Bayes Optimal DDoS Mitigation by Adaptive History-Based IP Filtering

Markus Goldstein*, Christoph Lampert*, Matthias Reif*, Armin Stahl* and Thomas Breuel*†
*German Research Center for Artificial Intelligence DFKI GmbH
Research Group Image Understanding and Pattern Recognition (IUPR)
D-67663 Kaiserslautern, Germany
†Technical University of Kaiserslautern
Department of Computer Science
D-67663 Kaiserslautern, Germany
Email: {goldstein,lampert,reif,stahl,breuel}@iupr.dfki.de

***Abstract*—Distributed Denial of Service (DDoS) attacks are today the most destabilizing factor in the global internet and there is a strong need for sophisticated solutions. We introduce a formal statistical framework and derive a Bayes optimal packet classifier from it. Our proposed practical algorithm "Adaptive History-Based IP Filtering" (AHIF) mitigates DDoS attacks near the victim and outperforms existing methods by at least 32% in terms of collateral damage. Furthermore, it adjusts to the strength of an ongoing attack and ensures availability of the attacked server. In contrast to other adaptive solutions, firewall rulesets used to resist an attack can be precalculated before an attack takes place. This ensures an immediate response in a DDoS emergency. For evaluation, simulated DDoS attacks and two real-world user traffic datasets are used.**

I. Introduction

Distributed Denial of Service attacks (DDoS) have become a major threat in the internet today. Large scaled networks of infected PCs (bots or zombies) combine their bandwidth and computational power in order to overload a publicly available service and denial it for legal users. Due to the open structure of the internet, all public servers are vulnerable to DDoS attacks. The bots are usually acquired automatically by hackers who use software tools to scan through the network, detecting vulnerabilities and exploiting the target machine.

DDoS attacks have become more and more frequent in the last years. The attacks against large e-commerce sites in February 2000 [1] and the attacks against root DNS servers in 2003 [2] and 2007 have drawn public attention to the problem of DDoS attacks. Today, mainly mid-sized websites are attacked by criminals in order to extort protection money from their owners without attracting too much public attention [3]. Besides that, also Internet Service Providers (ISP) have to deal with the problem that DDoS traffic is congesting their link bandwidths [4].

Also the bot software evolved alarmingly over time. Early tools like *TFN*, *Stacheldraht*, *Trinoo* or *Mstream* used unencrypted and hierarchically organized communication structures. Most of these tools used TCP-SYN, UDP or ICMP floods with possibly identifiable parameters. Since some of these attacks have successfully been mitigated, a new generation of bots arose. *SDBot*, *Agobot* or the very enhanced *Phatbot* are known representatives which use IRC as a robust and secure communication [5]. These tools also contain methods for spreading themselves and have more sophisticated attack algorithms, which could be upgraded over the internet. The attack traffic from those tools looks like legal traffic on the transport layer, which makes it nearly impossible to filter it effectively with standard firewalls.

Mitigating DDoS attacks at the origin or within the core of the internet seems to be an impossible task due to the distributed and authorization-free nature of the IP based network. Approaches to achieve this objective typically rely on changing current internet protocols [6], [7] and are therefore not easily applicable.

Ingress filtering as described in RFC 2827 [8] also helps mitigating DDoS attacks with forged source IP addresses (IP spoofing) and should be applied by every ISP. Since ingress filtering only helps other ISPs on the internet and not the one who is actually applying it, it took quite a long time until it was setup in many places. Furthermore, Savage et al. [9] suggested IP Traceback to find the source of spoofed IP addresses by probabilistically marking packets. Nowadays, IP spoofing is not that common any more in DDoS attacks, except for the last octet of an IP address.

Today, there is a strong need to mitigate DDoS attacks near the target, which seems to be the only solution to the problem in the current internet infrastructure. The aim of such a protection system is to limit their destabilizing effect on the server through identifying malicious requests.

