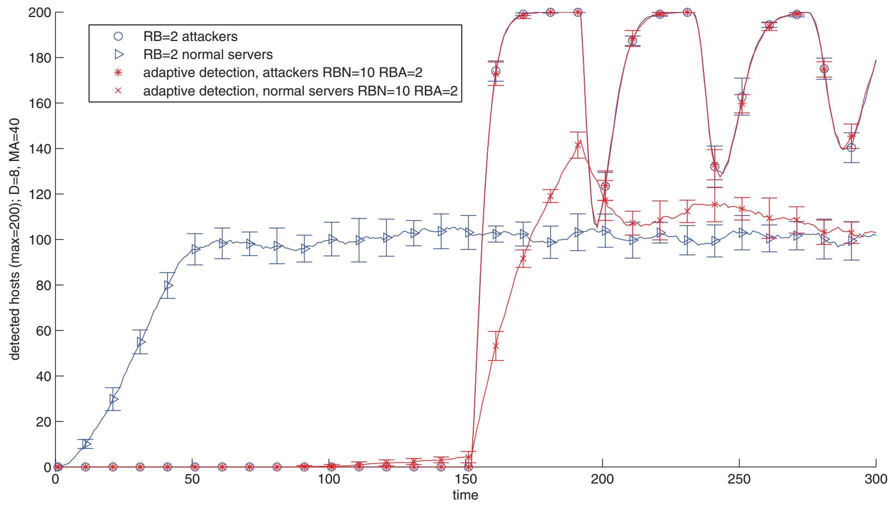


Figure 17: Adaptive vs. normal detection of attackers, D=8, MA=40, simple aging

5.9 DHA Protection Prototype

Based on the proposals, I developed a DHA protection prototype. The prototype system uses the DNS protocol for transferring queries and reports to the server and also to transfer the answer from the RBL server. The DNS protocol is widely used in the RBL environment as it is very robust and the inherited caching mechanism of the DNS servers can help lowering the workload of the server.

The procedure of incident reporting is presented in Figure 18.

step 1 The attacker sends an e-mail to an Internet mail server (MTA).

step 2 The server answers with valid information: the user is unknown in the system.

step 3a The MTA sends an incident report to the server. This is done by a DNS query with a special format. The queried DNS name contains the information about the offending host.

step 3b The DNS server of the MTA forwards the query to another DNS server or directly to the DHA RBL server. The RBL server decodes the query and processes it.

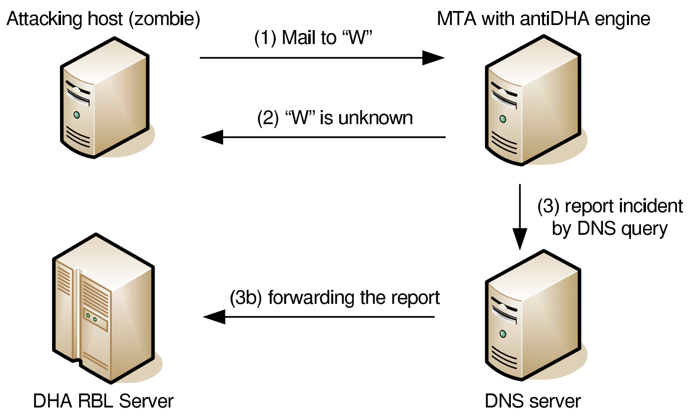


Figure 18: Reporting an incident to the antiDHA system

The filtering mechanism in our prototype system is very simple: After a low number of e-mails sent to unknown addresses (currently set to 10) we insert the offending address into the RBL list.

Figure 19 shows the procedure when an enlisted attacker tries to send a new mail to an arbitrary protected host.

step 1 The attacker tries to send an e-mail to an Internet mail server (MTA).

step 2 The protected hosts sends a DNS query with the address of the client (the attacker) embedded in the query.

step 3 The DNS server of the MTA forwards the query to the server.

step 4 The RBL server answers the query with a special form of IP address, meaning "Yes, the computer is in the RBL list". The DNS server can cache the answer for a given address, the caching time (TTL- time to live) can be controlled by the RBL server.

step 5 The DNS server sends back the answer to the protected host.

step 6 The protected host denies the connection with the attacker. This can be done at TCP level, or the attacker can be denied with a proper SMTP error code and explanation.

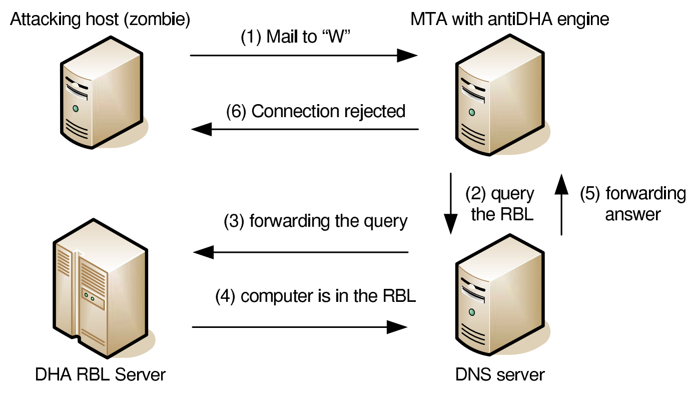


Figure 19: Filtering attacker using the data from the DHA RBL

The results of centralized filtering that the attacker can carry out only a very limited number trials against the protected domains. Of course, our method does not limit the attacks carried out against unprotected domains. The effect of the protection therefore modifies the expected income of the attacker.

5.9.1 Further applications of the results

In the introduction of claim 3 the e-mail Directory Harvest Attack (DHA) was described as a special form of brute force attack. The attacker's goal is to gain information about the e-mail addresses used in a domain name.

However, there are other network attacks that behave very similar to the case of the DHA. These are mainly scanning attacks: vulnerability scanning, SSH login scanning, IP portscan, etc. The attacker tries to identify vulnerable hosts, or hosts with special resources, e.g. web servers. In most cases, the attacker's goal is to collect a list of vulnerable hosts or to infect as many computers as it can.

In the IPv4 Internet, the attacker should try most of the 2^{32} possible IP addresses to find the targets. However, a large part of the Internet IP address range is unused, or filtered, and the density of IP address occupation is not uniform. More information on IP address occupation can be found in [106, 107].

The fact that the distribution of the used IP addresses is not uniform leads to the possibility of the optimization of scanning attacks in a similar fashion that I described in the case of DHA.

