**International Journal of Science and Research (IJSR)**
**ISSN (Online): 2319-7064**
**Impact Factor (2012): 3.358**

# An Efficient Approach in Protection of Information Security via Collaborative Inference Detection

**Abilesh. P**[1]

[1]Sri Ramalinga Sowdambigai College of Science and Commerce, Department of Computer Science,
Affiliated to Bharathiar University, Vadavalli – Thondamuthur Road, Onappalayam, Tamilnadu, India

**Abstract:** ***This paper examines a wide variety of DoS and scanning attacks and shows that several categories (bandwidth based, claim-and-hold, port-scanning) can be scalably detected. In addition to existing approaches for scalable attack detection, the proposed innovative data structure called Fuzzy partial completion filter (FPCF) that can detect claim-and-hold attacks scalable in the network with the introduction of membership and non membership degree of attack prediction. This paper analyzes FPCF both analytically and using experiments on real network traces to demonstrate FPCF can be tuned to achieve extremely low false positive and false negative probabilities. The experimental result shows that the proposed method performs better than the existing partial completing filter.***

**Keywords:** fuzzy partial completion filter, information security via collaborative inference detection, efficient approach for information security, information security using partial completion filters

## 1. Introduction

Intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. Intrusion detection does not, in general, include prevention of intrusions. Intrusion detection can be performed manually or automatically. Manual intrusion detection might take place by examining log files or other evidence for signs of intrusions, including network traffic. A system that performs automated intrusion detection is called an Intrusion Detection System (IDS).

In the last three years, the networking revolution has finally come of age. More than ever before, we see that the Internet is changing computing as we know it. The possibilities and opportunities are limitless; unfortunately, so too are the risks and chances of malicious intrusions.

### 1.1 Detection of DoS Attack

Initially, network administrators will first detect symptoms such as uniform degradation of network or device performance. Uniformly degraded performance could be due to resource consumption of bandwidth attack. Point-to-point attack can also occur to specific devices in the network, causing the CPU utilization to run up and failure of the host to serve other users. Investigating Denial of Service Attacks often require the use of sniffers or logging at the router to determine the extent of the attack, whether it is propagating to other hosts in the network, and to identify the pattern or signature of the attack. Analyzing router and host logs may or may not show the real nature of the attack or may cause false reporting. In some experience with organizations installing commercial network Intruder Detection System, mis-configured attack signature, provided wrong alert indicators. A sniffer at this point helps to identify the real threat. Based on experience, mis-configuration of devices such as hubs and routers can also cause DoS effect. Thus, it is advisable not to eliminate any possibility until the packets are thoroughly examined.

DoS attacks are often double edged sword, the source host (or spoofed host) will be affected just as much as the target host. Due to this situation, an attacker will have to have means to monitor if the attack is successful, by planting a sniffer in the spoofed network or the target network. This situation is proven in incidents involving smurf attacks and syn flood attacks since these connection requests create a massive spur of return packets to the source IP, and often causing a similar impact to the source and the destination IP. Using spoofed IP, the spoofed machine will be swamp with return packets instead. Spoofed source IP makes the attack very difficult to be traced to the originator machine. However, it is also very difficult to spoof IPs, especially when the attacker is within a network with Ingress filters at the routers.

In my experience in handling Incident Response, there were a few incidents involving both parties experiencing DoS attack reporting to us, claiming the attack was initiated vice-versa, due to the fact that their respective firewalls were logging only one direction of the traffic rather than bi-directional. Further analysis and correlation of the logs revealed that the attack was coming from one of them.

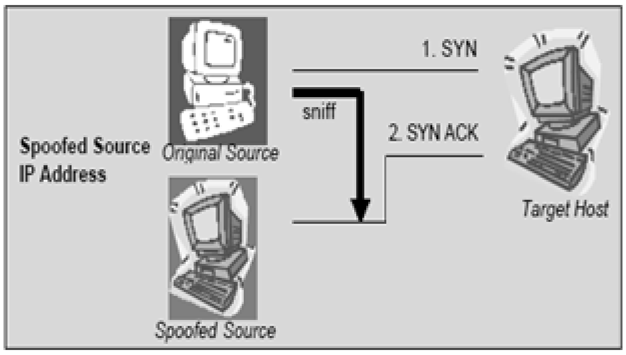

**Figure 1:** Detection of DoS Attack

The NANOG ISPSec Meeting/DDoS BoF described the initial intrusions in which hosts are compromised using known exploits and later rootkit to take full control of the host, before the agents are planted on the hosts. Networks with close proximity to high-volume backbones, large population of vulnerable hosts and weak system administration make good agent sites.

Coordination and cooperation between network providers are crucial for diagnosis, tracing, and control of distributed attacks.

### 1.2 An Introduction to DoS Countermeasures

A comprehensive countermeasure for DoS attacks has four distinct elements: prevention, detection, mitigation, and trace back, shown together in Figure 1. Before an attack occurs, there should be existing prevention mechanisms that are capable of eliminating the threat of the attack. When the attack does occur, only a successful and timely detection of the attack will allow the appropriate mitigation mechanism to be deployed. During or following the attack, a method called trace back can be used to determine the source of the attack. In addition, trace back can also help improve future methods for detecting and preventing a DoS attack. The dependence upon all four of these items is crucial for a successful DoS countermeasure.