After discussing the problem of near target DDoS mitigation in Section II in detail, we give a short overview of Bayesian Decision Theory in Section III. Then, we introduce a novel Bayes optimal framework for implementing DDoS mitigation systems in Section IV and V. In order to demonstrate the practical impact of our framework, we present results of a first experimental evaluation based on real-life and simulated data in Section VI. Finally, we conclude in Section VII with a discussion of our system and an outlook of possible improvements.

II. Near Target DDoS Mitigation

In the following, we focus on DDoS mitigation, which relies on a successful DDoS detection algorithm. Carl et al. [10] give a survey about recent DDoS detection research.

Only few solutions on near target DDoS mitigation have been proposed [11], [12], [13], [14], [15], [16]. Peng et al. [17] propose a mechanism called *History-based IP Filtering (HIF)*, which keeps track of previously seen hosts and is effective in large scaled attacks. Pack et al. [18] also use the idea of modelling historic traffic, but extends this approach with clustering observed traffic and creating a limited number of *Access Control Lists (ACL)* heuristically. This approach is very compute intensive and therefore limited to a small historic time span.

In this paper, we propose a new method to mitigate DDoS attacks based on observed data. The system does not aim at the detection of such attacks, but at the automatic creation of filter rules for IP firewalls like *iptables*, *nf-HiPAC* or *Cisco ACLs* that will allow the server to continue serving legitimate users even while under a large scaled DDoS attack.

In contrast to all previous approaches, we rely on a rigorous statistical framework of Bayesian decision theory. This allows us to abstract from the implementational details and obtain theoretical predictions about performance, identifying decision rules which are optimal in the chosen scenario. The main contributions of our work are (1) we present a statistical framework for mitigating DDoS attacks based on Bayesian decision theory to infer optimal rules and (2) we present a new method based on this framework called Adaptive History-Based IP Filtering (AHIF), which minimizes collateral damage (blocking legal users) during near target DDoS mitigation.

Unlike other approaches, our method can be used to prepare certain filter rule sets before a DDoS attack takes place. Not analyzing IP traffic during an attack in order to mitigate it has a number of advantages: (1) The mitigation can take place immediately once an attack has been detected. (2) The resources of the firewall system can be used completely for attack packet filtering and (3) Rules can be propagated to distributed firewalls in order to filter out DDoS traffic at the borders of an ISP core network (*edge router*).

III. Bayesian Decision Theory

In this section we will give a short overview of the statistical background that we need, following the notation of Duda et al. [19].

In the following, the term "request" is not necessarily associated with a user request, it is a request in an abstract way. It might be an IP packet captured on the wire, an e-mail, which is delivered, or a HTTP request sent to a webserver. Since we will use Apache logfiles as our primary data source in Section VI, we refer to HTTP requests pointed to our target server.

Requests are modeled as discrete random variables x_i from a feature space X. We assume that they are independently and identically distributed. Each request is either caused by a legal user request or by an attack attempt, and we denote this origin by assigning a label $y_i \in \{\mathtt{L}, \mathtt{I}\}$ to each request, where `L` stands for `legal` and `I` for `illegal`.

During normal operation, all requests come from legal users and follow a probability distribution $P(x|\mathtt{L})$. In the case of an attack, additional attack requests reach the host following a distribution $P(x|\mathtt{I})$. To the server, the labels y_i are unknown, and it can only measure the raw arriving requests and their distribution $P(x)$. This will be a mixture between $P(x|\mathtt{L})$ and $P(x|\mathtt{I})$, but the mixing coefficients are unknown.

In this setup, filtering can be stated as a classification task: for each request it must be decided whether to `accept` (`A`) or `reject` (`R`) it. Each such decision causes a certain loss, where this term is used in an abstract sense, not necessarily referring to a financial value or IP packet loss.

