An Aggregative Approach For Scalable Detection of DoS Attacks

by

Alireza Hamidi

B.Sc., Sharif University of Technology, Tehran, Iran, 2006

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Computer Science

University of Victoria

An Aggregative Approach For Scalable Detection of DoS Attacks

by

Alireza Hamidi

B.Sc., Sharif University of Technology, Tehran, Iran, 2006

Supervisory Committee

Dr. Sudhakar Ganti, Supervisor (Department of Computer Science)

Dr. Gholamali C. Shoja, Departmental Member (Department of Computer Science)

Dr. Kui Wu, Departmental Member (Department of Computer Science)

Dr. Stephen Neville, External Examiner (Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Sudhakar Ganti, Supervisor (Department of Computer Science)

Dr. Gholamali C. Shoja, Departmental Member (Department of Computer Science)

Dr. Kui Wu, Departmental Member (Department of Computer Science)

Dr. Stephen Neville, External Examiner (Department of Electrical and Computer Engineering)

Abstract

If not the most, one of the serious threats to data networks, particularly pervasive commercial networks such as Voice-over-IP (VoIP) providers is Denial-of-Service (DoS) attack. Currently, majority of solutions for these attacks focus on observing detailed server state changes due to any or some of the incoming messages. This approach however requires significant amount of server's memory and processing time. This results in detectors not being able to scale up to the network edge points that receive millions of connections (requests) per second. To solve this problem, it is desirable to design stateless detection mechanisms. One approach is to aggregate transactions into groups. This research focuses on stateless scalable DoS intrusion detection mechanisms to obviate keeping detailed state for connections while maintaining acceptable efficiency. To this end, we adopt a two-layer aggregation scheme termed Advanced Partial Completion Filters (APCF), an intrusion detection model that defends against DoS attacks without tracking state information of each indi-

vidual connection. Analytical as well as simulation analysis is performed on the proposed APCF. A simulation test bed has been implemented in OMNET++ and through simulations it is observed that APCF gained notable detection rates in terms of false positive and true positive detections, as opposed to its predecessor PCF. Although further study is needed to relate APCF adjustments to a certain network situation, this research shows invaluable gain to mitigate intrusion detection from not so scalable state-full mechanisms to aggregate scalable approach.

Table of Contents

List of Tables

List of Figures

Abbreviations

TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
DoS	Denial of Service
DDoS	Distributed Denial of Service
PCF	Partial Completion Filter
APCF	Advance Partial Completion Filter
ANN	Artificial Neural Network
VoIP	Voice over Internet Protocol
ISP	Internet Service Provider
PSTN	Public Switched Telephone Network
QoS	Quality of Service
SVM	Support Vector Machine
SIP	Session Initiation Protocol
BAC	Behavior Alarming Counter
ROC	Relative Operating Characteristic
RTP	Real-time Transport Protocol

Acknowledgements

I wish to express my sincere gratitude to my supervisor, Dr. Sudhakar Ganti, for his advice, supervision, and crucial contribution, which made him the backbone of this research and so to this thesis. My deepest thanks are also due to the members of the supervisory committee, Dr. Shoja and Dr. Wu, for their invaluable assistance, support and guidance.

Many thanks go in particular to Dr. Kui Wu, to whom I am indebted for his valuable advice in science discussion and supervision in writing my paper.

I also wish to express my love and gratitude to my family, for their understanding and endless support, throughout the duration of my studies. I appreciate you with all my heart.

Chapter 1

Introduction

1.1 Description

The Internet provides an open low-cost platform over which many new applications are developed. These applications can roughly be categorized into two groups: connectionless applications and connection-oriented ones. While connection based applications provide reliable, controllable and in-order communications, they suffer from drawbacks such as connection establishment overhead, bandwidth usage and security issues due to connection oriented nature.

One of the most hazardous security threats for connection based protocols is the Denial-of-Service (DoS) attacks. These have attracted significant attention among both the attackers and network administrators [25] due to their significant destructive nature, variety, and relatively easy methods to launch them. Protecting VoIP (or any connection oriented communication paradigm) services from these malicious attacks is of great importance especially when commercial or industrial revenue is at stake. The Achilles' heel is the signaling policy that each protocol maintains. Its main duty is to initiate a session and terminate the connection when done. By exploiting this part, attackers can launch heavy loads of fake connections on the victim server and prevent it from serving legitimate users.

