Analysis of DDoS Detection Systems

Michael Singhof
Heinrich-Heine-Universität
Institut für Informatik
Universitätsstraße 1
40225 Düsseldorf, Deutschland
singhof@cs.uni-duesseldorf.de

ABSTRACT

While there are plenty of papers describing algorithms for detecting distributed denial of service (DDoS) attacks, here an introduction to the considerations preceding such an implementation is given. Therefore, a brief history of and introduction to DDoS attacks is given, showing that these kind of attacks are nearly two decades old. It is also depicted that most algorithms used for the detection of DDoS attacks are outlier detection algorithms, such that intrusion detection can be seen as a part of the KDD research field.

It is then pointed out that no well known and up-to-date test cases for DDoS detection system are known. To overcome this problem in a way that allows to test algorithms as well as making results reproducible for others we advice using a simulator for DDoS attacks.

The challenge of detecting denial of service attacks in real time is addressed by presenting two recently published methods that try to solve the performance problem in very different ways. We compare both approaches and finally summarise the conclusions drawn from this, especially that methods concentrating on one network traffic parameter only are not able to detect all kinds of distributed denial of service attacks.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering, Information filtering*

Keywords

DDoS, Intrusion Detection, KDD, Security

1. INTRODUCTION

Denial of service (DoS) attacks are attacks that have the goal of making a network service unusable for its legitimate users. This can be achieved in different ways, either by targeting specific weaknesses in that service or by brute force approaches. A particularly well-known and dangerous kind of DoS attack are distributed denial of service attacks. These kinds of attacks are more or less brute force bandwidth DoS attacks carried out by multiple attackers simultaneously.

In general, there are two ways to detect any kind of network attacks: Signature based approaches in which the intrusion detection software compares network input to known attacks and anomaly detection methods. Here, the software is either trained with examples for normal traffic or not previously trained at all. Obviously, the latter method is more variable since normal network traffic does not change as quickly as attack methods. The algorithms used in this field are, essentially, known KDD methods for outlier detection such as clustering algorithms, classification algorithms or novelty detection algorithms on time series. However, in contrast to many other related tasks such as credit card fraud detection, network attack detection is highly time critical since attacks have to be detected in near real time. This makes finding suitable methods especially hard because high precision is necessary, too, in order for an intrusion detection system to not cause more harm than being of help.

The main goal of this research project is to build a distributed denial of service detection system that can be used in today's networks and meets the demands formulated in the previous paragraph. In order to build such a system, many considerations have to be done. Some of these are presented in this work.

The remainder of this paper is structured as follows: In section 2 an introduction to distributed denial of service attacks and known countermeasures is given, section 3 points out known test datasets. In section 4 some already existing approaches are presented and finally section 5 concludes this work and gives insight in future work.

2. INTRODUCTION TO DDOS ATTACKS

Denial of service and distributed denial of service attacks are not a new threat in the internet. In [15] the first notable denial of service attack is dated to 1996 when the internet provider Panix was taken down for a week by a TCP SYN flood attack. The same article dates the first noteworthy distributed denial of service attack to the year 1997 when internet service providers in several countries as well as an IRC network were attacked by a teenager. Since then, many of the more elaborate attacks that worked well in the past, have been successfully defused.

Let us, as an example, examine the TCP SYN flood attack. A TCP connection is established by a three way hand-

25th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 28.05.2013 - 31.05.2013, Ilmenau, Germany.

shake. On getting a SYN request packet, in order to open a TCP connection, the addressed computer has to store some information on the incoming packet and then answers with a SYN ACK packet which is, on regularly opening a TCP connection, again replied by an ACK packet.

The idea of the SYN flood attack is to cause a memory overrun on the victim by sending many TCP SYN packets. As for every such packet the victim has to store information while the attacker just generates new packets and ignores the victim's answers. By this the whole available memory of the victim can be used up, thus disabling the victim to open legitimate connection to regular clients. As a countermeasure, in [7] SYN cookies were introduced. Here, instead of storing the information associated with the only half opened TCP connection in the local memory, that information is coded into the TCP sequence number. Since that number is returned by regular clients on sending the last packet of the already described three way handshake and initial sequence numbers can be arbitrarily chosen by each connection partner, no changes on the TCP implementation of the client side have to be made. Essentially, this reduces the SYN cookie attack to a mere bandwidth based attack.

