International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Implementation and Analysis of DoS Attack Detection Algorithms

Rupesh Jaiswal[1], Dr. Shashikant Lokhande[2], Aditya Gulavani[3]

[1]Assistant Professor, Dept. of E&TC, Pune Institute of Computer Technology, Pune, India

[2]Prof., Dept. of E&TC, Sinhgad College of Engineering, Pune, India

[3]Student, Dept. of E&TC, Pune Institute of Computer Technology, Pune, India

Abstract: ***Intrusion detection systems have been traditionally classified in three categories viz. Signature Based IDS, Anomaly Based IDS and Hybrid IDS. Each one of these have their own advantages and disadvantages. The anomaly based IDS can detect novel attacks without knowing the actual payload contents if tuned correctly. Anomaly based IDS depends on the rate of data packets at the interface. But the main drawback of using anomaly based IDS is it can produce large number of false positives. The signature based IDS while not producing false positives cannot detect new attacks until its database is updated. The hybrid IDS combines features of both the anomaly based and signature based IDS. In this paper, we discuss the implementation of the each type the IDS. And also we measure the performance of the IDS based on RAM utilization and shows that out detection algorithm consumes less RAM compared to SNORT. Other parameters of analysis are left for future research work.***

Keywords: IDS, DDoS, Attack, Anomaly.

1. Introduction and Related work

Denials of service attacks are network attacks that intend to block access to legitimate traffic [1]. This is usually done by flooding the server by sending huge number of illicit requests. So when a user tries to access the server, he might not get the service as the server is flooded with packets from hostile elements. If computers are hacked and then if they are used to send attack traffic to the server, the type of attack is called as DDoS (Distributed Denial of Service) attack. These kinds of attacks are relatively new.

The ubiquity in use of internet has made it mandatory to for network administrators to keep a tab on hostile elements who pose a direct threat to the function of the network as a whole. With a lot of products already in market to counter the DoS attack, we in this paper try to do a comparative study of the basic algorithm used in the three types of IDS [2] namely - Anomaly based [5], Signature based [6-8] and Hybrid IDS [3].

Ids can be host based (HIDS) or network based (NIDS). HIDS uses the data from host operating system as the main source of input to detect malicious activity [9-10], whereas NIDS uses its attack detection mechanism based on network traffic monitoring [11-12].

The algorithm used for detecting Intrusion should be fool proof, for the simple reason that it is the first step towards preventing attacks on the network i.e. an IDPS. The basic function of IDS is to scan the packets at the network interface of its host and determine whether or not the incoming traffic is an attack or not. It is for these decisions making the different algorithms are employed. An attack can be characterized in various ways - it may be the content or illegal combination of flags or the statistical data available from various packets.

The algorithm packet header anomaly detection [13] uses the statistical data extracted from the time stamps of two consecutive similar packets and uses mathematical modelling of the same to determine the attack. The signature based approach uses the actual data contents in the payload or specific header combinations to match with an existing known attack signature. The Hybrid Algorithm [14] uses both -the statistical calculation and the signature matching to detect an attack.

The rest of this paper is organized as follows. We discuss each of the aforementioned algorithms section wise. Section II describes Signature Based IDS and its implementation; Section III discusses Anomaly Based IDS and its implementation; Section IV describes Hybrid IDS. Section V ends with the result analysis and discussion.

2. Signature Based IDS

As mentioned earlier, a signature based IDS [4] uses the data content in the packet to match with an existing known signature. If a match is found, we know that an attack is going on. And since each attack is going to have its own signature, we can also detect the type of attack. Now, signature of an attack may imply the data content, invalid flag combinations (in case of TCP), specific port numbers or in some cases invalid fields of the IP packet. We will first see the implementation of the IDS. The basic algorithm for implementation of any IDS includes capturing the packets and then applying the detection technique. In case of signature based IDS, we first capture packets and then implement the string matching algorithm.

2.1 Implementing Packet Capture at Interface

Capturing packets from an interface can be done in various ways viz. using winpcap (for Windows OS), pcap (for Linux

OS) or using libpcap and writing a program to capture the packets. Linux OS provides pcap.h under the libpcap library. The detailed documentation of pcap.h can be found in reference [insert number here]. pcap.h provides various functions to access, capture on, handle and filter on a specified interface. Some useful functions are described here. pcap_lookupdev() returns a reference to a string containing the name of a network device which can be used with pcap_open_live() and with pcap_lookupnet(). pcap_open_live() is used to get a packet capture handle to capture and analyze packets on the network. pcap_compile() is used to compile the string str into a filter program. pcap_loop() processes packets from a live capture until count packets are processed. Using these functions, we can access and open an interface to capture packets. Now, we save the captured packets in a dump file. This dump file will be accessed again during the string matching stage of the algorithm.