As the extent of DoS attacks grew, network security measurements attracted more

attention. There are generally two complementary approaches to network security: *prevention* and *detection*. Prevention-based methods use authentication and encryption to ensure that users conform to predefined security policies. They can keep most illegitimate users from entering the system. However, preventing an outsider from exploiting network vulnerabilities without entering the system entails an ability to predict a malicious behavior, and a system always contains weak points which are hard to predict; as such it is necessary to deploy multi-layer defense [22]. *Intrusion detection* comes into place to serve as the second wall of defense by helping the system identify malicious activities. This thesis focuses only on the intrusion detection.

There are two classical approaches to intrusion detection: *signature-based detection* and *anomaly-based detection*. Signature based detection [26] relies on typical characteristics of a malicious packet or a reassembly of several packets, whereas anomaly-based detection searches for unusual behavior across a set of packets, usually without reassembling them. While signature-based approach is very powerful and accurate in detecting many known attacks (e.g. existing worms and viruses), it is not helpful in detecting attacks whose signature is unknown or hard to characterize [22]. Anomaly-based approaches become very useful to defend against unknown attacks because they do not rely on signatures but on detecting abnormal system behavior. They try to discover common behaviors among intrusions in terms of network characteristics in a sequence of packets. Easily obtained information such as header fields could be extracted to characterize this sequence, which is termed as a *flow*.

Precision and *scalability* are two important factors for Intrusion Detection Systems (IDS), around which much of the research has been focused. The ideal requirement is to obtain a high precision - low and controllable false alarm - with the capability to scale up to gigabit per second link speeds in an edge router. In the next section we present some of the most relevant research with respect to these issues.

1.2 Current Research

Two major challenges of the current and the next generation of IDS devices are [26]:

i) to provide real time or wire-speed intrusion detection so as not to miss attacks at high speed

ii) to reduce false positives.

Signature based intrusion detection approach, utilized in IDS from vendors such as NetScreen [3], Cisco [2], Checkpoint [1] *etc.,* the decision is formed on the basis of knowledge of a model of the intrusive process [6]. According to Axelsson *et al.*, [6] signature based IDS can be divided into four categories:

1. *State-modeling:* It encodes the intrusion into a number of states. States structure and their interactions determine if the intrusion has taken place or not. They are by their nature time series models [18].

2. *Expert-system:* These systems reason about the security state of the network, given rules that describe intrusive behavior with complicated nature. They are often of considerable power and flexibility, comes at a cost to execution speed [5, 34].

3. *String matching:* Simple substring matching in text that is transmitted between systems [41].

4. *Simple rule-based:* They are similar to expert systems, but not as advanced. They have origins from intelligent training mechanisms [32, 20].

Signature based detectors try to find clues or patterns that are thought by the designer to be representative of a potential attack, irrespective of the background traffic behavior. Therefore, they need to reassemble packets in the network layer to discover the suspicious patterns, and the designer should bring them up-to-date

as new patterns emerge. Also they need to maintain per-flow states in most of the case [41]. Thus, although they obtain notable precisions, this type of IDS is not efficient at high wire-speed throughputs or at high traffic network points.

In anomaly based IDS, the detector postulates certain states as *normal* behavior of the network [6]; hence, any violating behavior is marked as anomaly and could be a potential intrusion [23]. According to Axelsson *et al.*, [6], two major groups of anomaly based IDS are: *self-learning* and *Programmed* detectors. In the former, the detector is trained and learns by examples what constitutes normal for the installation, typically by observing the traffic. In the later, on the other hand, IDS needs to be programmed by someone to detect certain anomalous events.

Self-learning detectors divide into **"time series"** and **"non-time series"** categories.

- Non-time series learn the normal behavior of the system by the use of a stochastic model irrespective of time series behavior. Examples are *rule modeling* and *descriptive statistics*. In the first one the system formulates some rules based on the traffic studies it carries on, and raises alarm when a poor weighted match with the observed traffic occurs [11]. In the second one the system builds a statistic model from certain network parameters, construct distance vectors for the observed traffic and the profile, and raises the alarm when the distance exceeds certain threshold [34, 20, 5].