At the time of decision, it is unknown if an arriving request is legal or illegal, and we therefore cannot determine the exact loss. However, in a statistical sense we can calculate the expected loss for classifying a request x:

$$\begin{aligned}\mathrm{loss}(\mathtt{R}|x) &= \lambda(\mathtt{R}|\mathtt{L})P(\mathtt{L}|x) + \lambda(\mathtt{R}|\mathtt{I})P(\mathtt{I}|x)\\ \mathrm{loss}(\mathtt{A}|x) &= \lambda(\mathtt{A}|\mathtt{L})P(\mathtt{L}|x) + \lambda(\mathtt{A}|\mathtt{I})P(\mathtt{I}|x)\end{aligned}$$

where $\lambda(\omega|y)$ is the loss that is associated with deciding for class $\omega \in \{\mathtt{A}, \mathtt{R}\}$ for a sample that is of type $y \in \{\mathtt{L}, \mathtt{I}\}$. $P(\mathtt{L}|x)$ and $P(\mathtt{I}|x)$ are the *posterior* probabilities, i.e. the probabilities for a request x reaching the server to be of type `L` or `I`.

Whether the system accepts or rejects a request can be stated as a decision function $\alpha : X \to \{\mathtt{R}, \mathtt{A}\}$. Any such function comes with an expected overall loss, called the *risk*:

$$\mathrm{risk}(\alpha) = \sum_{x \in X} \mathrm{loss}(\alpha(x)|x)P(x).$$

Bayesian decision theory tells us that the best choice of a classifier (decision function) is one that minimizes $\mathrm{risk}(\alpha)$.

IV. Bayes Optimal Filtering of Network Packets

A. Classification Risk

In the setup of filtering requests, the loss that we associate with our decision is asymmetric. Accepting legal requests and rejecting malicious requests is the targeted behavior, and we associate a loss of 0 with it.

Accepting malicious requests may lead to a loss $\epsilon \in [0, 1]$, if the server is beyond its resources. But as long as the server has enough resources, there is no downside in accepting a request by an attacker. Therefore, we assign the loss of $\epsilon = 0$, but only under the constraint, that the server load (or the bandwidth usage) does not exceed its capacity.

On the other hand, it is always damaging to reject a request by a legal user and we assign a positive loss with that. Since the units in which we measure this loss are arbitrary, we can normalize the resulting loss matrix λ to be

$\lambda(\omega\|y)$	`legal`	`illegal`
`accept`	0	ϵ
`reject`	1	0

For the classification risk of a request directed to a server running below its capacity, we then obtain

$$\mathrm{risk}(\alpha) = \sum_{\{\alpha(x)=\mathtt{R}\}} P(\mathtt{L}|x)P(x), \tag{1}$$

which we can rewrite using Bayes rule as

$$\mathrm{risk}(\alpha) = P(\mathtt{L}) \sum_{\{\alpha(x)=\mathtt{R}\}} P(x|\mathtt{L}). \tag{2}$$

$P(\mathtt{L})$ denotes the *class prior*. It does not depend on the actual requests or the classifier and therefore stays constant.

To summarize, a Bayes optimal packet filter minimizes $\mathrm{risk}(\alpha)$ while at the same time adhering strictly to the global constaint that the server can only handle a limited number of requests per time unit. This Bayesian risk is also referred to as *collateral damage* [18].

B. Optimal Classifier

To construct such an optimal filter, we first look at the situation of static filtering: a number of requests $x_1, \ldots, x_M$ arrive within a certain time span, but the server is only able to handle N events out of those. From Equation (2) we see that the empirical risk, i.e. the excepted loss on this sample set is

$$\mathrm{risk}(\alpha) = P(\mathtt{L}) \sum_{\{\alpha(x_i)=\mathtt{R}\}} P(x_i|\mathtt{L}) \tag{3}$$

Since all terms in Equation (3) are positive, each dropped request (i.e. x_i with $\alpha(x_i) = \mathtt{R}$) can only increase the risk. It is therefore advisable to drop only as many requests as necessary to prevent a server overload, that is $M - N$.

Also from Equation (3) we see that the higher the class conditional density $P(x_i|\mathtt{L})$ for a request, the more the risk increases when dropping it.

The optimal classifier α^* is therefore given by

$$\alpha^*(x_i) := \begin{cases} \texttt{reject} & \text{if } x_i \text{ is one of the } M-N \\ & \text{requests with lowest } P(x_i|\mathtt{L}), \\ \texttt{accept} & \text{otherwise,} \end{cases} \tag{4}$$

In other words, this is the classifier that let only N packages with highest $P(x_i|\mathtt{L})$ pass.