2.2 Implementing String Matching Algorithm

There are a number of string matching algorithms for searching a substring in a string. For a time constrained application like this, we need an algorithm that can perform string matching with the least possible time complexity. The Boyer-Moore algorithm is the most widely used string matching algorithm. It is considered to be the fastest algorithm for string matching. Horspool's algorithm (also referred to as Boyer-Moore-Horspool algorithm) is a reduced version of the Boyer-Moore algorithm that reduces the space complexity in trade of time complexity. We will see the implementation of this algorithm in detail.

This algorithm carries out the matching from right to left. This algorithm uses a bad match table. This bad match table tells how many characters are to be shifted or skipped for the next iteration. In naive algorithm, we shift by just one character. It has been shown that we can skip a certain number of characters to improve the time complexity of the algorithm. Now, the bad match table has in its simplest form, two columns. First column has the character from the substring and the second column has the respective shifts. Whenever a bad match occurs, the bad match table is consulted and corresponding characters are skipped. The bad match table is constructed by using the following algorithm:

```
for(i = 0; i < substring.length; i++)
 bmtable[substring[i]] = substring.length – i – 1;
```

Bad match table for the word "TRUTH" is shown in table1.

Table 1: Bad match table for the word "TRUTH"

Index	Shift
T	1
R	3
U	2
?	5

The basic idea of using the bad match table is, whenever a bad match occurs at a character not present in the substring, we will not have to check all the characters in the remaining substring. So, we can shift straight away by length of the substring. An example is shown in Table 2.

2.3 Signature Database

We need to have a database of all the attack signatures to match them with the packets captures. We managed to procure a handful of signatures from snort IDS [8]. These signatures are used as the primary signatures for attack detection. Also, we have gathered a few tools for simulating DoS attacks. Using these attack tools and analyzing the packets, we also added some more signatures of our own.

Table 2: String Matching for word "TRUTH"

```
String:  THIS  IS THE TRUTH ANALYSIS OF PAPER
Key:     TRUTH
```

Mismatch occurs at " "; since " " is not present in the key, we shift by 5 characters.

```
String:  THIS IS  THE TRUTH ANALYSIS OF PAPER
Key:          TRUTH
```

Mismatch occurs at " "; Again we shift by 5 characters.

```
String:  THIS IS THE TRUTH ANALYSIS OF PAPER
Key:               TRUTH
```

Mismatch occurs at "U"; since "U" is present in the key, we shift by 2 characters (from bad match table).

```
String:  THIS IS THE TRUTH ANALYSIS OF PAPER
Key:               TRUTH
```

And now, we have the match.

2.4 Implementation of IDS Algorithm

Now that we have seen packet capture, string matching algorithm and signature database we are ready to implement our signature based intrusion detection system. First, we will see the basic algorithm and the flow of operations and then we will see the method of implementation.

Note that the packet capture program and string matching program have to be separate programs to avoid dropping packets. Initially, the packet capture program is running. After a certain packet are captured (number of packets set by the user), the packet capture program invokes the string matching program for the first time. At this point, the packet capture program changes the dump file so that string matching can be carried out on the previous file.

Again after packet capture, the program changes dump file and sets a mutex variable. The string matching program which was invoked previously, now waits for the mutex to be set. Once set, it checks to attacks. This process continues. The number of dump files to be created depends on the traffic rate and can be set by the user.

To invoke other programs, Linux provides two important functions – fork() and exec(). Each process has its own process ID, referred to as PID here onwards. fork() is a system call that makes a copy of the current process in execution in a different address space. So, now there are two exactly identical processes running in parallel. The only difference between these two processes is their PID. The parent process in which fork() was called retains its PID and the new born process called the child gets 0 as its PID. The exec() function replaces the process in which it is called with the process whose path is provided to it as an argument.

Since we had two similar processes, we can replace the child with the string matching program. Now, our programs, packet capture and string matching are running in parallel. This is the complete implementation of the signature based IDS.