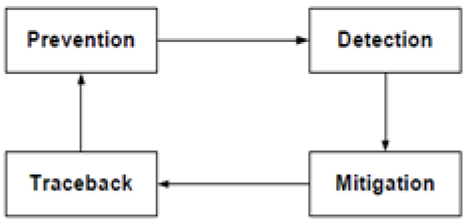

**Figure 2:** Components of a DoS Countermeasure

In addition to these four items, a DoS countermeasure should be designed to defend against attacks on various layers of the Internet stack. Among the five layers of the Internet protocol stack, a network-based DoS attack is associated with three of the five layers: the application, transport, and network layer. In order to design the most effective countermeasure against DoS attacks, we must address the security vulnerabilities at each layer and introduce defense mechanisms that are capable of mitigating specific threats at the appropriate layer.

In computer security, there is no “magical panacea” for all potential threats. The only way to defend a computer from attacks is to design and employ a number of protection mechanisms that are designed to combat a specific threat. The combined use of all of these mechanisms will provide the greatest protection against a wide range of attacks. The most important aspect to the design of a DoS countermeasure is to prevent the countermeasure from becoming the target of a new and unique DoS attack. All DoS attacks target some kind of vulnerability and if the vulnerability exists within the DoS countermeasure, then it needs to be redesigned.

### 1.3 Detection

One of the most important components in designing a DoS countermeasure is to determine and establish the optimal methodology to detect an ongoing attack. Most detection mechanisms rely on some form of an application or software that resides on a host system or within the network that can observe traffic patterns or resource usage. These programs typically are configured to detect anomalies or deviations from normal behavior. When anomalies are detected, alerts are created so that either a system administrator or an automated program can quickly determine the type of the attack and decide which actions to take to safely minimize the effects of the attack and return the system back to its original state.

Most attacks are detected by an end host or server. However, in many cases a DDoS can affect the routers within the network. In the case of a large-scale DDoS flooding attack, detecting the attack should be done throughout the network. To combat a DDoS attack, a distributed defense is essential. In general, the further downstream the detection process is implemented, the easier it will be to determine if there is an ongoing attack. The study of detecting a DoS attack is a very important area for network security research. However, it is beyond the scope of the work for this paper. Despite its importance, we assume that throughout the remainder of this paper that there exists a simple and basic detection mechanism that will activate the appropriate mitigation mechanism.

## 2. Literature Review

Although DoS attacks existed during the 1980s and early 1990s, at the beginning they were not viewed as high-profile security incidents by the general public[1,2]. This perception started to change as the Internet was becoming a mainstream medium. In this section, we present a timeline of the most notable DoS incidents, followed by a brief description of each new type of attack used at each incident.

Before the year 2000 In September 1996, a “SYN Flood" DoS attack took the New York City Internet service provider Panix off-line for a week, while subsequent attacks disabled the web servers of the Internet Chess Club and The New York Times. Two months later, the first commercial product specifically designed for DoS attacks was released. It detected attacks by watching for incoming SYN packets, and responded by resetting the connections if the victim computer received traffic at rate higher than a certain threshold. However, it failed to halt an attack on Webcom's main server which knocked thousands of commercial websites off-line. The attacker had randomized the IP addresses and the attack rate was 200 packets/sec, which was very high at the time[3].

In “SYN Flood" attacks, the attacking system sends SYN messages to the victim server system that appear to be legitimate but in fact reference a client system that is unable to respond to the SYN/ACK messages. This means that the final ACK message will never be sent to the victim server system and due to the many half-open connections, the victim server system becomes eventually unable to accept any new incoming connections.

In January 1997, a teenager attacked the IRC network Undernet and several ISPs in Norway, Romania, the United Kingdom and the United States, with a combination of “ping" and “SYN Flood" attacks [4]. At each stop, he logged onto the server, obtained root access, then deleted files and cancelled accounts. The “ping attack" is one of the simplest DoS attacks, where the victim is flooded with more TCP/ICMP packets than it can handle. In “IRC-based DDoS attacks", an IRC communication channel is used to connect the client to the agents. The attackers can use legitimate IRC ports for sending commands to the agents, which makes their tracking more difficult, because IRC servers tend to receive large volumes of traffic. The attacker no longer needs to maintain a list of agents, since she can simply log on to the IRC server and see a list of all available agents.

The agent software installed in the IRC network usually communicates with the IRC channel and notifies the attacker when the agent is up and running. IRC networks also provide for easy file sharing, which is one of the passive methods of agent code distribution and an easy way for attackers to secure secondary victims to participate in their attacks.

In January 1998, DALnet and other IRC networks became targets of “smurfing", where the attacker is using ICMP echo request packets directed to IP broadcast addresses from remote locations to generate DoS attacks. There are three parties in these attacks: the attacker, the intermediary, and the victim [5,6]. The intermediary receives an ICMP echo request packet directed to the IP broadcast address of their network. If the intermediary does not filter ICMP traffic directed to IP broadcast addresses, many of the machines on the network will receive this ICMP echo request packet and send an ICMP echo reply packet back. When all the machines on a network respond to this ICMP echo request, the result can be severe network congestion and outages. When the attackers create these packets, they do not use the IP address of their own machine as the source address. Instead, they create forged packets that contain the source address of the attacker's intended victim.