C. Practical Filtering

In practice, dropping requests has to be performed depending on the server load, i.e. each request has to be decided on without information about which and how many other requests will arrive in the future.

Instead of comparing all requests by their $P(x_i|\mathtt{L})$ value, we have to set a variable threshold θ on this quantity and define a parametric family of classifiers

$$\alpha_\theta(x) := \begin{cases} \texttt{accept} & \text{if } P(x|\mathtt{L}) \geq \theta, \\ \texttt{reject} & \text{if } P(x|\mathtt{L}) < \theta, \end{cases} \tag{5}$$

A high value for theta, i.e. $\theta \geq \max(P(x_i|\mathtt{L}))$, will cause all requests to be rejected, including the legal ones. For $\theta = 0$, every request will be accepted, including all the malicious ones. For a very small θ, e.g. $\theta \to 0$, every request with $P(x_i|\mathtt{L}) > 0$ will be accepted. For $\theta \to 0$, our approach is comparable to HIF proposed by Peng et al. [17].

However, a low value like $\theta \to 0$ yields a classifier that accepts too many requests in the case of an attack, thus not preventing the overload. $\theta = 0$ can be considered as the normal operation mode of the mitigation system without dropping any connections, whereas during an ongoing attack, a higher, but not too restrictive threshold is better. To find the optimal threshold value, we adjust the parameter in an adaptive process: starting from $\theta = 0$, we monitor the requests passing the mitigation system. If more requests are measured than the server could constantly handle, we raise the threshold. If $\theta > 0$ and the measured connections are below the limit, we lower the threshold. For a constant attack strength, this scheme converges to a limit θ^*, where the resulting classifier α_{θ^*} is tuned to let the server operate just below its maximum acceptable load level. If the attack load varies over time, θ will adapt to it, keeping the server always below its capacity limit.

We will show next that the expected performance of α_{θ^*} is in fact the same as for optimal classifier α^*. For this, we look again at a unit time span with M samples x_i arriving. α_{θ^*} will filter them immediately, and because of the way θ^* was chosen, on average it will accept just as many requests as the server can handle, i.e. N out of M. Also by construction, all accepted requests have a higher $P(x_i|\mathtt{L})$ value than all rejected ones.

α_{θ^*} therefore shows the same filtering behavior as α^*: it accepts the N requests with highest likelihood $P(x_i|\mathtt{L})$ and reject the remaining $M - N$ ones.

D. Estimating $P(x|\mathtt{L})$

As it is obvious from the previous section, it is crucial to know $P(x|\mathtt{L})$ for building a Bayes optimal classifier. Fortunately, this quantity is relatively easy to obtain, since it depends only on the distribution of legal user requests and does not require any information on a possible upcoming attack.

Generally, pattern recognition provides many ways and algorithms to estimate $P(x|\mathtt{L})$ from previously observed data. For a first evaluation of our framework, we estimate $P(x|\mathtt{L})$ directly from a histogram count. Given a large number of training samples $x_1, \ldots, x_K$, i.e. requests from a characteristic server observed without ongoing DDoS attacks, we estimate

$$P(x|\mathtt{L}) \approx \frac{1}{K} \left|\{i : x_i = x; i = 1, \ldots, K\}\right|, \tag{6}$$

where $|.|$ denotes the number of elements in the set. If the number of training is too small for a reliable histogram count, one can rely on smoothing or interpolation or any other method for non-parametric density estimates, but one has to be aware that —depending on the representation chosen for x— this only might be a heuristic procedure.

V. Finding Filter Rules

In a practical implementation, the classifier α must be efficiently representable. In our scenario we want a set of

firewall instructions for the *nf-HiPAC* firewall.

This imposes two restrictions: (1) Our feature vector x must only rely on features which can be efficiently calculated from the requests and checked by firewall rules. (2) The number of rules a system can process in real time is limited.