3. Anomaly Based IDS

In the anomaly based detection the IDS is tuned to normal traffic conditions. It does so by keeping a database of various fields of the normal traffic. When the IDS is made to function in a real time environment, it checks for anomalies in the incoming traffic with respect to the database it has created while learning.

3.1 Choice of Fields for Detection

The choice of the fields for detecting attacks is at the discernment of the designer. The most commonly used fields are those of the IP address and port numbers. The use of increased number of fields will obviously reduce the number of false positives but in turn also increase the memory and training overheads required for the IDS to function. For this paper we take into consideration some fields like Type of Service, Source port, IP address, Destination Port, Header Length etc. There may be many fields that do not generate anomaly even in the event of an attack which means that the attack packets and normal traffic has same characteristics when compared with respect to those fields.

To improve on the results obtained we can keep a threshold level for the rate of attack packets. This allows us to reduce the number of false positives. But this on the other hand will require more precise tuning in its training mode and addition of another parameter with every field in its database. One of the common parameters used is the time stamp between two consecutive packets of the same type. This can be used to calculate the rate of incoming packets with from a specific source or of the same type.

3.2 Training Mode

The training mode includes training the IDS under an attack free environment for a sufficiently long time. The training mode makes the IDS aware of what is considered to be normal. Values and thresholds of parameters for various fields are calculated during this period. Many a times an anomaly occurs due to certain changes in the hardware of the network. Such kinds of anomalies are non-recurring in nature.

We use 20 different fields for detection of anomalies in the data packet at the interface. Some of the values of the common fields are listed in the table for reference. The more exhaustive the training will be the less the number of false outputs during the detection mode. The use of machine learning to compute a mathematical model will increase the speed at run time but will have higher training and pre-processing overheads for the same.

The system training should be done on various types of traffic like FTP, HTTP etc. This would be helpful in reducing the number of false positives and give a more faithful result when run on the host network because when the system is installed on the network it will counter all sorts of traffic and the attack may be in any form.

3.3 Detection Mode

The functioning mode of the IDS is known as the detection mode. It captures data packets at the interface and computes the statistical data. It then cross-verifies the obtained data of each field during run time with acceptable value or thresholds of parameters. When an attack occurs it gives values other than the trained values and thus an attack is detected. The event may be classified as an attack due to anomaly in any of the fields. For example the rate of a specific packet from the same source may be higher than the acceptable value or the number of packets from a specific source in a time frame may be higher than the trained set.

3.4 Implementation of IDS Algorithm

We capture each data packet at a given interface by using the functions defined in the pcap.h like pcap_loop(), pcap_open_live() and pcap_compile(). We need to define structures according to different headers in the protocol stack. We can thus extract the headers from data packets because the headers are to be of known length for each protocol. The protocol can be known by reading the IP. A sample packet header for IGMP header in C programming is shown:

```
struct igmphdr
 {
 u_char type;
 u_char code;
 u_short checksum;
 u_int identifier;
 u_int multi_addr;
 u_int access1;
 u_int access2;
};
```

We then use data types according to the length of field. For example unsigned char for 8bit, 32 for unsigned int and then type cast in C to view the same in the required format.

When running the data packets are captured over a time frame and the statistics are computed. The statistics at runtime are then compared with the normal traffic conditions and thresholds defined during the training and if they are found to be higher than the attack is said to be underway and the network administrator is alerted for the same.

4. Hybrid IDS

Anomaly and Signature based detection system are combined to form hybrid detection system, where the advantages of both are taken into use. The anomaly based algorithm is used to detect novel attacks which are not possible by the signature based approach.

The hybrid detection algorithm also has the scope of incorporating machine learning/ self-learning during run time because feedback from detection of new attacks can be taken and used to define the rule sets for detection using signature based algorithm. This increases the utility of the same and is generally the preferred algorithm over the earlier two algorithms for the sheer improvement of performance over the others.

Even though the Hybrid based Intrusion Detection theoretically may be a stronger system but they may not always be the better suited for all sorts of scenarios. Hybrid based IDS may have the issues of integration because it has the need of various technologies to interoperate successfully and efficiently.

5. Result Analysis and Discussion

Signature Based IDS has some glaringly evident shortcomings in the form of its inability to detect new attacks. Also little deviation in the attack packet can also lead to it not detecting the minor tweaked attacks. So it has to be updated with the latest database every now and then to keep up to the emerging attacks. However the signature based gives very accurate results in the form of almost no false positives. Also it is easier to design and does not need training like in the case of anomaly based detection.