The scanning for vulnerable computers is not the ultimate goal of the attacker, it is just a utility. The final goal is e.g. to gain profit from spam messages, and for sending out those messages the attacker needs to control a number of hosts, a botnet. To gain control over computers, the attacker uses exploits on vulnerable hosts, and thus the scanning effort is only one step in the workflow. The attacker does not want to find every single vulnerable host, it just needs to collect a certain number of vulnerable hosts to be able to successfully reach the final goal.

Also, in these scanning efforts the same trade-off can be identified, what I have shown in the case of the DHA. Using more zombie computers or scanning for a longer time can cause higher chance for identifying the attacker, therefore the attacker will use as few hosts as it is possible for the specific goal.

The above described scenario brings up the idea: The attacker should optimize the attack according to the needs and try to focus the scanning efforts onto those IP address ranges, where the preliminary success possibilities are the highest. In this case, those IP ranges should be scanned, which have the highest density of used IP addresses. In special cases, e.g. for finding vulnerable Windows computers, other ways might be possible to optimize the attack, e.g. there are more windows computers on dial-up lines and therefore the optimized attack might consider to scan known dial-up IP areas first.

The optimization of the attack can help the attacker also by other means:

In the case of IPv6 networks, it is infeasible to scan all of the 2^{128} IP range for vulnerable hosts, or hosts with particular services. In this case the distribution-based scanning is one of the possibilities to still successfully collect information from the network.

Another situation is the Internet worm propagation. After the worm successfully attacks a host, the new host will also start to search for possible targets. This way, the higher number of infected hosts will increase the speed. Due to this exponential effect, higher attack success ratio at the beginning of the attack will speed up the propagation of the worm seriously.

A secondary effect of the attack optimization is that the IP address ranges with more density can receive more attacks from more attackers than the sparse IP areas. The authors of [106] also identify this problem:

> *If malware creators have knowledge of address statistics, then we might expect attacks of increasing ferocity in the regions highlighted by the graphs. Protection of the high-occupancy regions should therefore be a high priority.*

Chen and Li in [110] give more insight into the topic of optimized worm attacks, which are very similar to my analyzed optimal DHA attack. The main difference is that in the worm attack the attacking computers are currently mainly individual computers without coordination of their scanning efforts, and the target hosts can be handled as larger blocks in contrast to the DHA case. The paper mainly investigates the different scanning characteristics of worms and analyzes these methods (importance scanning, localizes scanning, modified sequential scanning). They results also show that with such optimized worm vulnerability scanning attacks are still feasible on IPv6.

My results on DHA and the results for the optimized worm attacks give more insight into what we have to count on in the future for Internet attacks. A results is that the propagation of the worms might be better forecasted, but this is not the only possible application for the results.

5.9.2 Botnet, attacking network size estimation based on measurements on optimized attacks

One of the largest threats to the Internet are currently the botnets. These networks of zombie computers together have an enormous amount of resources to carry out any kind of harmful network activity. The researchers also began investigating botnets and analyze their basic properties.

One of the simplest questions regarding botnets is how big a specific botnet is, how many computers are participating in them. However, estimating the size of the botnets is not simple. The botnet owner wants to hide all information about the botnet itself and the analysis of the controlling mechanisms of the botnets might be difficult. A number of papers, like [108, 109] made efforts on this topic, but still, the results might be questionable. Traditional botnet controlling mechanisms used such tools, as the iRC, the Internet relay chat protocol, which made it directly possible to measure the size of the botnet. New botnets use state-of-the-art peer-to-peer technologies, such as distributed hash tables (DHT) and encryption and authentication schemes. These advancements make the task of analysis of the botnet very hard.

The optimization effort of the network scanning in the activity of an attacker (or a botnet or worm) might make it possible to measure the size of the botnet. The sketch of this estimation is the following:

Let's consider the most simple case: An attacker tries to find e.g. 1000 hosts that are vulnerable for a specific attack to use them as spam servers. The attacker knows the vulnerability distribution of the hosts, therefore the attacker starts the vulnerability scanning with the most probable subnets. After enough computers are infected, the attacker stops the scanning and no further computers will be scanned.

Now, let's consider that for our estimation we put a number of honeypots inside those subnets that are probable targets for scanning efforts. What we will see is that on some of our computers attack trials will be reported, while on the rest no attack trials are identified. Of course the attacked honeypots will belong to those subnets, where the successful attack probability is the highest. Now, If we know the successful attack probability distribution of the network and we identify where the attacker stopped the attack, we can calculate the estimation on the successful attacks which is the same as the probable size of the botnet.

Of course, the exact method of such estimation highly depends on the situation, the actual algorithm that the attacker (or botnet,worm), but clearly, it opens up a new possibility to estimate the size of an attacking network, botnet.

5.10 Summary

In this section I analyzed e-mail Directory Harvest Attacks. Any protection can be the most efficient if we know the best possible attacks. An optimal attacking algorithm based on wordlist statistics has been presented.

I designed an optimized DHA attack, where the optimization is based on the probability distribution of e-mail local parts and on the expected number of valid addresses in the targeted domain. I gave formula for the expected number of successfully collected valid e-mail addresses. I proved that in my model the attack is optimal, the expected number of the successfully collected e-mail addresses is maximal. I illustrated with simulation results that the optimal attack method is more efficient than the non-optimized attack.

I derived formulae for the expected number of successfully attacked e-mail addresses for the presented algorithms. The simulation results based on real data supported the feasibility and efficiency of my proposed algorithm.

I investigated in detail the possible countermeasures against Directory Harvest Attacks. I proposed a new architecture for the protection against DHA with centralized approach. The proposed method uses a centralized, real-time blacklist (RBL) based protection method. I also developed a prototype system to show the applicability of my proposal. The prototype is able to identify attackers, report the attackers to a central server and filter out the attackers based on the information retrieved from the server.

6 Conclusion and outlook

My research is presented in three main sections. In the first claim, I focused on the Denial-of-Service problems with a general approach. My proposal is to use the client-side puzzle technique combined with a game theoretical approach. New protocols, resistant to DoS attacks can be developed by using my work.

In the first part of my second claim I show that an SMTP server is vulnerable to DoS attacks. A more specific countermeasure opportunity, a method based on traffic analysis is proposed and analyzed within this section. The proposed method was also supported by a prototype application. My results show that traffic analysis might help to handle the problem of DoS in specific services.

In the third claim I propose a technique to prevent DoS attacks on SMTP systems based on DHA attacks. In the I analyzed the problem of Directory Harvest Attacks, and proposed a method to give protection against it. I

I always tried to support my results with practical prototypes during my research. I designed and developed the following prototypes during my work:

- DoS front-end based on traffic level measurements
- Protection prototype against DHA attacks

With all these prototypes I proved that my proposed methods are workable and ready to deploy in a real-life environment.