First, we choose the most significant $n = 24$ bit of the sender IP as unique feature from a request (alternatives will be discussed in Section VII). The resulting feature space has 2^{24} entries. The reason for neglecting (at least) the last 8 bit of the sender IP has two main reasons: (1) IP addresses can be spoofed easily on the last 8 bits by bots like current malware does and (2) many users often use different IPs within a 24 bit network mask (due to changing hosts or using DHCP).

Still this approach could result in 2^{24} rules whereas a firewall system operates usually with a maximum amount well below this value. We therefore only use positive Access Control Lists (ACL) and aggregate rules in order to reduce their number. A binary tree with a maximum depth of n is created, each leaf represents one "accept" ACL of a n-bit network. Then, the tree is processed bottom up and whenever two leaves share the same father, both are deleted (pruning). The remaining father node automatically becomes a new leaf node (new ACL) with an one bit smaller network mask. If there are only nodes with one descendent left, the minimum number of rules has been found.

For example, if there are two leaf nodes, representing the networks 131.246.64.0/24 and 131.246.65.0/24 they are pruned and result in a new leaf node 131.246.64.0/23.

This algorithm results in the minimum number of rules for our given feature based on n and can be considered as a "lossless compression". Please note, that this is the upper boundary for rules, which may occur. By choosing the θ^*, the actual number of rules in the firewall might be lower.

It might turn out, that the number of rules is still too much to be applicable on a firewall. In this case we have to use a "lossy compression": We select one bit less of the sender's IP address as our unique feature and rebuild the tree with $n = n - 1$. This process is iterative until it results in an appropriate upper boundary for applicable firewall rules. But having less rules comes at a price: the classifier α will move away from the Bayes optimal classifier α^*.

Since the threshold θ^* depends on the rate of incoming traffic, we perform the rule generation for different discrete values of θ. In the case of an attack, these precomputed rulesets can be loaded and activated individually. For finding the most appropriate rule, one can use the adaption procedure as previously described in Section IV-C for θ^*.

VI. Results

A. Datasets and simulated bot network

Two datasets have been used to illustrate the effectivity of AHIF and compare it with other approaches. Both datasets comprise 100 days of Apache logfile data.

The first dataset (DS-1) was obtained by the webserver www.xvid.org, containing 53,828,308 requests from 1,284,213 different IP addresses. The addresses are highly distributed over all IP ranges from 200 different countries. On average, there are 6.23 HTTP requests per second (rps) with a maximum peak of 149 rps. We assume, that the server can handle a maximum request rate of 3,000 rps.

Our second dataset (DS-2) contains data from an internationally operating web-community project. In total, 8,464,608 requests were observed with 144,995 different IP addresses, which corresponds to 0.98 rps in average. As the website was announced in the media, there was a flash crowd event, with a peak of 325 rps. In DS-2, there are more recurring regular users (IPs) and they sent in average 28.2% more requests than in DS-1. For this server, we assume a maximum capacity of 1,000 rps.

Generation of attack traffic is done by simulating a bot network comprising of 100,000 bots. All together, they are able to send about 40,000 rps, which is clearly over both server limits and therefore achieving a successful DDoS attack. Due to the lack of real bot network data, we assumed, that the source IP addresses of bots are randomly distributed over the whole IP space. IP addresses for bots are selected such that they do not belong to private networks, multicast addresses or to networks, which are not assigned by the IANA (IANA-reserved pool).

B. AHIF

In both datasets, the first 90 days are used as training data to obtain the histogram counts and the last 10 days are used for testing purposes. Within the 10 days of testing, we launch our attack and studied the effectivity of our approach. To obtain $P(x|\mathtt{L})$ in the first 90 days, 48,633,275 requests in DS-1 and 7,628,328 requests in DS-2 are used.