The same applies to many other attack methods that have been successfully used in the past, such as the smurf attack [1] or the fraggle attack. Both of these attacks are so called reflector attacks that consist of sending an echo packet – ICMP echo in case of the smurf attack and UDP echo in case of the fraggle attack – to a network's broadcast address. The sender's address in this packet has to be forged so that the packet occurs to be sent by the victim of the attack, so that all replies caused by the echo packet are routed to the victim.

Thus, it seems like nowadays most denial of service attacks are distributed denial of service attack trying to exhaust the victims bandwidth. Examples for this are the attacks on Estonian government and business computers in 2007 [12].

As already mentioned, distributed denial of service attacks are denial of service attacks with several participating attackers. The number of participating computers can differ largely, ranging from just a few machines to several thousands. Also, in most cases, the owners of these computers are not aware that they are part of an attack. This lies in the nature of most DDoS attacks which consist of three steps:

1. Building or reusing a malware that is able to receive commands from the main attacker ("master") and to carry out the attack. A popular DDoS framework is Stacheldraht [9].

2. Distribute the software created in step one to create a botnet. This step can essentially be carried out in every known method of distributing malware, for example by forged mail attachments or by adding it to software like pirate copies.

3. Launch the attack by giving the infected computers the command.

This procedure – from the point of view of the main attacker – has the advantage of not having to maintain a direct connection to the victim. This makes it very hard to track that person. It is notable though, that during attacks originating to Anonymous in the years 2010 and 2012 Low Orbit Ion Cannon [6] was used. This is originally a tool for stress testing that allows users, among other functions, to voluntary join a botnet in order to carry out an attack. Since the tool is mainly for testing purposes, the queries are not masqueraded so that it is easy to identify the participating persons. Again, however, the initiator of the attack does not necessarily have to have direct contact to the victim and thus remains unknown.

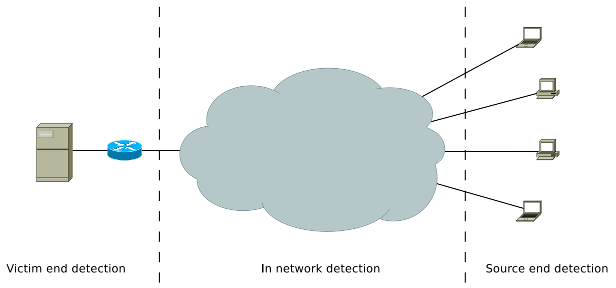

Figure 1: Detection locations for DDoS attacks.

A great diversity of approaches to solve the problem of detecting DDoS attacks exists. Note again, that this work focuses on anomaly detection methods only. This describes methods, that essentially make use of outlier detection methods to distinguish normal traffic and attack traffic. In a field with as many publications as intrusion detection and even, more specialised, DDoS detection, it is not surprising, that many different approaches are used, most of which are common in other knowledge discovery research fields as well.

As can be seen in Figure 1 this research part, again, can be divided in three major categories, namely distributed detection or in network detection, source end detection and end point or victim end detection.

By distributed detection approaches we denote all approaches that use more than one node in order to monitor the network traffic. This kind of solution is mostly aimed to be used by internet providers and sometimes cooperation between more than one or even all ISPs is expected. The main idea of almost all of these systems is to monitor the network traffic inside the backbone network. Monitors are mostly expected to be backbone routers, that communicate the results of their monitoring either to a central instance or among each other. These systems allow an early detection of suspicious network traffic so that an attack can be detected and disabled – by dropping the suspicious network packets – before it reaches the server the attack is aimed at. However, despite these methods being very mighty in theory, they suffer the main disadvantage of not being able to be employed without the help of one or more ISPs. Currently, this makes these approaches impractical for end users since, to the knowledge of the author, at this moment no ISP uses such an approach.