The protection against DHA and DOS (based on traffic level measuremets) were deployed into a real-life product environment to protect hundreds of users of some SMTP servers. To successfully deploy my proposed methods I had to refine parameters to avoid false positives. I also had to white-list some servers that my protection mistakenly identified as attackers. These hosts in fact can be considered as real attackers, but many times these are servers of reputable Internet providers forwarding attacks from infected hosts in their internal network.

The SMTP related protection methods are also used in our reseearch during the DESEREC (Dependable security by enhanced reconfigurability) project of the EU. FP6 framework programme (contract no. 026600).

In the future I intend to generalize my proposed techniques. The real goal behind my research is to make the Internet more secure. As this goal seems to be unreachable without fundamental, architecture-level changes, I will try to focus on how these methods can be incorporated into the future Internet.

References

[1] T. Aura and P. Nikander "Stateless protocols" In Proceedings of the ICICS'97 Conference, Springer-Verlag, LNCS volume 1334, 1997.

[2] D. Dean and A. Stubblefield "Using client puzzles to protect TLS" Proceedings of the USENIX Security Symposium, August 2001.

[3] C. Dwork and M. Naor "Pricing via processing or combatting junk mail" In *Advances in Cryptology – Crypto '92*, Springer-Verlag, LNCS volume 740, pp. 139–147, August 1992.

[4] P. Eronen. "Denial of service in public key protocols" In Proceedings of the Helsinki University of Technology Seminar on Network Security (Fall 2000), December 2000.

[5] T. M. Gil "MULTOPS: A data structure for denial-of-service attack detection" Technical Report, Vrije Universiteit, 2000.

[6] J.P. Hespanha and S. Bohacek "Preliminary results in routing games" In Proceedings of the American Control Conference, volume 3, pp. 1904–1909, 2001.

[7] A. Juels and J. Brainard. "Client puzzles: A cryptographic countermeasure against connection depletion attacks" In Proceedings of the IEEE Network and Distributed System Security Symposium (NDSS '99), pp. 151–165, February 1999.

[8] C. Dwork, A. Goldberg, and M. Naor, "On Memory-Bound Functions for Fighting Spam" Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2003), August 2003.

[9] B. Laurie, R. Clayton ""Proof-of-Work" Proves Not to Work " WEAS 04, http://www.apache-ssl.org/proofwork.pdf, May 2004.

[10] A. Back, "The Hashcash Proof-of-Work Function" Jun 2003.

[11] A. Back, "Hashcash - Amortizable Publicly Auditable Cost-Functions" Tech Report, Aug 2002.

[12] X. Wang, D. Feng, X. Lai, H. Yu "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD." Cryptology ePrint Archive, Report 2004/199, Aug 2004.

[13] X. Wang, Y.L. Yin, H. Yu "Collision Search Attacks on SHA1." Feb 2005.

[14] S. Brands. "Untraceable off-line cash in wallets with observers." Lecture Notes in Computer Science, Vol. 773. Advances in Cryptology - CRYPTO 93, pp. 302-318, Springer, 1994.

[15] S. Brands. "An efficient off-line electronic cash system based on the representation problem.", CS-R9323 report, Centrum voor Wiskunde en Informatica, Computer Science/Departement of Algorithmics and Architecture, 1993.

[16] D. Chaum, J.-H. Evertse, J. van de Graaf. "An Improved Protocol for Demonstrating Possession of a Discrete Logarithm and Some Generalizations", Advances in Cryptology EUROCRYPT 87, Springer-Verlag, pp. 127-141.

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

[18] J. Leiwo, T. Aura, P. Nikander. "Towards network denial of service resistant protocols" In Proceedins of the IFIP SEC 2000 Conference, August 2000.

[19] K. Lye and J. M. Wing "Game strategies in network security" In Proceedings of the Workshop on Foundations of Computer Security, Copenhagen, Denmark, 2002.

[20] K. Matsuura and H. Imai "Protection of authenticated key-agreement protocol against a denial-of-service attack" In Proceedings of the International Symposium on Information Theory and Its Applications (ISITA'98), pp. 466–470, October 1998.

[21] C. Meadows "A formal framework and evaluation method for network denial of service" In Proceedings of the IEEE Computer Security Foundations Workshop, June 1999.

[22] R. C. Merkle "Secure communications over insecure channels" *Communications of the ACM*, 21:294–299, April 1978.

[23] P. Michiardi and R. Molva. "Core: A COllaborative REputation mechanism to enforce node cooperation in mobile ad hoc networks" In Proceedings of the 6th IFIP Communications and Multimedia Security Conference, September 2002.

[24] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. 1924.

[25] M. Osborne and A. Rubenstein. *A Course on Game Theory*. MIT Press, 1994.

[26] K. Park and H. Lee "On the effectiveness of probabilistic packet marking for IP Traceback under denial of service attack" Technical Report No. CSD-00-013, Department of Computer Sciences, Purdue University, June 2000.

[27] R. L. Rivest and A. Shamir and D. A. Wagner "Time-lock puzzles and timed-release crypto" Technical Report No. MIT/LCS/TR-684, 1996.

[28] R. Shirey "Internet security glossary" Internet RFC 2828, May 2000.

[29] Agah, A. and Asadi, M. and Das, S.K. "Prevention of dos attack in sensor networks using repeated game theory". Proceedings of the International Conference on Wireless Networks, 2006.

[30] Agah, A. and Das, S.K., "Preventing DoS Attacks in Wireless Sensor Networks: A Repeated Game Theory Approach". International Journal of Network Security, Vol. 5. No. 2. pp. 145-153. 2007.

[31] Hamilton, S.N. and Miller, W.L. and Ott, A. and Saydjari, O.S. "The role of game theory in information warfare". 4th Information survivability workshop (ISW-2001/2002), 2002.

[32] Dingankar, C. and Brooks, RR. "Denial of Service Games", Proceedings of the Third Annual Cyber Security and Information Infrastructure Research Workshop, pp. 7-17. 2007.

[33] E. Altman and K. Avrachenkov and G. Miller and B. Prabhu. Discrete Power Control: Cooperative and Non-Cooperative Optimization, Proc. of IEEE Infocom 2007, Anchorage, Alaska, USA, May 6-12, 2007.

[34] Mehran Fallah, A Puzzle-Based Defense Strategy Against Flooding attacks Using Game Theory, IEEE Transactions on Dependable and Secure Computing, 12 Feb. 2008.

[35] A. Patcha and J-M. Park. A Game Theoretic Formulation for Intrusion Detection in Mobile Ad Hoc Networks, International Journal of Network Security, Vol. 2, No. 2, 2006, pp. 131-137.