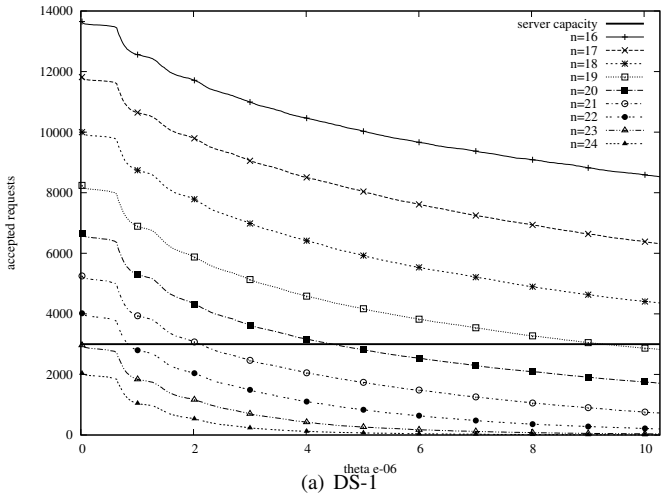
To obtain the maximum number of possible firewall rules, we used a standard PC with 2x Opteron 270 CPU, 4GB of RAM running a Linux kernel 2.6.13.5 and nf-HiPAC version 0.9.1. The test setup was able to process about 100,000 firewall rules (filtering source IP addresses) at a rate of about 40,000 packets per second.

For each n, the corresponding θ^* could be found at the intersection of the server capacity line and the curve in Figure 1. We could easily see, that the best choice would be to use $n = 24$, because it has the lowest θ^* for all curves. But as already mentioned in Section V, we are restriced in choosing n, which depends on the maximum number of applicable rules. In Figure 2 the dependency of θ and the number of rules is shown. It can be seen, that $n = 21$ fits the rule constraint for DS-1. Referring to Figure 1, it can be seen, that the best θ^* in this case is $2.12 \cdot 10^{-6}$.

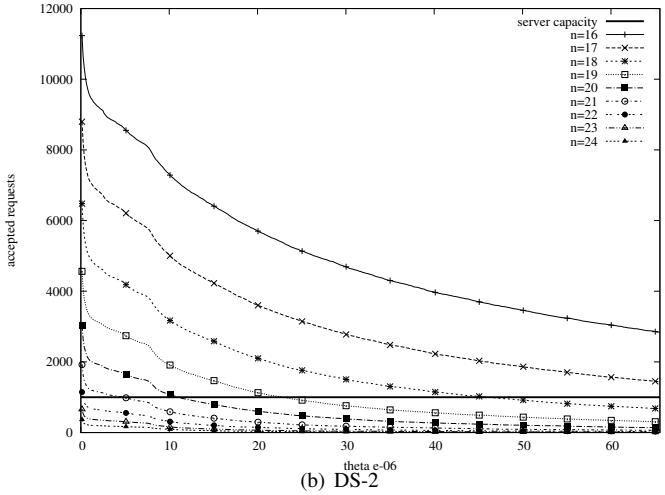
The obtained optimal $n = 21$ and $\theta^* = 2.12 \cdot 10^{-6}$ are used to compare our results in the following Section for DS-1. For DS-2, the firewall rule constraint does not apply in our case and we can select $n = 24$ and selecting an arbitrary small θ^*.

C. Comparing AHIF

As a baseline (BASE), we use the policy "drop requests randomly such the server limit constraint is fullfilled". Furthermore, we compare our method AHIF with HIF as proposed