In June 2002, the Website of the government of Pakistan was the victim of a politically motivated attack launched by Indian hackers that used “YAHA", a worm with Denial of Service payload. Similarly to “Code Red", “YAHA" caused an infected computer to make repeated connection attempts to the Pakistani government's website and attempted to terminate anti- virus and firewall software [7].

From 2004 up to till date since 2004, DoS incidents have been deliberately not widely publicized, as the scene has shifted to the sensitive field of economic crime and DoS incidents harm the victims' reputation in the eyes of the increasingly security-aware public. Major new trends include Cyber-extortion and bot armies [8].

In January 2006, the million dollar page, a British teenager's novel advertising idea to earn him $1m in 4 months, became very quickly famous around the world [9].

In May 2006, a 20-year old “botmaster" was sentenced to five years in prison for hijacking 500,000 computers. He was selling access to them to other hackers, who used them to launch Dos attacks and send spam emails. More recently, during the August 2008 armed conflict between Russia and Georgia, a series of coordinated DoS attacks of unidentified origin crippled Georgia's Internet infrastructure. Similar DoS attacks had been launched a year earlier against Estonia's Internet infrastructure [18-20].

Although DoS attacks are launched since the beginning of computer networks, they were not considered a significant topic of research until relatively recently, when they started harming ISPs, governmental websites and the e-commerce. The effectiveness of these attacks and their subsequent publicity prompted the influx of newer and even more effective attacking techniques against an increasingly wide range of targets. With DoS techniques becoming distributed and powerful attacking tools being readily available on the Internet, it became quickly apparent that DoS cannot be handled in the same way as other computer security issues.

For example viruses have always been countered with dedicated antivirus software running on the victim computer, but DoS attacks are aiming at overwhelming the target resource altogether, so that the victim cannot employ a defence on its own. The fact that the Internet operates on old networking protocols with limited provision for security is yet another advantage for the attackers. Of course, the increase in research interest did provide solutions, which have recently managed to halt the escalation of the DoS phenomenon. Distributed defence techniques have been designed and the majority of the DoS attacks can now be countered in networks where some sort of defence has been deployed. This can be considered as the end of an era, during which a “script-kiddie" could download a tool and launch an attack against practically any website.

Today, attacks have shifted towards economic crime and cyber-warfare, and although less widespread they can be much more harmful. As a result, the new era of DoS research has to produce even more effective solutions with even less overhead in the absence of an attack and as small disruption in the presence of one.

## 3. Proposed Framework

This paper concentrates on Scalable Intrusion Detection Network systems (SIDS). This is slightly a different approach to intrusion detection. The concept of the this system is simple: it determines "normal" network activity and then all traffic that falls outside the scope of normal is flagged as anomalous (not normal). SIDS systems attempt to learn network traffic patterns on a particular network. This process of traffic analysis continues as long as the SIDS system is active, so, assuming network traffic patterns remain constant, the longer the system is on the network, the more accurate it becomes. By analyzing network traffic and processing the information with complex statistical algorithms, SIDS systems look for anomalies in the established normal network traffic patterns. All packets are given an anomaly score (indicating the degree of irregularity for the specific event) and if the anomaly score is higher than a certain threshold, the IDS will generate an alert.

While the very success of the Internet is due to its open model in which any computer can send to any other computer, this openness also allows attackers to send malicious messages that can cause damage to other hosts and networks, sometimes at great cost. Thus, the field of network security has sprung up in an attempt to prevent or mitigate attacks against campus, enterprise, and ISP networks. The earliest network security solutions attempted to secure Internet hosts using anti-virus software running at end-nodes, and firewalls installed at network vantage points (or, more recently, at hosts themselves). Unfortunately, end-node based approaches must be widely deployed within a network to protect against attacks. They also do very little to mitigate bandwidth attacks that may be blocked at the end-nodes but consume so much internal network bandwidth that the network is unusable. However, we can divide the techniques of intrusion detection into two main types.

### 3.1 Partial Completion Attacks

Such attacks are also known as claim-and-hold attacks. The basic theme in these attacks is to grab a precious resource and not release it thereby denying service to legitimate clients. The classic example of a partial completion attack is syn-flooding [13]. In syn-flooding, the attacker initiates several connections to the server by sending TCP SYN packets with spoofed IP addresses and never terminates any of these connections. In a variant called Naptha, the attacker initiates a connection and finishes the initial three way handshake, but does not do any further activity forcing the connection to time out. In both these attacks, we can see that the precious resource namely, connection memory, is claimed but never released. Port scans are often performed by attackers as preliminary reconnaissance to identify a large number of vulnerable hosts in the Internet. Henceforth, we refer to these myriad activities as just scanning.