- Time series self-learning IDS takes time sequence into account and have a more complex nature. Examples of these type of detectors are Hidden Markov Model and Artificial Neural Networks [31].

Programmed detectors fall into two categories as well: **"descriptive statistics"** and **"default-deny"**.

- *Descriptive statistics* systems build a profile from some normal statistical be-

havior of the network, with respect to a number of parameters; these parameters can be any traffic feature of the network. The idea is to provide a higher layer of detection with the statistical profile made by these detectors. The information is usable for simpler rules in the same layer as well, such as a threshold over a specific parameter [44, 26, 12, 22].

- *Default-deny* IDS models the normal behavior in a stringent model and mark all violations as intrusions. **State series modeling** falls into this category since it models the normal network in state series [32].

Figure 1.1 summarizes the above discussion on various IDS in a hierarchy. In this work, we focus on programmed anomaly detection based IDS that utilizes descriptive statistics with certain thresholds.

1.3 Motivation

When a detection system is deployed at the edge of a network, it should be able to operate at high link speeds (e.g. 1 Gb/s or higher). Detection systems that maintain per-flow state information and handle huge number of flows, such as signature based IDS or default-deny anomaly detectors, can create implementation problems at these speeds. The reason is that they have to either reassemble single packets, keep information for each flow or connection, or even both. Besides, signature based approach is particularly not helpful in detecting attacks which are not characterized by signature within a single packet, but by unusual behavior across a set of packets.

Scalability of detection system has become an important and hard problem to solve. Currently, the detection of several types of attacks such as evasion and TCP hijacking has been proved impossible to perform in a scalable fashion [26], and some attacks such as bandwidth attacks already have scalable solutions [16]. Distributed DoS (DDoS) and scanning attacks, however, are still unattended [22]. Existing solutions like *Partial Completion Filters* (PCF) [22] introduce a scalable approach by