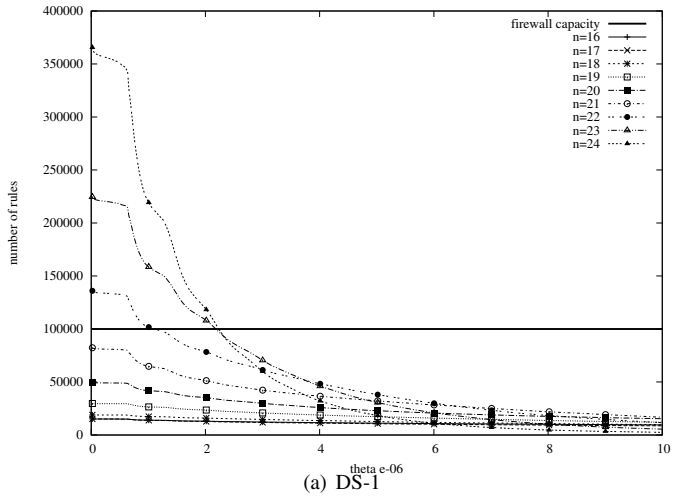


(a) DS-1

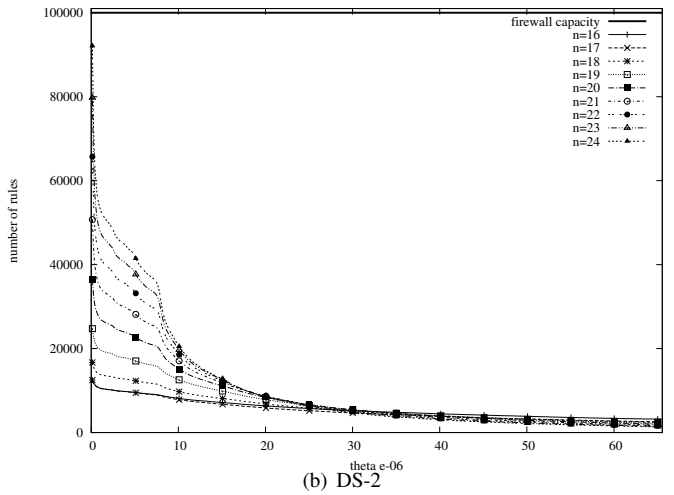


(b) DS-2

Fig. 1. Simulation Results. The number of requests passing the firewall during an attack depending on the decision boundary θ are plotted. Each curve represents a different netmask in bits used as primary feature. The horizontal red line shows the server limit.



(a) DS-1



(b) DS-2

Fig. 2. Simulation Results. The number of resulting firewall rules is plotted depending on θ and the used netmask n as feature. In DS-1, the horizontal line shows the limit of applicable firewall rules, in DS-2 we are below the limit for every n.

TABLE I
COMPARING COLLATERAL DAMAGE (DENYING LEGAL USERS) OF DIFFERENT APPROACHES. THE HIGHER VALUES FOR COLLATERAL DAMAGE IN DS-2 ARE LIKELY DUE TO THE LOWER NUMBER OF TRAINING SAMPLES WITHIN THE SAME TIME SPAN.

	DS-1	DS-2
BASE (random)	4805457 (92.50%)	773559 (92.50%)
HIF (Peng et al.)	4283645 (82.46%)	572319 (68.44%)
AHIF (this paper)	768569 (14.79%)	389112 (46.53%)

by Peng et al. [17] as illustrated in Table I. Unfortunately, we were not able to compare our method with the clustering solution from Pack et al. [18], because the clustering would at least require 66.0GB of memory with our smaller dataset DS-2 and 90 days of training.

VII. DISCUSSION

A. Discussion of Results

The results show that our method is superior on both datasets. A big advantage of our method is that our firewall rules are prepared before a DDoS attack takes place. This allows network operators (or an automated system) to react fast to detected attacks. Through adapting the decision boundary θ, it is also possible to mitigate attacks of changing strength.

BASE might be considered as a method currently applied by system administrators, which rejects new connections if the server limit has been reached. This is a widely used method today, where users can see "Service Temporarily unavailable" or similar interim pages.

HIF already decreases the collateral damage, but cannot prevent many legal users from being excluded. Our adaptive method performs very good on the large dataset DS-1 and also outperformes HIF based on DS-2.

The reason, why our method performs better on DS-1, might be due to the fact that there are more than 6 times as many training samples in DS-1.

To summarize, our proposed algorithm is effective in mitigating highly distributed DDoS attacks and minimizes collateral damage.

B. Future Outlook

We showed that our method is Bayes optimal in our scenario chosen, but to make it applicable on the limited hardware of an internet firewall, we had to rely on simplifying assumptions. For estimating $P(x|\mathtt{L})$, we only used histograms of historic data. $P(x|\mathtt{L})$ might be estimated by other features as well, not necessarily referring to the source IP address. One might think about using features like request rates, OS fingerprints, country or time zone information. This would increase the accuracy for estimating $P(x|\mathtt{L})$ for hosts, which are unknown to the server at the time of the attack and therefore increase filter accuracy.