### 3.2 Bandwidth Attacks

Finally, the third kind of attacks we discuss in this paper are what are commonly called bandwidth attacks. In such attacks, an attacker or a set of compromised slaves (zombies), continuously pound a victim with a large number of packets, crippling normal services. In other such attacks, the attacker can take advantage of other machines to amplify the magnitude of traffic directed towards a particular destination. Smurf, fraggle, and reflector attacks fall into this category. The common theme in all such attacks is huge traffic volume.

### 3.3 Attacks That Do Scanning

Host scanning represents an important component of several attacks including most worm epidemics2 [14]. Thus, several recent worms such as Code Red-II, Nimda, etc., propagated by scanning other vulnerable hosts in the Internet. A second example is probing for backdoors installed on various machines either installed during worm infection or by other means such as viruses. Such activity also exhibits scanning behavior. Finally, horizontal (multiple hosts and same port) and vertical (one machine, multiple ports).

### 3.4 Problem Statement

A computer system should provide confidentiality, integrity and assurance against denial of service. However, due to increased connectivity (especially on the Internet), and the vast spectrum of financial possibilities that are opening up, more and more systems are subject to attack by intruders. These subversion attempts try to exploit flaws in the operating system as well as in application programs.

## 4. Proposed System

The proposed system concentrates on any scalable intrusion detection mechanism must deal with these two issues. Thus, the contributions of this paper are as follows.

1) **Framework:** Our paper initiates the study of scalable attack detection schemes. We use behavioral aliasing and spoofing as a framework to analyze such techniques.
2) **Technique:** As a specific example, we focus on scalable DDoS and scan detection, and propose a specific new scalable technique called partial completion filters (PCFs).We analyze behavioral aliasing and spoofing characteristics of PCFs in different deployment scenarios.
3) **Evaluation:** To evaluate the efficacy of PCFs, we use a theoretical model later validated by real traces from two different ISPs. For example, in an OC-48 traffic trace for an entire day.

### 4.1 Proposed Framework

1) **Packet Sniffer:** capture the packets when data is transferred to or from any PC in the network.
2) **Network Testbed:** build a network of nodes, where any node can send and receive data with other nodes.
3) **Streaming:** one of the nodes will be acting as a server and all the other nodes will be sending data to the receiving node. The sending nodes will send data at various rates (speeds).
4) **Detection:** the receiving node will calculate the bandwidth from each sending node in order to identify the node that is sending too much data in short time. The packet sniffer will be running in the receiving node. It is performed using Fuzzy Partial Completion filter.
5) **Countermeasure:** the node which is sending too much data in very short time is identified as attacker, and that node is removed from the network, and the attacker cannot communicate with the server from then on. Thus, the attacker is identified and removed bandwidth from each node is shown in kbps and graphs.

### 4.2 Intelligent fuzzy logic based Partial Completion Filter

Intelligent fuzzy logic decision disposes information based on fuzzy or non-fuzzy reasoning rules[10-12]. It makes self-adaptive decision in light of mature experience. The general fuzzy decision process consists of three parts: fuzzy quantitative disposal, fuzzy decision rules and fuzzy decision. The fuzzy quantitative disposal makes the real input parameter as a fuzzy set, and then the fuzzy decision carries out the output calculation based on the fuzzy set and fuzzy operators defined at fuzzy decision rules. This section will describe in detail how fuzzy logic can be utilized in DDoS flood attack intensity decision.

### 4.3 Attack intensity decision

Based on the basic theory and method of fuzzy mathematics, we propose an intelligent DDoS flood attack intensity decision system. DDoS flood attack intensity itself includes fuzziness, because the boundary between the light attack, moderate attack and severe attack is not well defined. So when judging the intensity of attack, one should take the intensity of background traffic into consideration. For example, a DDoS flood attack is considered as light attack if it causes slight decline of the network performance when the traffic load is high, but is considered as severe attack if it causes serious decline of the network performance when the network load is light [17]. In the proposed decision system, the DDoS flood attack intensity decision rules and operations are expressed by fuzzy sets, and then we feed these fuzzy decision rules and related information into knowledge repository.

The network elements take the dynamic process of actual attack into consideration, and then use fuzzy reasoning to determine the intensity of attack dynamically and intelligently. In this paper, the structure of fuzzy decision is two dimensional input and one-dimensional output. The two inputs are the Hurst parameter and its changing rate. The Hurst parameter reflects the influence of dynamic normal traffic on attack intensity and the changing rate reflects the influence of attack on normal traffic. The output is the intensity of the attack. As shown in Figure 3, the fuzzy decision process of the intensity of the attack consists of three parts: Hurst parameter and its changing rate fuzzification, fuzzy decision rules of attack intensity and fuzzy reasoning of attack intensity.