[36] A. Patcha. A game theoretic approach to modeling intrusion detection in mobile ad hoc networks, IEEE Workshop on Information Assurance and Security, June 2004.

[37] T.J. McNevin and J.M. Park, and R. Marchany Chained puzzles: a novel framework for IP-layer client puzzles Wireless Networks, 2005 International Conference on Communications and Mobile Computing, pp. 298-303.

[38] Yi Gao, Willy Susilo, Yi Mu, and Jennifer Seberry, Efficient Trapdoor Based Client Puzzle Against DoS Attacks, Book Chapter in Network Security, 2006

[39] T. J. McNevin, J.M. Park, and R. Marchany. pTCP: A Client Puzzle Protocol For Defending Against Resource Exhaustion Denial of Service Attacks, Technical Report TR-ECE-04-10, Dept. of Electrical and Computer Engineering, Virginia Tech, Oct. 2004.

[40] V.Laurens and A. El Saddik and A. Nayak. Requirements for Client Puzzles to Defeat the Denial of Service and the Distributed Denial of Service Attacks The International Arab Journal of Information Technology. Vol. 3. No. 4., pp. 326-333, 2006.

[41] Lin, C. and Wang, Y. and Wang, Y. and Beijing, PR. A Stochastic Game Nets Based Approach for Network Security Analysis CHINA 2008 Workshop (Concurrency metHods: Issues aNd Applications), pp. 24-35.

[42] Yuanzhuo Wang, Chuang Lin, Yang Yang, Junjie Lv, Yang Qu, A Game-Based Intrusion Tolerant Mechanism for Grid Service,pp.380-386, Fifth International Conference on Grid and Cooperative Computing (GCC'06), 2006

[43] Network Research Foundations and Trends EU FP6 project, Deliverable D4.1 State-of-the-art report on tools and techniques for achieving Autonomous Network Operation, 2006.

[44] Sen, J. and Chowdhury, P.R. and Sengupta, I. A Mechanism for Detection and Prevention of Distributed Denial of Service Attacks, Lecture Notes in Computer Science, Vol. 4308, pp. 139-144, Springer, 2006.

[45] A. E. Goodloe. A foundation for tunnel-complex protocols. PhD Thesis, University of Pennsylvania, 2008.

[46] J. Klensin, Editor. Simple Mail Transfer Protocol, RFC 2821, April 2001.

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

[48] T.K aragiannis, M. Molle, M. Faloutsos, A. Broido. "A nonstationary Poisson view of Internet traffic" INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies Volume 3, Issue, pp. 1558-1569, 7-11 March 2004.

[49] 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

[50] 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.

[51] 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)

[52] 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

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

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

[55] "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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[70] Aura, T. and P. Nikander. “Stateless Connections.” In ICICS’97, LNCS 1334. Springer-Verlag, pp. 87-97, 1997.

[71] Park, K. and H. Lee. “On The Effectiveness Of Probabilistic Packet Marking For IP Traceback Under Denial Of Service Attack.” Tech. Rep. CSD-00-013, Department of Computer Sciences, Purdue University, June 2000.

[72] Ferguson, P. and D. Senie. “Network Ingress Filtering: Defeating Denial Of Service Attacks Which Employ IP Source Address Spoofing.” RFC 2827, May 2000.

[73] Ioannidis, J. and S. M. Bellovin. "Implementing Pushback: Router-based Defense Against DDoS Attacks." In Proceedings of Network and Distributed System Security Symposium, Reston, VA, USA, Feb. 2002, The Internet Society.

[74] Dwork, C. and M. Naor. “Pricing Via Processing Or Combatting Junk Mail.” In Advances in Cryptology. In Proceedings of the Crypto ’92: 12th Annual International Cryptology Conference, Lecture Notes in Computer Science volume 740, pp 139-147, Santa Barbara, California, August 1992. Springer.

[75] Jakobsson, M. and A. Juels. “Proofs Of Work And Bread Pudding Protocols.” In Proceedings of the IFIP TC7 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS ’99), pp. 258-272, Leuven, Belgium, Spetember 1999. Kluwer.

[76] Forristal, J. “Fireproofing Against DoS Attacks.” http:// www.networkcomputing.com/1225/1225f3.html, Network Computing

[77] Mutaf, P. “Defending against a Denial-of-Service Attack on TCP.” In Proceedings of the Recent Advances in Intrusion Detection Conference, 1999.

[78] Ramanathan, A. “WADeS: A Tool For Distributed Denial Of Service Attack Detection.” Thesis at Texas A&M University, August 2002.

[79] Gross, D. and C. M. Harris. *Fundamentals of Queueing Theory*, Wiley-Interscience; ISBN 0471170836

[80] Hoeffding, W. “Probability Inequalities For Sums of Bounded Random Variables”. American Statistical Association Journal, pp 13-30, 1963.

[81] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In Proceedings of Eurocrypt, 2003.

[82] Kerio MailServer - state-of-the-art secure e-mail server, http://www.kerio.com/kms_home.html. 2004.

[83] Postini Enterprise Spam Filtering. The Silent Killer: How Spammers are Stealing Your Email Directory, http://www.postini.com/whitepapers/, June 2004.

[84] Project Honeypot - Distributed system for identifying spammers, http://www.projecthoneypot.org.

[85] Corsaire Limited, Stephen de Vries. Corsaire White Paper: Application Denial of Service Attacks, April 2004, http://www.corsaire.com/white-papers/040405-application-level-dos-attacks.pdf

[86] Steve Gibson. The strange tale of the denial service attack against grc.com. http://www.grc.com/files/grcdos.pdf.

[87] E. Harris. The Next Step in the Spam Control War: Greylisting, http://greylisting.org/articles/whitepaper.shtml, 2003.

[88] S.Hird. Technical Solutions for Controlling Spam. In proceedings of AUUG2002.

[89] Eugene H. Spafford, "Computer Viruses — A Form of Artificial Life?", in Artificial Life II, ed. Langton, Taylor, Farmer and Rasmussen, Addison-Wesley 1992.

[90] Sobig.F Virus Fastest Spreading Ever.
http://www.messagelabs.com/news/virusnews/detail/default.asp?
contentItemId=528, August 20 2003.

[91] The Honeynet project, Know Your Enemy: Learning About Security Threats. Addison-Wesley. 2002.

[92] L. Spitzner, Honeypots: Tracking Hackers, Addison-Wesley, 2002.

[93] C. Endorf, E. Schultz, J. Mellander, Intrusion Detection, Osborne/McGraw-Hill, 2003.