The advantage of our statistical treatment is that it is generic and does not make use of specific properties of the features chosen. The system can therefore easily be adapted to include other, possibly more descriptive features in the future. Also, we can easily integrate prior knowledge, e.g. if certain attack patterns are known, the features can be chosen to reflect this. This also carries over to the estimation process for $P(x|\mathtt{L})$ itself. Instead of a static procedure like the histogram count, a higher order estimator like a Bayesian network can be used in order to optimally combine multiple features. We are currently investigating this field.

ACKNOWLEDGMENT

This work is part of NetCentric Security, a project of Deutsche Telekom Laboratories supported by German Research Center for Artificial Intelligence DFKI GmbH.

REFERENCES

[1] A. Harrison, "Cyberassaults hit Buy.com, eBay, CNN, and Amazon.com," February 2000,
Available at:
http://www.computerworld.com/printthis/2000/0,4814,43010,00.html.

[2] P. Vixie, G. Sneeringer, and M. Schleifer, "Events of 21–Oct–2002," November 2002, available at: http://d.root-servers.org/october21.txt.

[3] D. McPherson, C. Labovitz, and M. Hollyman, "Worldwide infrastructure security report," 2007, available at: http://www.arbornetworks.com/report.

[4] D. Pappalardo and E. Messmer, "Extortion via DDoS on the rise," May 2005, available at: http://www.networkworld.com/news/2005/051605-ddos-extortion.html?page=1.

[5] D. Dittrich, "Distributed Denial of Service (DDoS) Attacks/tools," available at: http://staff.washington.edu/dittrich/misc/ddos/.

[6] J. Leiwo, P. Nikander, and T. Aura, "Towards network denial of service resistant protocols," in *In Proceedings of the 15th International Information Security Conference (IFIP/SEC 2000)*, 2000.

[7] M. Handley and A. Greenhalgh, "Steps towards a DoS-resistant internet architecture," in *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. New York, NY, USA: ACM Press, 2004, pp. 49–56.

[8] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing," United States, 2000, available at: http://rfc.net/rfc2827.html.

[9] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson, "Practical network support for IP traceback," in *SIGCOMM*, 2000, pp. 295–306.

[10] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet Computing*, vol. 10, no. 1, pp. 82–89, 2006, available at: http://doi.ieeecomputersociety.org/10.1109/MIC.2006.5.

[11] R. K. Chang, "Defending against Flooding-Based Distributed Denial-of-Dervice Attacks: A Tutorial," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 42–51, 2002.

[12] M. P. Collins and M. K. Reiter, "An empirical analysis of target-resident dos filters." in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2004, pp. 103–114.

[13] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed ddos traffic," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2003, pp. 30–41, available at: http://doi.acm.org/10.1145/948109.948116.

[14] T. K. T. Law, J. C. S. Lui, and D. K. Y. Yau, "You Can Run, But You Can't Hide: An Effective Statistical Methodology to Trace Back DDoS Attackers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 9, pp. 799–813, 2005, available at: http://dx.doi.org/10.1109/TPDS.2005.114.

[15] O. Paul, "Improving Web Servers Focused DoS Attacks Detection," in *Proceedings of the IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006)*, Tuebingen, Germany, 2006.

[16] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending Against Distributed Denial-of-Service Attacks With Max-Min Fair Server-Centric Router Throttles," *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 1, pp. 29–42, 2005, available at: http://dx.doi.org/10.1109/TNET.2004.842221.

[17] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from Distributed Denial of Service attack using history-based IP filtering," in *Proceedings of the IEEE International Conference on Communications (ICC 2003)*. Anchorage, AL, USA: IEEE, 2003.

[18] G. Pack, J. Yoon, E. Collins, and C. Estan, "On Filtering of DDoS Attacks Based on Source Address Prefixes," in *Proceedings of the 2nd International Conference on Security and Privacy in Communication Networks (SecureComm 2006)*. IEEE, 2006.

[19] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.