The description of each part of the fuzzy decision process is as follows:

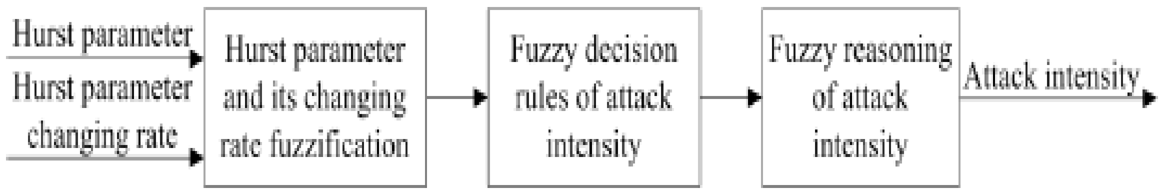
Hurst parameter
Hurst parameter changing rate
Hurst parameter and its changing rate fuzzification
Fuzzy decision rules of attack intensity
Fuzzy reasoning of attack intensity
Attack intensity

**Figure 3:** Fuzzy Decision Process

### 4.4 Hurst Parameter and its Changing Rate

**Fuzzification-** Fuzzification makes the real input parameters of Hurst parameter and its changing rate as fuzzy sets[15]. According to the change scope of Hurst parameter and its changing rate, we define the universe of discourse of the Hurst parameter as *UH*={0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1}; the universe of discourse of the changing rate as *UHC*={0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50}. The fuzzy sets of *UH* and *UHC* are *H'*={*S*, *M*, *B*} and *HC'*={*S*, *M*, *B*}, where "*S*" stands for small, "*M*" the moderate, and "*B*" the big. The variable's membership degree function of each fuzzy language satisfies normality assumption

$$\mu(x)=\exp\left[-(x-v)^2/b^2\right],$$

where *v* and *b2* are the mean and variance of the membership degree function. Through the above equation, we can obtain fuzzy judgment model of every parameter as well as the membership degree assignments of every fuzzy subset.

**Fuzzy decision rules of attack intensity-** The decision rules take note of the relationship between input fuzzy sets and output fuzzy sets [18]. Define the fuzzy decision result of DDoS flood attack intensity as a variable *L*, and the fuzzy set of *L* as *L'*={*LA*, *MA*, *SA*}, where "*LA*", "*MA*" and "*SA*" represent light DDoS flood attack, moderate DDoS flood attack and severe DDoS flood attack, respectively. Considering the relationship between Hurst parameter, its changing rate and DDoS flood attack intensity, we can get the fuzzy decision rules displayed in Table 1.

**Table 1:** Fuzzy decision rules

| *HC'* | *H'* | | |
|---|---|---|---|
| | *S* | *M* | *B* |
| *S* | *MA* | *LA* | *LA* |
| *M* | *SA* | *MA* | *LA* |
| *B* | *SA* | *SA* | *MA* |

**Fuzzy reasoning of attack intensity-** After fuzzifying the input parameters Hurst parameter and its changing rate, we can reason the intensity of attack according to decision rules presented in Table 1. For example, when the Hurst parameter is considered moderate, we infer there is a light DDoS flood attack if the changing rate of the Hurst parameter is considered small. In a similar way, there is a moderate DDoS flood attack if the changing rate of the Hurst parameter is moderate, and severe DDoS flood attack if the changing rate of the Hurst parameter is big.

**Description**-Distributed denial-of-service detection:
*start listener
*create a node and enter node name as 'server'
*create two or more nodes and enter its name as client1, client2...

The concept is that, we create a network containing a server and multiple clients. Client nodes will be sending data to server node. The server node will display the bandwidth (speed) at which the client nodes sending the data. The client nodes can send data in two modes: request mode and attack mode. While sending in request mode, the data will be sent at normal speed. While sending in attacking mode, the data will be sent in abrupt speed. That is, too much requests will be sent to the server which slows down the available network bandwidth of the server. After some time, click the prevent button in the server node. The client nodes which are sending data at bandwidth greater than threshold bandwidth are kicked out from the network. That is, the attack nodes are identified and are removed from the network. They no longer can communicate with the server. The data we are sending is a big matrix. The user needs to specify the number of rows and columns and a big matrix will be created with random 'double' data type numbers.

**Network Setup-** A network is setup with configurable number of nodes in the system. The nodes in the setup can communicate each other. We generate huge data in the form of matrices in different nodes, and the data are sent between the nodes.

**Packet Sniffer-** We capture the packets sent between the computers or through any network interface available in the computer. The sniffer works by calling WinPcap DLL, through JNI (java native interface).

**Packets Data Parser-** Here, we decode captured or stored (tcpdump files) packets for obtaining the packet data and its header fields like source ip, destination ip, source port, destination port, ttl, length, etc. Both incoming and outgoing data are captured so that data routed to and from a specific computer can be separated.

**Attack Detection-** Here, we implement the proposed method to detect the attack nodes. The nodes, which send data in abrupt speed are isolated in the network, by analyzing the network traffic between each node in the network.

**Results and Analysis-** The proposed method is tested with various network traffics, and a log containing the statistical data like time, kbps, etc. is generated.

## 5. Experimental Results

In this section, we conduct sets of experiments on data sets in terms of the classification error using the optimum training sets for testing. The classification performance is investigated by considering the performance with the lowest error rate and the corresponding value of k.

### 5.1 Evaluation of Attack Detection

The performance evaluation of proposed system is shown using accuracy rate and recall rate. Its measurement using F-measure with matrix confusion standard are shown in the Table 2.