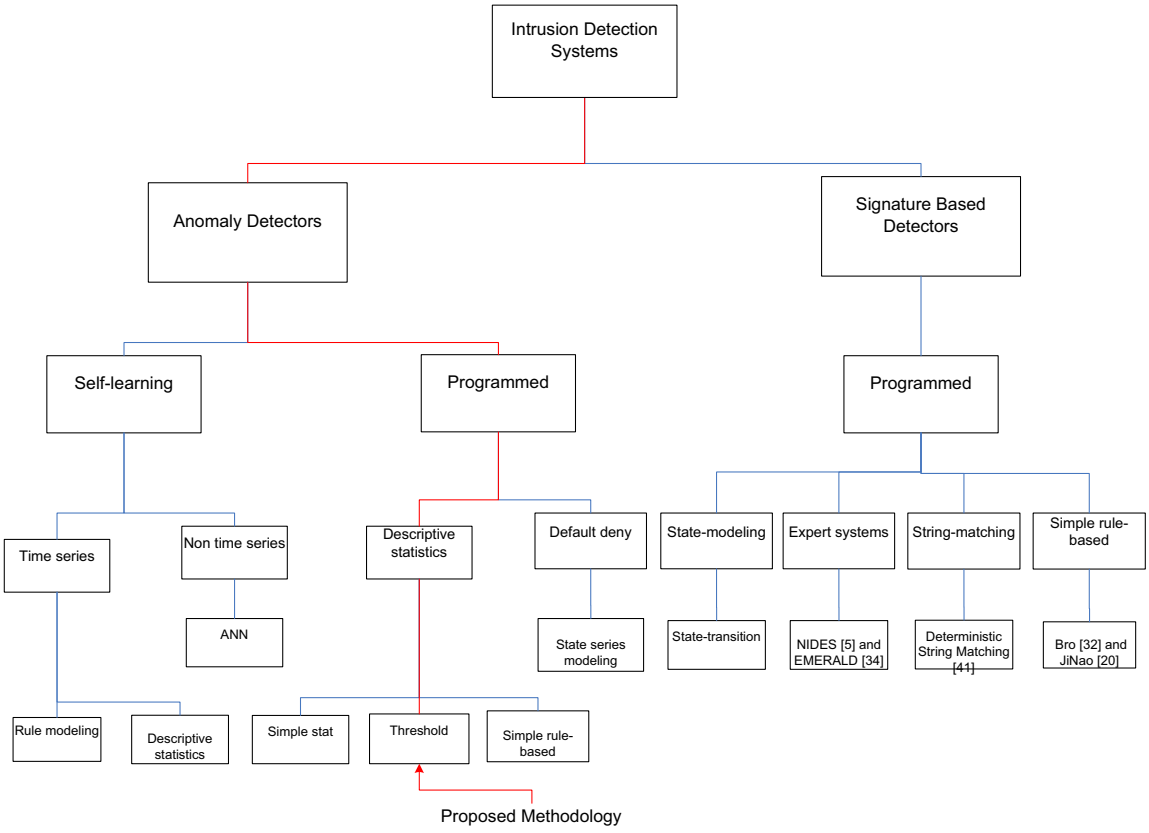


Figure 1.1: General Classification of IDS Technology

aggregating incoming flows and by not tracking each individual connection; this way, the detector has less intensive memory and processing requirements. PCF, however, is still not efficient with respect to false detection ratio and sensitivity. By false detection ratio we mean what portion of alarms have been false alarms, and by sensitivity we measure the portion of attacks that have not been detected. The question is: *Is it possible to design a scalable detection system that works at high link rates and at the same time provides high detection ratio and low error rate?*

The intention of this thesis work is to provide a positive answer to the above question. PCF mechanism is appropriate for this purpose for several reasons. The implementation is easy on a hardware module on a router without imposing significant additional resources to the usual functionality; it uses already extracted packet header information for its processing. Aggregation of flows causes a significant reduction in the memory needed to keep this information; in addition, the amount of data that has to be stored for every aggregated group is very limited. The intrusion detection mechanism, which is anomaly based on programmed, descriptive statistic approach, is applicable to a high data rates and it is a combination of simple counters and comparators.

This thesis extends the functionality of PCF to include better detection mechanisms by introducing *Advanced PCF (APCF)*. In this thesis we:

- change the structure of PCF to make it more accurate and easy to control.
- prove that our new detection model is more sensitive to attacks and more robust against behavioral aliasing (see Chapter 2). This *efficiency* is measurable and considerably under control.
- show that in noisy environment, which is a matter of concern in almost all practical networks, our new model behaves more reliably than PCF since it can preserve same efficiency while its sensitivity can be adjusted.

1.4 Thesis Organization

This work describes an improvement over an existing IDS approach to ameliorate the false detection rate and sensitivity, while maintaining the scalability to high data rates. The remaining of the thesis is organized as follows. Chapter 2 describes the PCF model and some other recent scalable approaches for intrusion detection. Chapter 3 elaborates on the problem space, sets forth the assumptions that are made and provides detail specification of our solution. This chapter also includes the theoretical analysis of the proposed model, the specification of its parameters and their relationship. Simulation of the model in a network environment and performance evaluation results are provided in Chapter 4. We conclude the work finally in Chapter 5 with a summary of major contributions and possible directions for future work.

Chapter 2

Background

A denial-of-service (DoS) attack is characterized by an explicit attempt to prevent the legitimate use of a service [29]. DoS intrusions potentially target any connection-oriented network protocol. This does not entail that DoS attacks can not go beyond such networks; it means that any communication which is based on a signaling policy is potentially exposed to DoS attacks. Partial Completion Attack is a most significant form of DoS attack in which the attackers try to overwhelm the serving capacity of the target by initiating many useless sessions and refrain from terminating them. The classic example of such attack is syn-flooding [29] in TCP networks. In syn-flooding, the attacker sends TCP SYN packets with spoofed IP addresses and initiates numerous connections to the server, without terminating any of them.

In this work we consider Voice over IP signaling protocol as an example of applying the DoS attack detection mechanism due to the following reasons:

- VoIP is easy to exploit by attackers because it uses text based signaling protocols that do not use any encryption or authentication mechanism by default, such as SIP (Session Initiation Protocol) [36] and H.323 [45];

- VoIP technology is gaining excessive industrial attention. From 2005, the approximate VoIP subscribers has increased by 115 million. Skype, the leader in today's VoIP global market, has over 170 million users currently. The market

for VoIP equipment is growing at around 25% per year [27]. This investment comes with all kind of intruders trying to investigate security holes and torment systems, and therefore, necessitates more attention towards the defense mechanisms in such networks.