Anomaly based IDS has the drawback of giving vague results when it comes to detecting attacks. It gives pretty much number of false positives which may lead to dropping of legitimate traffic packets. It also needs to be trained at the node of the network where it is to be used. Higher level of Manual intervention is not needed like in the case of Signature based detection because updates aren't generally required in this case. And one of the biggest pros of the anomaly based system is the detection of new attacks, which is not possible in any other system.
Hybrid based detection has the advantages of anomaly in detecting new attacks but can fall short on the ease of designing such a system because for the interoperability of discrete systems and to integrate it to one is a pretty difficult task and also the resources needed for the same will be pretty higher as compared to the other two.

5.1 Metrics

Any system's performance has to be measured with certain metrics to have a thorough comparison and the efficiency of a system will thus be known with respect to others. Two of very important metrics for performance analysis are the resources of the host needed and the number of false positives or negatives.

5.2 Resources Needed

This includes all sorts of resources needed by the system to run successfully on the host machine. This may be the minimum free RAM, number of files written or sent over the network, specific library headers, ROM space etc. Here are some graphs showing the RAM usage of different algorithm plotted in linux with respect to time. Snort, an established and widely used IDS for UNIX and Windows takes about average of 352 MB of RAM while signature based algorithm designed takes lowest resources with the maximum resources utilized on the y axis peaking when both the string matching and packet capturing are running simultaneously. However the total space on ROM required in much higher than the anomaly based detection technique because of the database of the known signatures and the rule set has to be stored in a file on the host.

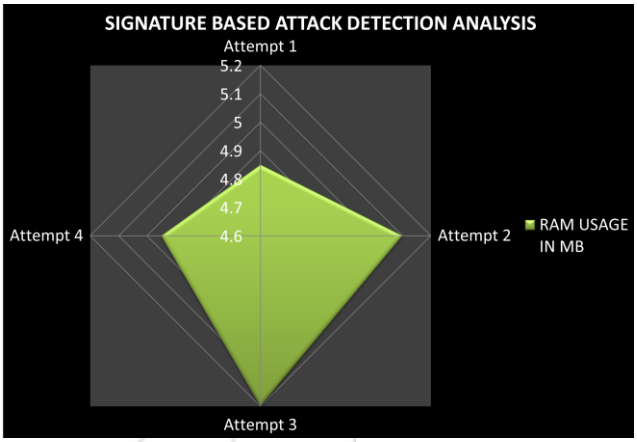

Figure 1: Average Memory usage in Signature based IDS during simultaneous packet capture and pattern matching.

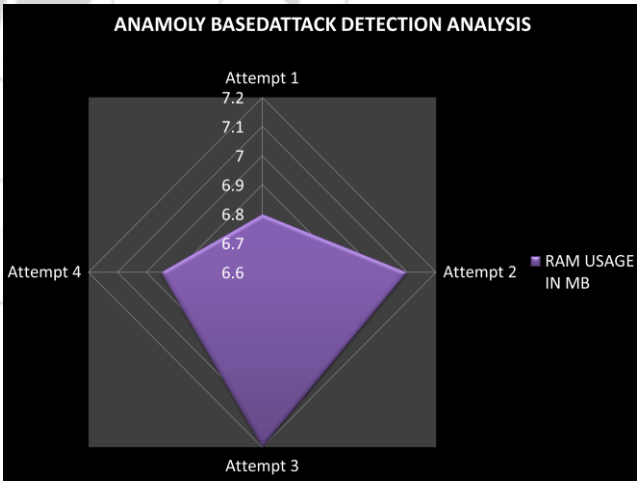

Figure 2: Average Memory usage in Anomaly IDS during training process for two parameter calculation

The Anomaly based detection algorithm requires higher resources due to computation of statistics of the various parameters pertaining to attack detection. The higher the number of parameters calculated for the same the higher will be the RAM required. The executable code section of the anomaly based detection is much higher. Snort on the other hand is a comprehensive IDS which encapsulates many types of attacks and illegitimate traffic, thus requiring much higher RAM than the designed IDS.