**Table 2:** Confusion matrix for a binary classification problem in which the classes are not equally important

| *Actual class* | *Predicted Class* | |
|---|---|---|
| | + | - |
| + | TP | FN |
| - | FP | TN |

- The **True Positive (TP):** corresponds to the number of positive examples correctly predicted by the classification model.
- The **False Negative (FN):** corresponds to the number of positive examples wrongly predicted as negative by the classification model.
- The **False Positive (FP):** corresponds to the number of negative examples wrongly predicted as positive by the classification model.
- The **True Negative (TN):** corresponds to the number of negative examples correctly predicted by the classification model.

Accuracy means probability that the algorithms can correctly predict positive and negative examples. Sensitivity means probability that the algorithms can correctly predict positive examples. Specificity means probability that the algorithms can correctly predict negative examples.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (3)$$

The purpose of Experimental results is to show synflood attacks in the high speed network. In the High speed network attacker can attack server system by sending more requests as a authorized host. Finally attacker denial the service of actual user.

The following are the Fuzzy PCF experimental results:

**Table 3:** Attacks on Victim IP

| *Source IP* | *Victim IP* | *Attack Name* | *No.of Syns* |
|---|---|---|---|
| 192.168.1.1 | 192.168.1.2 | Syn-flood | 512 |
| 192.168.1.4 | 192.168.1.2 | Syn-flood | 512 |
| 192.168.1.5 | 192.168.1.2 | Syn-flood | 512 |
| 192.168.1.10 | 192.168.1.2 | Syn-flood | 512 |
| 192.168.1.12 | 192.168.1.2 | Syn-flood | 512 |
| 192.168.1.14 | 192.168.1.2 | Syn-flood | 512 |
| 192.168.1.16 | 192.168.1.2 | Syn-flood | 512 |

The table shows the evaluation of the proposed method Fuzzy Partial Completion Filter, a prototype of DDoS attack system has been established, as an example, the SYN Flooding, which is the most well-known DDoS attacks, is employed in our method. We used this proposed approach for attacking detection, and the proposed method works as the packet filter at the victim side. For contrast, we also implement the 32 bits strict filtering algorithm of FPCF.

**Table 4:** Performance Comparison based on Testing Criterion with Statistical, Partial Completion and Fuzzy Partial completion Filters

| Testing Criterion | Statistical Filter | | Partial Completion Filter | | Fuzzy Partial Completion Filter | |
|---|---|---|---|---|---|---|
| | Correctly Classified Instances | Time to build Model (seconds) | Correctly Classified Instances | Time to build Model (seconds) | Correctly Classified Instances | Time to build Model (seconds) |
| Training set itself | 51.28 | 0.15 | 91.81 | 0.74 | 76.55 | 0.01 |
| Percentage split (50% train 50% test) | 59.67 | 0.07 | 69.91 | 0.57 | 70.8 | 0.01 |
| Percentage split (70% train 30% test) | 58.09 | 0.08 | 74.26 | 0.44 | 75 | 0.01 |
| Percentage split (80% train 20% test | 56.04 | 0.08 | 67.03 | 0.42 | 74.73 | 0.02 |

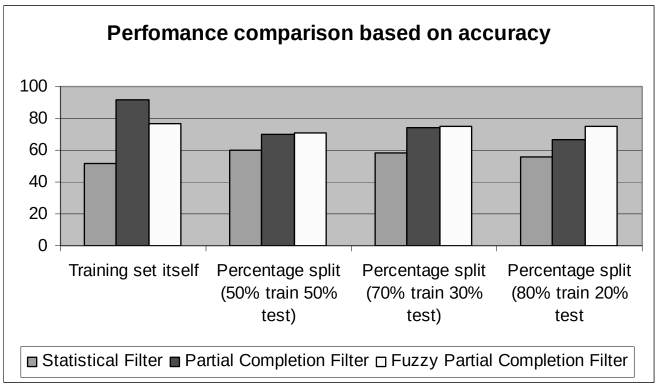

**Figure 4:** Comparing Accuracy among Statistical, Partial Completion and Fuzzy Partial completion Filters

As shown in Figure 4, the highest accuracy was observed. Despite the high accuracy rate of Fuzzy Partial Completion filter, the accuracy curve is unstable when the data is spilt into training and test, whereas Statistical and Partial Completion filter show stable accuracy for the same dataset. The accuracy rate of Statistical Filter is the lowest among the three algorithms.

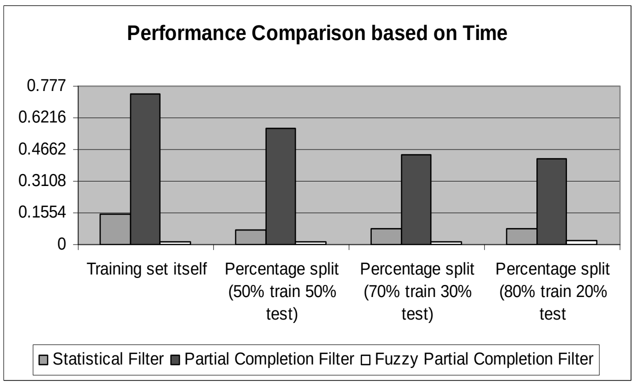

**Figure 5:** Comparing Learning Time among Statistical, Partial Completion and Fuzzy Partial completion Filters

Figure 5 illustrates the learning time comparison of the algorithms. The Fuzzy Partial completion Filter algorithm consumes far more learning time than the other algorithms. The learning time of Fuzzy Partial completion Filter drops drastically at percentage split of 50% and 70%. The learning time of Statistical drops at percentage split of 50%. The differences in learning time for Partial Completion filter for different percentage split was found to be not significant.