Source end detection describes approaches that monitor outgoing attack streams. Of course, such methods can only be successful if the owner of an attacking computer is not aware of his computer participating in that attack. A widely used deployment of such solutions is necessary for them to have an effect. If this happens, however, these methods have the chance to not only detect distributed denial of service attacks but also to prevent them by stopping the attacking traffic flows. However, in our opinion, the necessity of wide deployment makes a successful usage of this methods – at

Packet type	No of packets	Percentage
IP	65612516	100
TCP	65295894	99.5174
UDP	77	0.0001
ICMP	316545	0.4824

Protocol	Incoming Traffic	Outgoing Traffic
IP	24363685	41248831
TCP	24204679	41091215
UDP	77	0
ICMP	158929	157616

Table 1: Distribution of web traffic on protocol types and incoming and outgoing traffic at the university's web server.

least in the near future – difficult.

In contrast to the approaches described above, end point detection describes those methods that rely on one host only. In general, this host can be either the same server other applications are running on or a dedicated firewall in the case of small networks. Clearly, these approaches suffer one disadvantage: Attacks cannot be detected before the attack packets arrive at their destination, as only those packets can be inspected. On the other hand end point based methods allow individual deployment and can therefore be used nowadays. Due to this fact, our work focuses on end point approaches.

3. TEST TRACES OF DISTRIBUTED DENIAL OF SERVICE ATTACKS

Today, the testing of DDoS detection methods unfortunately is not easy, as not many recordings of real or simulated DDoS attacks exist or, at least, are not publicly available. The best known test trace is the KDD Cup 99 data set [3]. A detailed description of this data set is given in [18]. Other known datasets are the 1998 DARPA intrusion detection evaluation data set that has been described in [14] as well as the 1999 DARPA intrusion detection evaluation data set examined in [13].

In terms of the internet, with an age of 14 to 15 years, these data sets are rather old and therefore cannot reflect today's traffic volume and behaviour in a realistic fashion. Since testing with real distributed denial of service attacks is rather difficult both on technical as well as legal level, we suggest the usage of a DDoS simulator. In order to get a feeling for today's web traffic, we recorded a trace at the main web server of Heinrich-Heine-Universität. Tracing started on 17th September 2012 at eight o'clock local time and lasted until eight o'clock the next day.

This trace consists of 65612516 packets of IP traffic with 31841 unique communication partners contacting the web server. As can be seen in Table 1 almost all of these packets are TCP traffic. This is not surprising as the HTTP protocol uses the TCP protocol and web page requests are HTTP messages.

About one third of the TCP traffic is incoming traffic. This, too, is no surprise as most clients send small request messages and, in return, get web pages that often include images or other larger data and thus consist of more than one package. It can also be seen, clearly, that all of the

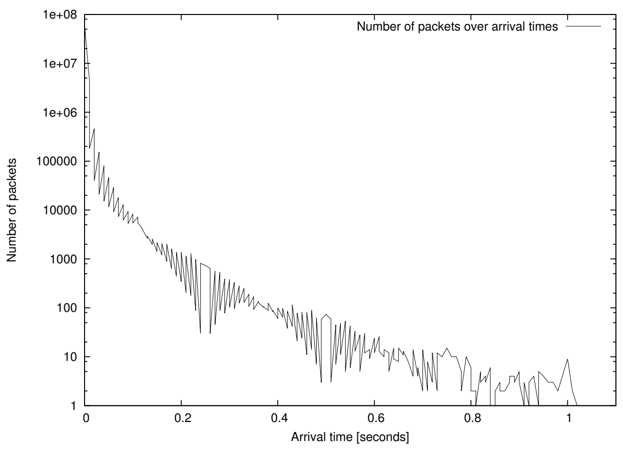

Figure 2: Arrival times for the university's webserver trace.

UDP packets seem to be unwanted packets as none of these packets is replied. The low overall number of these packets is an indicator for this fact, too. With ICMP traffic, incoming and outgoing packet numbers are nearly the same which lies in the nature of this message protocol.