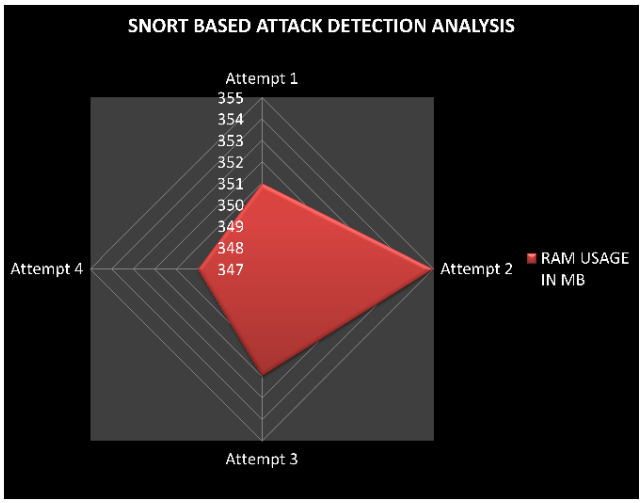

Figure 3: Average Memory usage by Snort IDS

5.3 False Positives and Negatives

False positive is an event where in an attack is detected even in the case of the incoming traffic not being hostile. This degrades the performance of the IDS because it sends control signals to the IDPS unit to drop such packets. A False Negative on the other is the event of the system not detecting an attack even if the attack is underway. This is a more dangerous situation and poses a great threat to the security of the host. The performance can be evaluated with well-known parameters like TP, TN, FP, FN, Precision, Recall, and Accuracy.

Precision (P)= TP / (TP+FP),
Recall (R) = TP / (TP+FN),
Accuracy (A) = (TP+TN) / (TP+TN+FP+FN).

A Signature based approach has no false positives because it gives pretty accurate results when it comes to known attacks. On the other hand in the event of new attacks, the false negative count increases greatly unless the database is updated. Unless there is machine learning or feedback of some sort involved the Signature based approach does not do justice to Intrusion detection in a real time scenario.

The anomaly based detection system has a relatively higher number of false positives because generally the threshold levels are set in a conservative manner to negate the threat posed in false negatives. The number of false positives in an anomaly based system is dependent on various factors like the parameters used, thresholds set during the training period etc.

6. Conclusion and Future Work

With the increase in use of internet in the day to day world, data security and threat management becomes an important issue. The basic requirement for any threat management unit is the detection block. The implementation and design of various algorithms becomes an important issue in the detection.

Two of the three algorithms were implemented and tested on real time emulated attacks. The memory resource is used as metrics to determine the performance and efficiency of the system. Average RAM utilization while attack detection is 5.025 MB for Signature based 7.0125 MB for Anomaly and 352 MB for SNORT IDS. Other performance parameters are left for future work. Also probabilistic approach can be used for intrusion detection in future which may surely consume least resources compared to other approaches. A trade off was seen to be established between the resources required and the efficiency of the system. Each algorithm has its own pros and cons, each being suitable for different types of environment and requirements.

References

[1] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," ACM SIGCOMM Computer Communication Review, vol. 34, pp. 39-53, 2004.
[2] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," 2002, pp. 255-264.
[3] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," Computer Networks, vol. 31, pp. 805- 822, 1999.
[4] F. Dressler, G. Munz, and G. Carle, "Attack detection using cooperating autonomous detection systems (CATS)," Wilhelm- Schickard Institute of Computer Science, Computer Networks and Internet, 2004.
[5] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," computers & security, vol. 28, pp. 18-28, 2009.
[6] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor. In Research in Security and Privacy", 1990. Proceedings. 1990 IEEE Computer Society Symposium on, 1990.
[7] V. Paxson. Bro: a system for detecting network intruders in real-time. Computer networks, 1999.
[8] M. Roesch, "Snort-lightweight intrusion detection for networks", In Proceedings of the 13th USENIX conference on System administration, 1999.
[9] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.- L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, "Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype", In Proceedings of the 14th National Computer Security Conference, 1991.
[10] S. Kumar and E. H. Spa_ord, "A pattern matching model for misuse intrusion detection", IEEE proceedings, 1994.
[11] K. Wang, G. Cretu, and S. Stolfo, "Anomalous payload-based worm detection and signature generation", In Recent Advances in Intrusion Detection, 2004.
[12] V. Paxson, "Bro: a system for detecting network intruders in real-time", Computer networks, 1999.
[13] M. Mahoney and P. Chan. Learning non stationary models of normal net- work track for detecting novel attacks. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002.
[14] Yu-Xin Ding, Min Xiao, Ai-Wu Liu, "Research And Implementation On-Snort-Based Hybrid Intrusion Detection System" Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12-15July IEEE, 2009.