[94] S. Staniford-Chen, B. Tung, P. Porras, C. Kahn, D. Schnackenberg, R. Feiertag and M. Stillma, "The Common Intrusion Detection Framework and Data Formats", 1998.

[95] S.Hird. Technical Solutions for Controlling Spam In the proceedings of AUUG2002, Melbourne, 4-6 September, 2002.

[96] Ronald F. Guilmette, wpoison – small CGI script to combat junk email, http://www.monkeys.com/wpoison/.

[97] Devin Carraway - Sugarplum automated spam-poisoner, http://www.devin.com/sugarplum/.

[98] B. Schneier, Applied Cryptography, John Wiley and Sons,Inc.

[99] G. Tsudik, "Message authentication with One-Way Hash Functions", ACM Computer Communications Review, v. 22, n. 5, 1992, pp. 29-38.

[100] RFC 1341. MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies, 1992.

[101] Sender ID Framework. Microsoft, http://www.microsoft.com/senderid

[102] M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber, "Bankable Postage for Network Services", Proceedings of the 8th Asian Computing Science Conference, Mumbai, India, December 2003.

[103] The Penny Black Project. Microsoft, http://research.microsoft.com/research/sv/PennyBlack/

[104] J. Levine, A. DeKok, et al., "Lightweight MTA Authentication Protocol (LMAP) Discussion and Comparison", Internet Draft, IETF.

[105] SPF: Sender Policy Framework. http://spf.pobox.com/

[106] Pryadkin, Y. and Lindell, R. and Bannister, J. and Govindan, R., An empirical evaluation of IP address space occupancy. USC/ISI, Tech. Rep. ISI-TR-598, 2004.

[107] Census and Survey of the Visible Internet (extended) John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevive Bartlett, and Joseph Bannister Technical Report ISI-TR-2008-649, USC/Information Sciences Institute, February, 2008.

[108] Rajab, M. A., Zarfoss, J., Monrose, F., and Terzis, A. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets (Cambridge, MA). USENIX Association, Berkeley, CA, 2007.

[109] Holz, T., Steiner, M., Dahl, F., Biersack, E., and Freiling, F. 2008. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (San Francisco, California, April 15, 2008). F. Monrose, Ed. USENIX Association, Berkeley, CA, 1-9.

[110] Chen, Z. and Ji, C. 2007. Optimal worm-scanning method using vulnerable-host distributions. Int. J. Secur. Netw. 2, 1/2 (Mar. 2007), 71-80.

The author’s publications in the subject of the dissertation

Journal papers

[J1] B. Bencsáth, I. Vajda, Internet Denial of Service attacks in game theoretical model (in hungarian), Alkalmazott Matematikai Lapok 23, 2006, pp. 335-348.

[J2] B. Bencsáth, I. Vajda, Efficient Directory Harvest Attacks and Countermeasures, International Journal of Network Security, vol 5. no 3. pp. 264-273. 2007.

[J3] Géza Szabó, B. Bencsáth, Protection against DHA attack with central filtering (in hungarian), Híradástechnika, 2006, vol. LXI, pp. pp. 2-9, 05.

[J4] I. Askoxylakis, B. Bencsáth, L. Buttyán, L. Dóra, V. Siris, D. Szili, and I. Vajda Securing Multi-operator Based QoS-aware Mesh Networks: Requirements and Design Options, Wireless Communications and Mobile Computing (Special Issue on QoS and Security in Wireless Networks), accepted for publication in 2009.

Conference papers

[C1] B. Bencsáth, M. A. Rónai Empirical Analysis of Denial of Service Attack Against SMTP Servers, Proceedings of The 2007 International Symposium on Collaborative Technologies and Systems, IEEE, 2007, pp. 72-79.

[C2] B. Bencsáth, L. Buttyán, I. Vajda, A game based analysis of the client puzzle approach to defend against DoS attacks, Proceedings of IEEE SoftCOM 2003 11., Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, 2003, pp. 763-767.

[C3] B. Bencsáth, I. Vajda, A game theoretical approach to optimizing of protection against DoS attacks, presented on the Second Central European Conference on Cryptography (Hajducrypt), July, 2002.

[C4] B. Bencsáth, I. Vajda, Protection Against DDoS Attacks Based On Traffic Level Measurements, 2004 International Symposium on Collaborative Technologies and Systems, The Society for Modeling and Simulation International, 2004, Waleed W. Smari, William McQuay, pp. 22-28., The Society for Modeling and Simulation International, San Diego, CA, USA, January, Simulation series vol 36. no. 1., ISBN 1-56555-272-5.

[C5] B. Bencsáth, The problems and connections of network virus protection and the protection against denial of service attacks, Proceedings of the Networkshop 2004 Conference, NIIF, Hungary, 2004, NIIF, Hungary.

[C6] Géza Szabó, B. Bencsáth, Statistical analysis of the results of the DHA protection system (in hungarian), Proceedings of Networkshop 2006 conference, NIIF, 2006, NIIF.

[C7] B. Bencsáth, I. Vajda, Efficient Directory Harvest Attacks, Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, pp. 62-68., IEEE Computer Society, July 2005.

[C8] B. Bencsáth, Az internetes vírus- és spamvédelem rendszerszemléletben, HISEC 2004 konferencia, 2004, 10., Budapest, in Hungarian.

[C9] B. Bencsáth, Géza Szabó, Components to improve the protection against spam and viruses, HSN LAB Workshop, 2005, Jun.

Other publications

[O1] B. Bencsáth, I. Zs. Berta, Empiric examination of random number generators of smart cards, HTE-BME 2002 Korszerű távközlő és informatikai rendszerek és hálózatok konferencia, BME, 2002, BME.

[O2] B. Bencsáth, I. Vajda, Collecting randomness from the net, Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security 2001, Kluwer, 2001, pp. 105-111, Kluwer, May.

[O3] I. Zs. Berta, B. Bencsáth, Hiteles üzenet küldése rosszindulatú terminálról, NetWorkShop2004, NIIF, CD Proceedings, Győr, 2004.

[O4] B. Bencsáth, S. Tihanyi, Home-made methods for enhancing network security (in Hungarian), Magyar Távközlés, 2000, vol. X, no. 4, pp. 22-27..

[O5] B. Bencsáth, T. Tuzson, B. Tóth, T. Tiszai, G. Szappanos, E. Rigó, Sz. Pásztor, M. Pásztor, P. Papp, P. Orvos, P. Mátó, B. Martos, L. Kún, Z. Kincses, T. Horváth, M. Juhász, B. K. Erdélyi, A. Bogár, G. Vid, Az informatikai hálózati infrastruktúra biztonsági kockázatai és kontrolljai, IHM - MTA-SZTAKI, 2004.