In order to overcome the problems with old traces, based on the characteristics of the web trace, as a next step we implement a simulator for distributed denial of service attacks. As the results in [20] show, the network simulators OMNeT++ [19], ns-3 [10] and JiST [5] are, in terms of speed and memory usage, more or less equal. To not let the simulation become either too slow or too inaccurate, it is intended to simulate a nearer neighbourhood of the victim server very accurately. With greater distance to the victim, it is planned to simulate in less detail. In this context, the distance between two network nodes is given by the number of hops between the nodes.

Simulation results then will be compared with the aforementioned network trace to ensure its realistic behaviour. After the simulation of normal network traffic resembles the real traffic at the victim server close enough, we will proceed by implementing distributed denial of service attacks in the simulator environment. With this simulator it will then, hopefully, be possible to test existing and new distributed denial of service detection approaches in greater detail as has been possible in the past.

4. EXISTING APPROACHES

Many approaches to the detection of distributed denial of service attacks already exist. As has been previously pointed out in section 1, in contrast to many other outlier and novelty detection applications in the KDD field, the detection of DDoS attacks is extremely time critical, hence near real time detection is necessary.

Intuitively, the less parameters are observed by an approach, the faster it should work. Therefore, first, we take a look at a recently issued method that relies on one parameter only.

4.1 Arrival Time Based DDoS Detection

In [17] the authors propose an approach that bases on irregularities in the inter packet arrival times. By this term

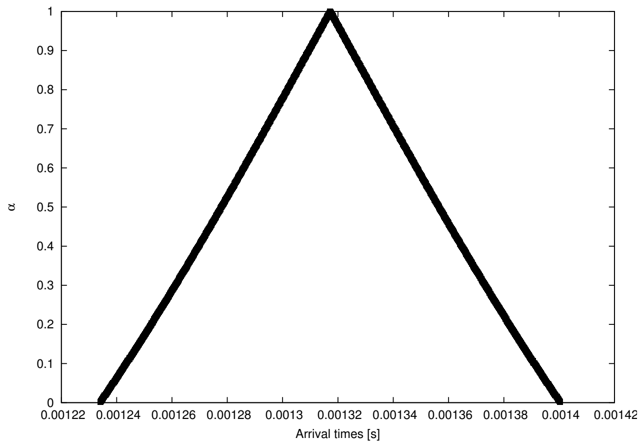

Figure 3: The fuzzy mean estimator constructed according to [17].

the authors describe the time that elapses between two subsequent packets.

The main idea of this work is based on [8] where non-asymptotic fuzzy estimators are used to estimate variable costs. Here, this idea is used to estimate the mean arrival time $\bar{x}$ of normal traffic packets. Then, the mean arrival time of the current traffic – denoted by t_c – is estimated, too, and compared to the overall value. If $t_c > \bar{x}$, the traffic is considered as normal traffic and if $t_c < \bar{x}$ a DDoS attack is assumed to be happening. We suppose here, that for a value of $t_c = \bar{x}$ no attack is happening, although this case is not considered in the original paper.

To get a general feeling for the arrival times, we computed them for our trace. The result is shown in Figure 2. Note, that the y-axis is scaled logarithmic as values for arrival times larger than 0.1 seconds could not been distinguished from zero on a linear y-axis. It can be seen here, that most arrival times are very close to zero. It is also noteworthy that, due to the limited precision of libpcap [2], the most common arrival interval is zero.

Computing the fuzzy mean estimator for packet arrival times yields the graph presented in Figure 3 and $\bar{x} \approx 0.00132$. Note, that since the choice of parameter $\beta \in [0, 1)$ is not specified in [17], we here chose $\beta = \frac{1}{2}$. We will see, however, that, as far as our understanding of the proposed method goes, this parameter has no further influence.

To compute the α-cuts of a fuzzy number, one has to compute

$$^{\alpha}M = \left[\bar{x} - z_{g(\alpha)}\frac{\sigma}{\sqrt{n}}, \bar{x} + z_{g(\alpha)}\frac{\sigma}{\sqrt{n}}\right]$$

where $\bar{x}$ is the mean value – i.e. exactly the value that is going to be estimated – and σ is presumably the arrival times' deviation. Also

$$g(\alpha) = \left(\frac{1}{2} - \frac{\beta}{2}\right)\alpha + \frac{\beta}{2}$$

and

$$z_{g(\alpha)} = \Phi^{-1}(1 - g(\alpha)).$$

Note, that $^{\alpha}M$ is the $(1-\alpha)(1-\beta)$ confidence interval for μ, the real mean value of packet arrival times.