- SIP as a major signaling protocol for VoIP, provides a simple signaling policy which can easily be mapped to TCP connection establishment. Hence, the contribution of this research, given that it reduces penetrability against intrusions in VoIP, can be generalized to other connection oriented web services on world wide web.

We refer to VoIP service providers as ISPs; Indeed, the study of this work is applicable to other types of Internet Service Providers. In the remaining of this chapter we present the background on different attack mechanisms and different IDS defense principles, as well as introduce our contribution.

2.1 DoS Attacks over VoIP Networks

VoIP technology is growing fast and is becoming more widely deployed due to its advantages over traditional PSTN (Public Switched Telephone Network) services [29, 43]. The twofold reason includes economical advantage and wider range of advanced services. Therefore, although currently the most common targets of DoS attacks are web servers, VoIP entities (servers and clients) are attracting more attention among intruders.

A VoIP network constitutes of endpoints and Call Controllers[1]. SIP follows a very simple structure. If a client from ISP A (client1@ISP-A) wants to call a client in ISP B (client2@ISP-B), an INVITE message is sent from client1 to the ISP A controller indicating client2@ISP-B as the callee. The ISP A controller opens a session for

[1]VoIP networks can have various other parties and contributors. Call Controllers are also termed as Call Proxy Servers.

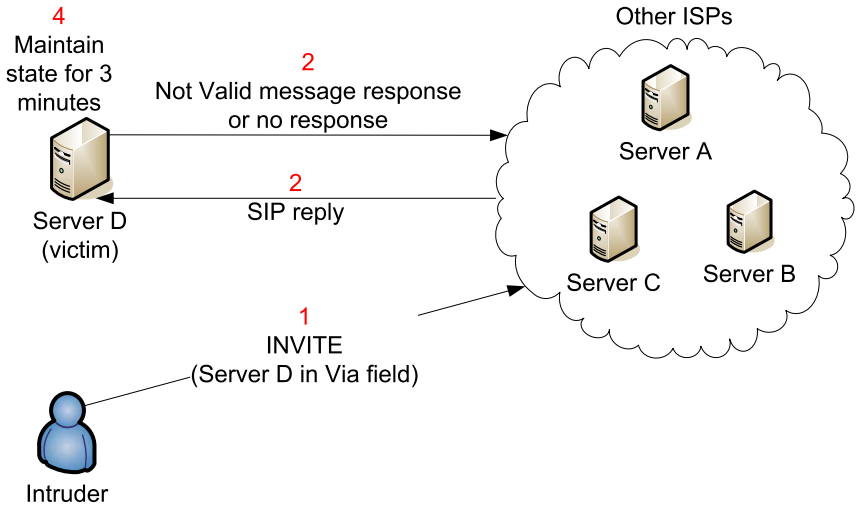
the connection by holding information of client1 and client2 in the memory. This information includes both sides' addresses, the current state (calling, trying, refused, canceled, connected, etc.,), duration, technical details of the network, devices, data streams, etc. ISP A controller then forwards the message to ISP B controller, which in turn keeps the state of the connection and forwards the message to client 2. Based on the response from client 2, the state of the connection changes and connection will be established until one of the clients closes the session.

According to the VoIP threat taxonomy compiled by VOIPSA [45], there are four basic DoS attack categories targeting VoIP networks. Following sections discuss them in detail.