[O6] I. Zs. Berta, I. Vajda, L. Buttyán, B. Bencsáth, T. Veiland, E-Group Magyarország Specification of the Hungarian electronic ID card (HUNEID) Információs Társadalom Koordinációs Tárcaközi Bizottság, Intelligens Kártya Munkacsoport, `http://www.itktb.hu`, 2004

[O7] B. Bencsáth, Simple, free encrypted tunnels using linux, Presented on Networkshop 2000, Gödöllő, Hungary, 2000

[O8] I. Vajda, B. Bencsáth, A. Bognár, Tanulmány a napvilágra került Elender jelszavakról, 2000, Apr. (átvéve: Chip, Alaplap, On-line oldalak)

[O9] B. Bencsáth, S. Tihanyi, Problem areas of the security aspects of network operating systems, Scientific student groups (TDK) 1999

[O10] B. Bencsáth, Multiple Security Flaws Lead to Netenforcer Privilege Escalation (TAR Issue Details), report, BugTraq, http://www.securiteam.com/securitynews/6V00R0K5QY.html, CVE-2002-0399, 2002.

[O11] B. Bencsáth, Sudo Private File Existence Information Leakage Vulnerability, report, Bugtraq http://www.securityfocus.com/bid/321, CAN-1999-1496, 1999.

[O12] A. Szentgyörgyi, G. Szabó, B. Bencsáth. Bevezetés a botnetek világába (in Hungarian), Híradástechnika, 2008. vol. LXIII, Nov. 2008. (Received the HTE Pollák-Virág Award)

[O13] B. Bencsáth, I. Vajda, Trap E-mail Address for Combating E-mail Viruses, Proceedings of IEEE SoftCOM 2004 12. International conference on software, telecommunications and computer networks, University of Split, 2004, pp. 220-224, University of Split, October.

List of Figures

Abbreviations

ADSL: Asymmetric Digital Subscriber Line

DHA: Directory Harvest Attack

DNS: Domain Name System

DDoS: Distributed Denial of Service Attack

DoS: Denial of Service

HTTP: Hypertext Transfer Protocol

IDS: Intrusion Detection System

IRC: Internet Relay Chat - protocol for Internet chat, RFC 1459

MDA: Mail Delivery Agent

MD5: MD5 message-digest algorithm RFC 1321

MTA: Mail Transfer Agent

NTP: Network Time Protocol

OSS: Open Source Software

SOHO: Small Office / Home Office

SSL: Secure Socket Layer protocol (check TLS: RFC 2246)

SSH: Secure shell RFC 4250-4254

SMTP: Simple Mail Transfer protocol RFC 0821

TCL: Tool Command Language (programming language)

TLD: Top-level Domain name

Annex I. SMTP traffic characterstics

The biggest concern regarding the proposed detection algorithm proposed in the second claim (Protection based on traffic analysis against Denial-of-Service problems in Internet e-mail environment) is the consideration that the network traffic is stationary, and the system is able to learn its characteristics. If this is not true, then the algorithm might not work.

To investigate this property, we collected SMTP intensity information from a real-life server dealing with ten thousands of e-mails daily. The duration of the data collection was 6 month long. The intensity was calculated in 1 minute long windows. We expected stationarity of the traffic intensity by a daily base, e.g. on every day the distribution of SMTP traffic intensities will be very similar. The data was then analyzed by statistical tools.

We have not expected that every daily intensity graph will be similar, instead we expected some seasonality of the graphs (e.g. lower traffic during the weekends), and also, we expected days with very special, odd intensity graphs. The goal of our investigation was to observe at least the basic characteristics of real-life servers, to decide if the proposed algorithm is feasible in real-life.

The statistical analysis was based on Kolmogorov-Smirnov test with $\alpha = 0.05$ parameter. The daily intensity distribution graphs of 6 month were tested with the test and the possible pairs were selected. On the Figure 20, we show the clusters of the days with very similar traffic intensity graph. On this figure, every edge means that they were found similar by the Kolmogorov-Smirnov test. This shows, that only about 12 clusters were defined. That means, that at least at some extent the intensity distribution graphs have similarities and this proves the feasibility of some kind of self-learning algorithm, such as we proposed in this section.

For additional information, from these clusters we chose individual days (12 daily graphs altogether) and produced the traffic intensity graph of these days, shown on Figure 21, Figure 22, Figure 23, Figure 24, Figure 25 and Figure 26. These figures also understate that although the there are huge differences in different daily traffic intensity graphs, our proposed detection method seems to be feasible.

As our primary goal was to investigate if the protection algorithm is feasible in real-life environment. We do not make any detailed analysis of the graphs, but it seems, that even if the intensity graphs seem very different, the contains similarities at higher intensities, e.g. (in case of the protection described in Section 4.10) if M is set to 250, then on most graphs, the average probability of false positives is very low.

The detailed analysis of the traffic intensities and further modification of the algorithms in real-life environments remains a possible future work.