Now, since we are solely interested in the estimation of $\bar{x}$, only ^{1}M is needed, which is computed to be $[\bar{x}, \bar{x}]$ since

$$g(1) = \left(\frac{1}{2} - \frac{\beta}{2}\right)1 + \frac{\beta}{2} = \frac{1}{2}(1 - \beta + \beta) = \frac{1}{2}$$

and

$$z_{g(1)} = \Phi^{-1}(1 - g(1)) = \Phi^{-1}(\frac{1}{2}) = 0.$$

During traffic monitoring, for a given time interval, the current traffic arrival times t_c are computed by estimating

$$[t_c]_{\alpha} = \left[\ln\left(\frac{1}{1-p}\right)\frac{1}{r_{\alpha}}, \ln\left(\frac{1}{1-p}\right)\frac{1}{l_{\alpha}}\right]$$

where p is some given but again not specified probability and $[l_{\alpha}, r_{\alpha}]$ are the α-cuts for $E(T) = \bar{t}$. As described above, the only value that is of further use is t_c, the only value in the interval of $[t_c]_1$. Since $[E(T)]_1 = [\bar{t}]_1 = [\bar{t}, \bar{t}]$ it follows that

$$[t_c]_1 = \left[\ln\left(\frac{1}{1-p}\right)\frac{1}{\bar{t}}, \ln\left(\frac{1}{1-p}\right)\frac{1}{\bar{t}}\right]$$

and thus

$$t_c = \ln\left(\frac{1}{1-p}\right)\frac{1}{\bar{t}} = \frac{1}{\bar{t}}\left(\ln(1) - \ln(1-p)\right).$$

As $\ln(1) = 0$ this can be further simplified to

$$t_c = -\frac{\ln(1-p)}{\bar{t}} \in [0, \infty)$$

with $p \in [0, 1)$.

By this we are able to determine a value for p by choosing the smallest p where $t_c \geq \bar{x}$ for all intervals in our trace. An interval length of four seconds was chosen to ensure comparability with the results presented in [17].

During the interval with the highest traffic volume 53568 packets arrived resulting in an average arrival time of $\bar{t} \approx 7.4671 \cdot 10^{-5}$ seconds. Note here, that we did not maximise the number of packets for the interval but instead let the first interval begin at the first timestamp in our trace rounded down to full seconds and split the trace sequentially from there on.

Now, in order to compute p one has to set

$$p = 1 - e^{-\bar{x}\bar{t}}$$

leading to $p \approx 9.8359 \cdot 10^{-8}$. As soon as this value of p is learned, the approach is essentially a static comparison.

There are, however, other weaknesses to this approach as well: Since the only monitored value is the arrival time, a statement on values such as bandwidth usage cannot be made. Consider an attack where multiple corrupted computers try to download a large file from a server via a TCP connection. This behaviour will result in relatively large packets being sent from the server to the clients, resulting in larger arrival times as well. Still, the server's connection can be jammed by this traffic thus causing a denial of service.

By this, we draw the conclusion that a method relying on only one parameter – in this example arrival times – cannot detect all kinds of DDoS attacks. Thus, despite its low processing requirements, such an approach in our opinion is not suited for general DDoS detection even if it seems that it can detect packet flooding attacks with high precision as stated in the paper.

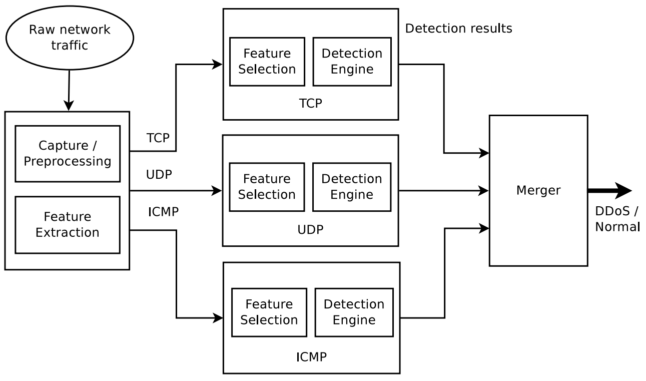

Figure 4: Protocol specific DDoS detection architecture as proposed in [11].