**Table 5:** Classification Result of Fuzzy Partial Completion Filter

| *Category* | *Number of Testing Packets* | *Correct Classification* | *Incorrect Classification* |
|---|---|---|---|
| Normal | 4813 | 4566 | 247 |
| Attack | 124562 | 123159 | 1403 |

The classification result of testing dataset using proposed algorithm only shows in Table 5. According to the statistics the system can correctly detect 94.87% for normal traffic and 98.87% for attack traffic. It incorrectly classified traffic in 5.13% for normal class and 1.13% for attack class. Out of total number of packets (Attack + Normal packets) 129375, 1650 packets are classified as others.

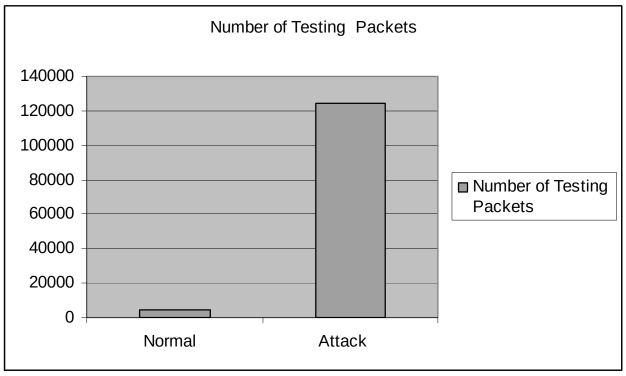

**Figure 6:** Number of Testing Packets

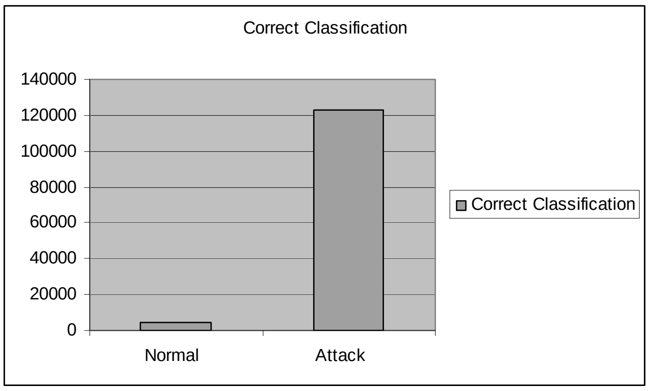

**Figure 7:** Correct Classification

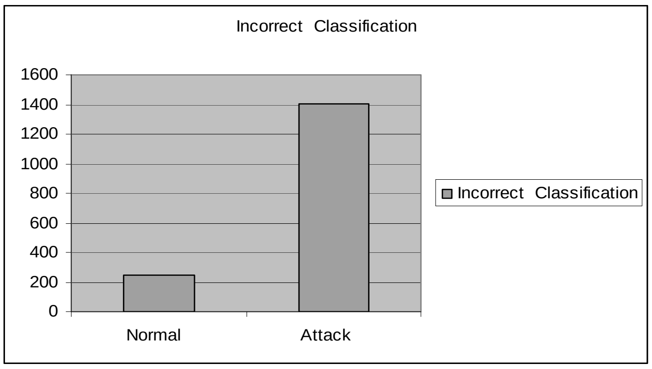

**Figure 8:** Incorrect Classification

### 5.2 Recall and Precision:

These are two widely used metrics employed in applications where successful detection of one of the classes is considered more significant than detection of the other classes. A formal definition of these metrics is given below. The recall and Precision values are already classified.

$$\text{Precision, p} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall, r} = \frac{TP}{TP + FN} \quad (2)$$

**Table 6:** Performance comparison based on Precision and Recall

| *Algorithm* | *Precision* | *Recall* |
|---|---|---|
| Statistical Filter | 68.2 | 69 |
| Partial Completion Filter | 75.6 | 76 |
| Fuzzy Partial Completion Filter | 96.3 | 98.2 |

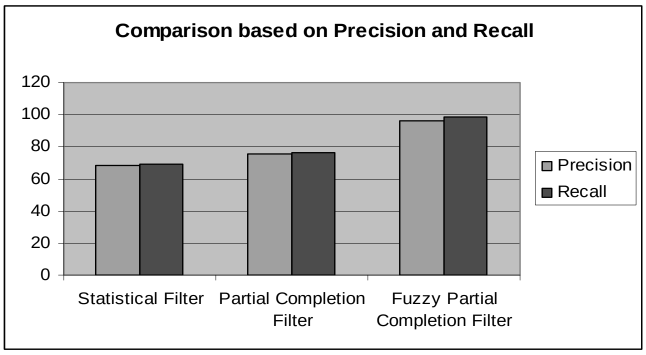

**Figure 9:** Comparison based on Precision and Recall

## 6. Conclusion

It appears to be widely perceived that detecting intrusions scalably within the network is a bad idea. Unfortunately, that causes security devices to choose between performance (which requires low memory) and completeness (which appears to require per-flow state). This paper is a gentle first step towards suggesting that this tradeoff may not be as Draconian as is commonly thought. While the general problem is still very hard (and indeed for attacks such as evasion attacks, we believe that aggregated solutions cannot work without causing unacceptably high false positives), our paper shows some progress for bandwidth- based and partial completion DoS attacks, and scan-based attacks including worms.