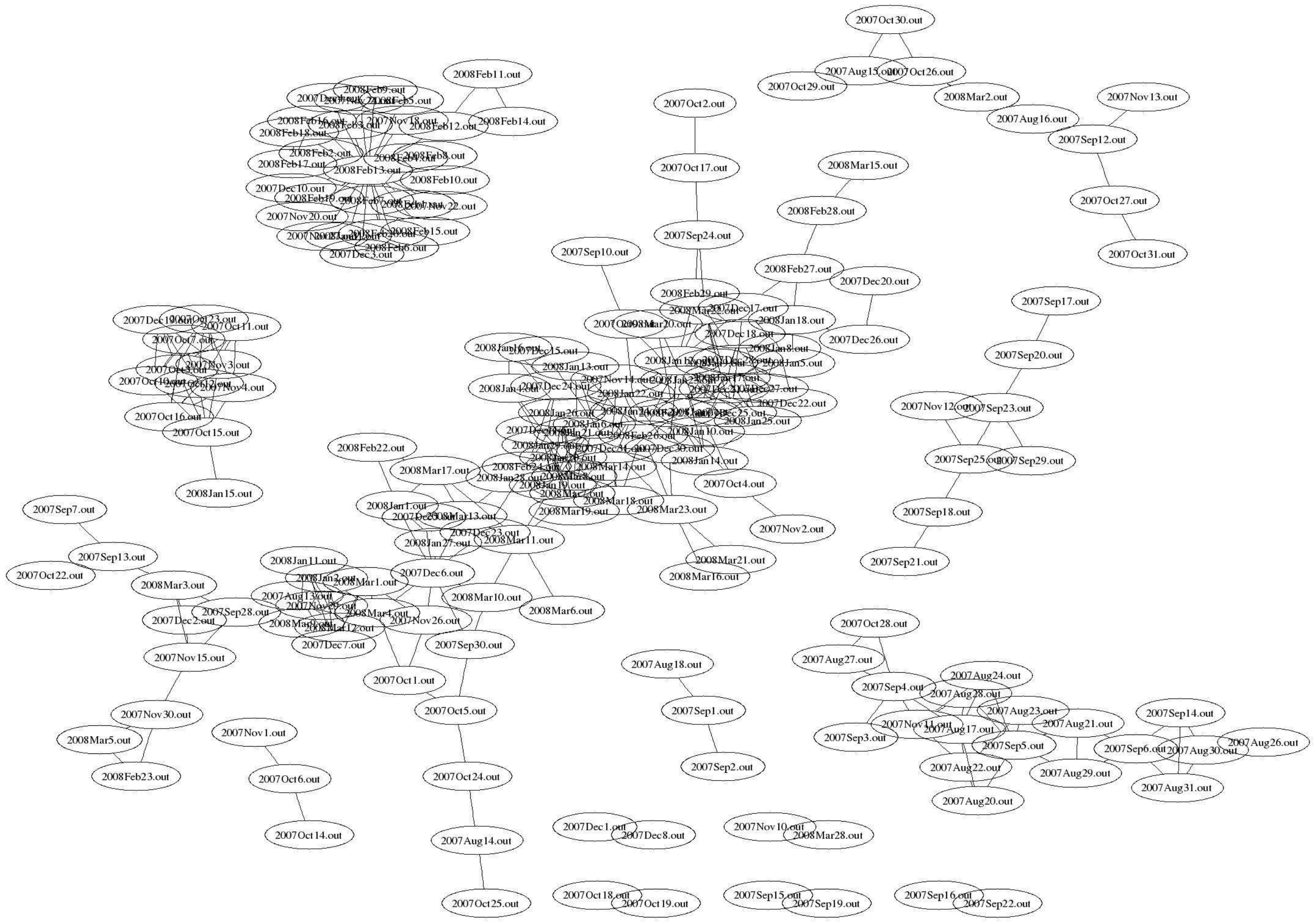


Figure 20: KsPairs

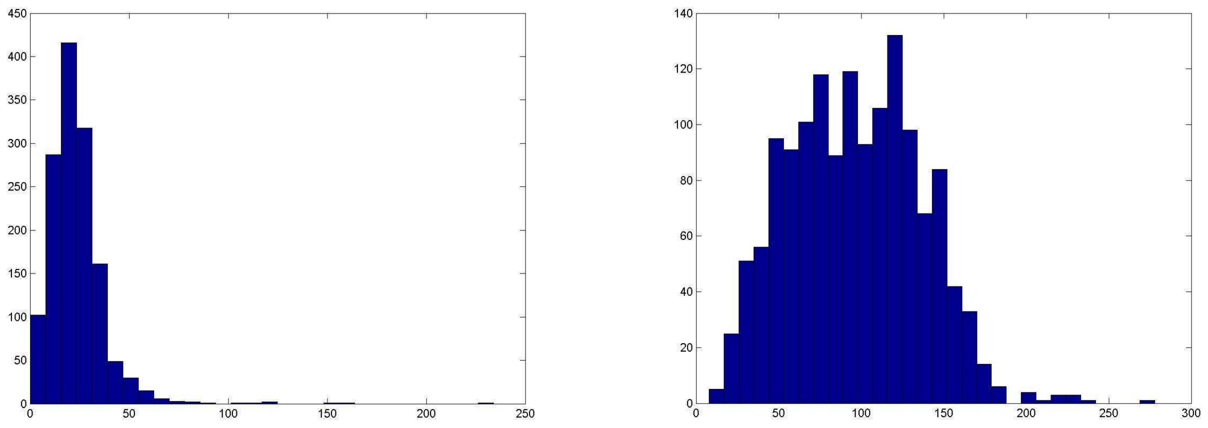


Figure 21: Intensity block 1,2 - 11/02/2008, 15/01/2008. X: intensity Y: occurences

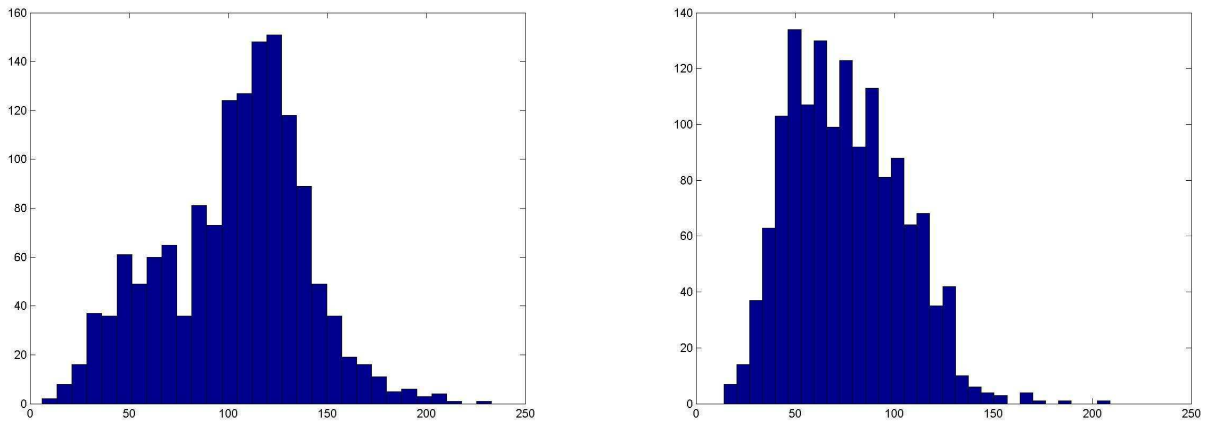
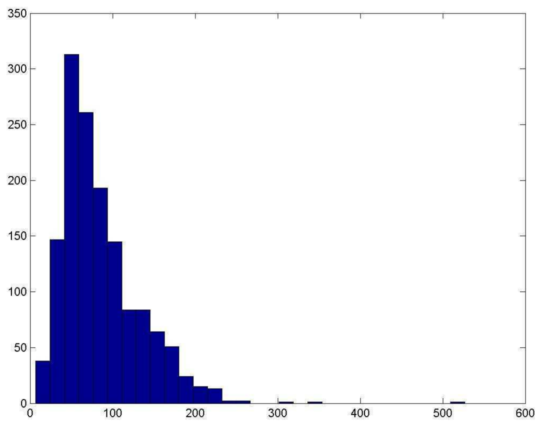