4.2 Protocol Type Specific DDoS Detection

In [11] another approach is presented: Instead of using the same methods on all types of packets, different procedures are used for different protocol types. This is due to the fact, that different protocols show different behaviour. Especially TCP traffic behaviour differs from UDP and ICMP traffic because of its flow control features. By this the authors try to minimise the feature set characterising distributed denial of service attacks for every protocol type, separately, such that computation time is minimised, too.

The proposed detection scheme is described as a four step approach, as shown in Figure 4. Here, the first step is the preprocessing where all features of the raw network traffic are extracted. Then packets are forwarded to the correspondent modules based on the packet's protocol type.

The next step is the protocol specific feature selection. Here, per protocol type, the most significant features are selected. This is done by using the linear correlation based feature selection (LCFS) algorithm that has been introduced in [4], which essentially ranks the given features by their correlation coefficients given by

$$corr(X,Y) := \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{x})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

for two random variables X, Y with values $x_i, y_i, 1 \leq i \leq n$, respectively. A pseudo code version of LCFS is given in Algorithm 1. As can be seen there, the number of features in the reduced set must be given by the user. This number characterises the trade-off between precision of the detection and detection speed.

The third step is the classification of the instances in either normal traffic or DDoS traffic. The classification is trained on the reduced feature set generated in the previous step. The authors tested different well known classification techniques and established C4.5 [16] as the method working best in this case.

Finally, the outputs of the classifiers are given to the merger to be able to report warnings over one alarm generation interface instead of three. The authors mention that there is a check for false positives in the merger, too. However, there is no further information given on how this check works apart from the fact that it is relatively slow.

The presented experiments have been carried out on the aforementioned KDD Cup data set as well as on two self-made data sets for which the authors attacked a server within the university's campus. The presented results show that on all data sets the DDoS detection accuracy varies in the range of 99.683% to 99,986% if all of the traffic's attributes are used. When reduced to three or five attributes, accuracy stays high with DDoS detection of 99.481% to 99.972%. At the same time, the computation time shrinks by a factor of two leading to a per instance computation time of 0.0116ms (three attributes) on the KDD Cup data set and 0.0108ms (three attributes) and 0.0163ms (five attributes) on the self-recorded data sets of the authors.

Algorithm 1 LCFS algorithm based on [11].

```
Require: the initial set of all features $I$,
    the class-outputs $y$,
    the desired number of features $n$
Ensure: the dimension reduced subset $F \subset I$

 1: for all $f_i \in I$ do
 2:     compute $corr(f_i, y)$
 3: end for
 4: $f := \max\{correlation(f_i, y) | f_i \in I\}$
 5: $F := \{f\}$
 6: $I := I \setminus \{f\}$
 7: while $|F| < n$ do
 8:     $f := \max\left\{ corr(f_i, y) - \frac{1}{|F|}\sum_{f_j \in F} corr(f_i, f_j) \middle| f_i \in I \right\}$
 9:     $F := F \cup \{f\}$
10:     $I := I \setminus \{f\}$
11: end while
12: return $F$
```

Taking into account the 53568 packets in a four second interval we recorded, the computation time during this interval would be about ($53568 \cdot 0.0163\text{ms} \approx$) 0.87 seconds. However, there is no information about the machine that carried out the computations given in the paper such that this number appears to be rather meaningless. If we suppose a fast machine with no additional tasks, this computation time would be relatively high.

Nevertheless, the results presented in the paper at hand are promising enough to consider a future re-evaluation on a known machine with our recorded trace and simulated DDoS attacks.