This paper explores this possibility in the specific context of DoS attacks and scan attacks by introducing fuzzy partial completion filter. While the existing approaches have not harped on this point of membership value, doing DoS detection in the network also finesses the need for traceback and/or manual intervention, and allows enterprise networks and ISPs to automatically filter out attacks before they enter (or leave) their networks. More fundamental than the specific techniques discussed in this paper is the general question of scalable behavior-based detection of attacks within the network.

This paper concentrates on considerable attention in the research and product literature, and solutions that scale are developed under java platform. As security functions become more prevalent in the edge first and then the core, it is natural to expect the same attention to be paid to scalable security solutions. More than just introducing the question and suggesting a specific mechanism for some problems, our paper shows that the issues of *behavioral aliasing* and *spoofing* are key questions that must be addressed in any scalable solution, even if the only response is to simply ignore the problem. These two provide a simple lens to view existing and future work in attack detection, and can perhaps suggest new solutions to an even broader class of attacks. The experimental result concludes that the proposed fuzzy based partial completion filter performs the detection of IDS very effectively comparing the existing approaches represented in this work.

## References

[1] M. Roesch, Snort. [Online]. Available: http://www.snort.org
[2] P. Barford, J. Kline, D. Plonka, and A. Ron, “A signal analysis of network traffic anomalies,” in *Proc. 2nd ACM SIGCOMM Internet Measuremen Workshop*, 2002, pp. 71–82.
[3] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, “Sketch-based change detection: Methods, evaluation, and applications,” in *Proc. 3rd ACM SIGCOMM Internet Measurement Conf.*, 2003, pp. 234–247.
[4] S. J. Staniford, “Containment of scanning worms in enterprise networks,” *J. Computer Security*, 2004, to be published.
[5] ForeScout Technologies. [Online]. Available: http://www.forescout.com
[6] D. Moore, G. Voelker, and S. Savage, “Inferring Internet denial of service activity,” in *Proc. 10th USENIX Security Symp.*, Aug. 2001, pp. 9–22.
[7] Mazu Publishing. [Online]. Available: http://www.mazu.com
[8] Arbor Networks. [Online]. Available: http://www.arbornetworks.com
[9] H.Wang, D. Zhang, and K. Shin, “Detecting SYN flooding attacks,” in *Proc. IEEE INFOCOM*, 2002, pp. 1530–1539.
[10] V. Paxson, “Bro: A system for detecting network intruders in realtime,” *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, 1999.

[11] K. Levchenko, R. Paturi, and G.Varghese, “On the difficulty of scalably detecting network attacks,” in Proc. 11th ACM Conf. Computer and Communications Security, 2004, pp. 12–20.
[12] R. Keyes, “The Naptha DoS vulnerabilities,” [Online]. Available:http://www.cert.org/advisories/CA-2000-21.html
[13] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, “A taxonomy of computer worms,” in Proc. ACM Workshop of Rapid Malcode (WORM), 2003, pp. 11–18.
[14] S. Staniford, V. Paxson, and N. Weaver, “How to 0wn the Internet in your spare time,” in Proc. 11th USENIX Security Symp., Aug. 2002, pp. 149–167.
[15] MyDoom.B Virus. [Online]. Available: http://www.us-cert.gov/cas/techalerts/TA04-028A.html
[16] CERT Advisory CA-2001-19, “Code Red” Worm Exploiting Buffer Overflow In IIS Indexing Service DLL, [Online]. Available:http://www.cert.org/advisories/CA-2001-19.html
[17] CERT Advisory CA-2001-26 NimdaWorm, [Online].Available: http:// www.cert.org/advisories/CA-2001-26.html
[18] CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks,[Online]. Available: http://www.cert.org/advisories/CA-1998-01.html
[19] V. Paxson, “An analysis of using reflectors for distributed denial-ofservice attacks,” Comput. Commun. Rev., vol. 31, no. 3, Jul. 2001.
[20] T. M. Gill and M. Poletto, “MULTOPS: A data-structure for bandwidth attack detection,” in Proc. 10th USENIX Security Symp., 2001, pp. 23–38.

## Author Profile

**Abilesh** received the M.Sc. Degree in Information Technology from Shri Nehru Maha Vidyalaya College of Arts & Science, Affiliated to Bharathiar University, Coimbatore in 2012 and pursuing M.Phil. degree in Computer Science from Sri Ramalinga Sowdambigai College of Science & Commerce (Affiliated to Bharathiar University), Coimbatore.