Figure 22: Intensity block 3,4 - 02/10/2007, 30/10/2007. X: intensity Y: occurences



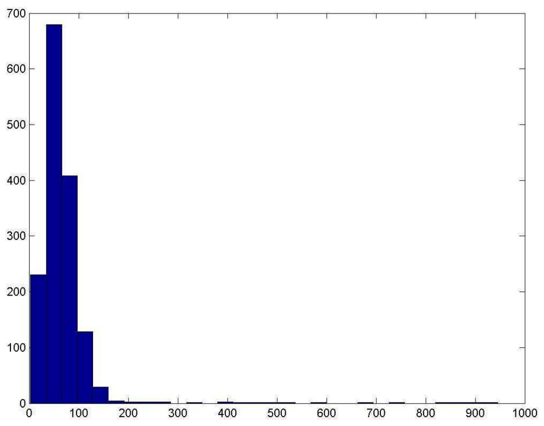
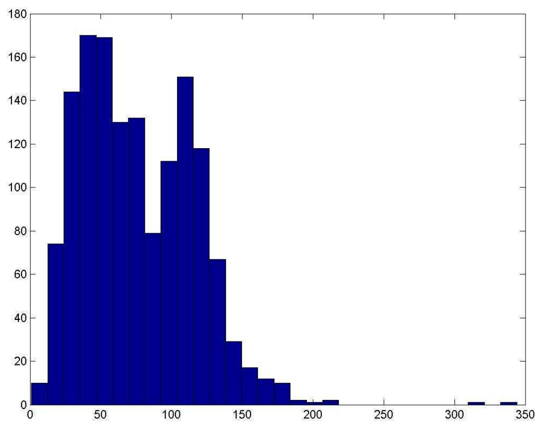


Figure 23: Intensity block 5,6 - 17/09/2007, 28/10/2007. X: intensity Y: occurences



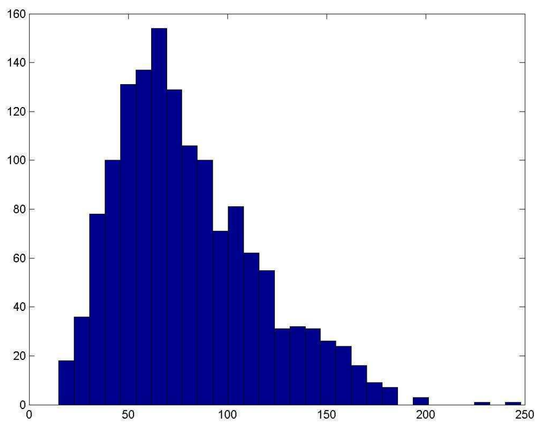


Figure 24: Intensity block 7,8 - 01/11/2007, 01/12/2007. X: intensity Y: occurences

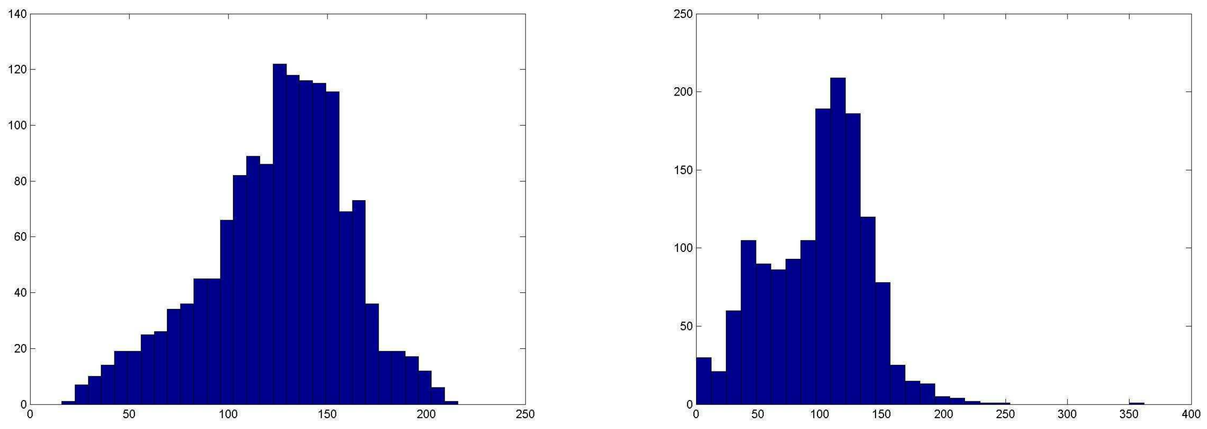


Figure 25: Intensity block 9,10 - 10/11/2007, 18/10/2007. X: intensity Y: occurences

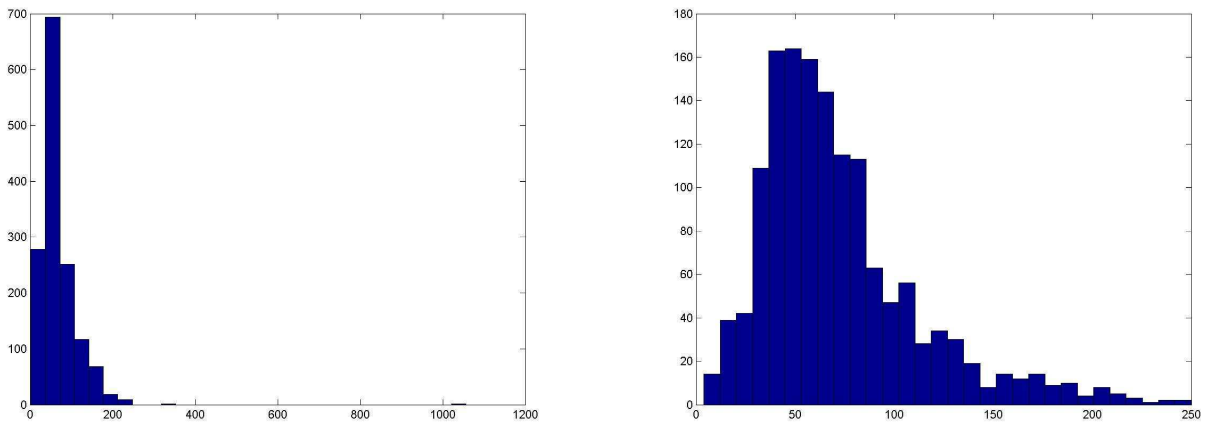


Figure 26: Intensity block 11,12 - 15/09/2007, 16/09/2007. X: intensity Y: occurences