5. CONCLUSION

We have seen that distributed denial of service attacks are, in comparison to the age of the internet itself, a relatively old threat. Against many of the more sophisticated attacks specialised counter measures exist, such as TCP SYN cookies in order to prevent the dangers of SYN flooding. Thus, most DDoS attacks nowadays are pure bandwidth or brute force attacks and attack detection should focus on this types of attacks, making outlier detection techniques the method of choice. Still, since many DDoS toolkits such as Stacheldraht allow for attacks like SYN flooding properties of this attacks can still indicate an ongoing attack.

Also, albeit much research on the field of DDoS detection has been done during the last two decades that lead to a nearly equally large number of possible solutions, in section 3 we have seen that one of the biggest problems is the unavailability of recent test traces or a simulator being able to produce such traces. With the best known test series having an age of fourteen years, today, the results presented in many of the research papers on this topic are difficult to compare and confirm.

Even if one can rate the suitability of certain approaches in respect to detect certain approaches, such as seen in section 4, a definite judgement of given methods is not easy. We therefore, before starting to implement an own approach to distributed denial of service detection, want to overcome this problem by implementing a DDoS simulator.

With the help of this tool, we will be subsequently able to compare existing approaches among each other and to our ideas in a fashion reproducible for others.

6. REFERENCES

[1] CERT CC. Smurf Attack. *http://www.cert.org/advisories/CA-1998-01.html.*

[2] The Homepage of Tcpdump and Libpcap. *http://www.tcpdump.org/.*

[3] KDD Cup Dataset. *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*, 1999.

[4] F. Amiri, M. Rezaei Yousefi, C. Lucas, A. Shakery, and N. Yazdani. Mutual Information-based Feature Selection for Intrusion Detection Systems. *Journal of Network and Computer Applications*, 34(4):1184–1199, 2011.

[5] R. Barr, Z. J. Haas, and R. van Renesse. JiST: An Efficient Approach to Simulation Using Virtual Machines. *Software: Practice and Experience*, 35(6):539–576, 2005.

[6] A. M. Batishchev. Low Orbit Ion Cannon. *http://sourceforge.net/projects/loic/.*

[7] D. Bernstein and E. Schenk. TCP SYN Cookies. *on-line journal, http://cr.yp.to/syncookies.html*, 1996.

[8] K. A. Chrysafis and B. K. Papadopoulos. Cost–volume–profit Analysis Under Uncertainty: A Model with Fuzzy Estimators Based on Confidence Intervals. *International Journal of Production Research*, 47(21):5977–5999, 2009.

[9] D. Dittrich. The 'Stacheldraht' Distributed Denial of Service Attack Tool. *http://staff.washington.edu/dittrich/misc/stacheldraht.analysis*, 1999.

[10] T. Henderson. ns-3 Overview. *http://www.nsnam.org/docs/ns-3-overview.pdf*, May 2011.

[11] H. J. Kashyap and D. Bhattacharyya. A DDoS Attack Detection Mechanism Based on Protocol Specific Traffic Features. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, pages 194–200. ACM, 2012.

[12] M. Lesk. The New Front Line: Estonia under Cyberassault. *Security & Privacy, IEEE*, 5(4):76–79, 2007.

[13] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 DARPA Off-line Intrusion Detection Evaluation. *Computer networks*, 34(4):579–595, 2000.

[14] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, et al. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 12–26. IEEE, 2000.

[15] G. Loukas and G. Öke. Protection Against Denial of Service Attacks: A Survey. *The Computer Journal*, 53(7):1020–1037, 2010.

[16] J. R. Quinlan. *C4.5: Programs for Machine Learning*, volume 1. Morgan Kaufmann, 1993.

[17] S. N. Shiaeles, V. Katos, A. S. Karakos, and B. K. Papadopoulos. Real Time DDoS Detection Using Fuzzy Estimators. *Computers & Security*, 31(6):782–790, 2012.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani. A Detailed Analysis of the KDD CUP 99 Data Set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.

[19] A. Varga and R. Hornig. An Overview of the OMNeT++ Simulation Environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[20] E. Weingartner, H. vom Lehn, and K. Wehrle. A Performance Comparison of Recent Network Simulators. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–5, 2009.