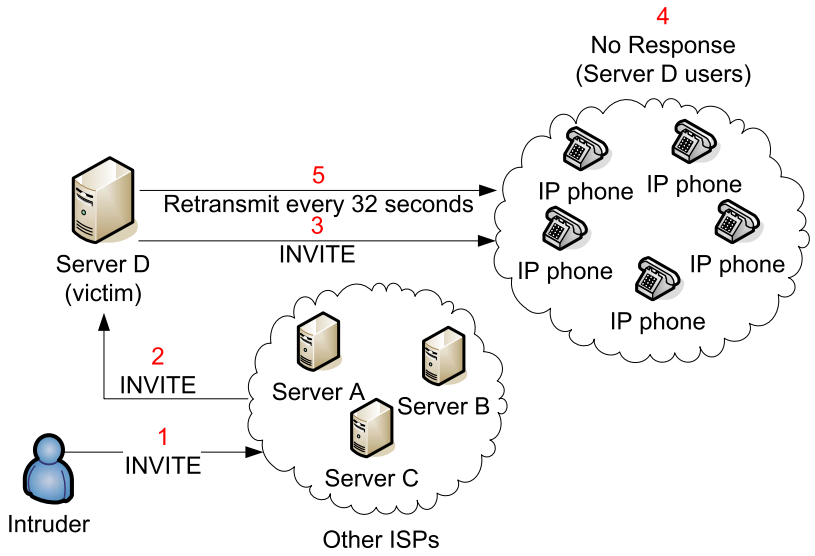
2.1.1 Request Flooding

The attacker tries to overwhelm a victim server (controller) or even an endpoint by sending abundance of requests. These requests are usually legitimate INVITE signals aimed to initiate a session, not proceeded by any session termination signals. The intruder might send messages directly, given that the user has a valid access to the network, or might have actual members of the network create the ambush. This activity is performed by infecting as many legitimate users as needed by necessary viruses/trojans beforehand.

Request flooding attacks vary from very simple, easy to trace forms (single source attacker) to complex frustrating forms, specially when *spoofed messages* are involved in the process. For instance, Distribute Reflection DoS (DRDoS) [33] intrusions are requests with a spoofed IP address as the requester (i.e., the victim) that are sent to a large number of SIP proxy servers (i.e., reflectors); thereupon, the victim will be swamped with the subsequent response messages (not found, authentication challenge, call does not exist, etc.), causing a DRDoS attack [38]. In this case, if a compromised user in the victim network answers back to some of replies, the server has to maintain the state for at least 3 minutes which in turn exacerbates the



(a) A common DRDoS attack Reply Forwarding



(b) A distributed partial completion attack Response Forwarding

Figure 2.1: Examples of DoS Attacks in a VoIP Network

situation [40]. Figure 2.1(a) illustrates such a scenario. In Figure 2.1(b) the attacker uses groups of users to send request to fake destinations through the victim server; since there is no response, the server keeps re-sending the request every 32 seconds. Remember that the important characteristic of all these attacks is that they are not proceeded with any termination signals. Request Flooding is also known as Partial Completion attacks [22]. They have vast variety and are hard to detect in a scalable way. Popular IDS used today are based on state transition policy [35, 38] or check routines at the expense of processor usage [40]. Different types of IDS in VoIP are discussed in Section 2.2.

2.1.2 Malformed Messages

The specifications for control messages in many VoIP implementations are kept open-ended to allow the addition of new capabilities in future. This type of specification imposes the problem that it is hard to test messages for being accurate or implementations for correct processing. Consequently, valid but complex messages are at risk of being discarded, and the processing systems themselves are vulnerable against sufficiently devious invalid messages. This, gives complex invalid messages the ability to be accepted by a call processing element and to trigger self-destructive behavior in endpoints or proxy servers [45]. This type of intrusion challenges the operating systems or server implementations. It can be avoided by well designed implementations and stricter authentication.

2.1.3 QoS Abuse

VoIP is a QoS sensitive application. This can be exploited by an attacker in which the user violates the QoS negotiated at setup. For example, the user could use a different media coder than what was declared during call setup. Also the user can send periodic bursts of packets at the rate equal to or greater than the bottleneck capacity. The burst period is tuned to be equal to the round trip time (in TCP

or VoIP). It is also possible for data applications to encroach on or misuse the QoS defined for voice. This would have the effect of introduced latency which adversely affects voice quality during a call.

The effect of such an attack has been discussed in [39]. The paper suggests that QoS abuse can reduce the quality of calls, from good quality to acceptable quality or from acceptable to a full DoS. Nevertheless, the impact is not severe enough to prevent a network from serving in an ambush. This reduces the necessity of scalable approach .

2.1.4 Call Hijacking

The system is susceptible to a call hijacking attack when information exchange security between two endpoints is compromised. This type of attack deals with stealing data or interrupting an ongoing session. The defense mechanism against such attack includes authentication and encryption, and call state peruse.

2.2 Anomaly Based IDS

As mentioned before, anomaly based IDS is able to scale up to higher network link rates, as opposed to signature based IDS. While higher link capacity and routing mechanisms have entailed scalable intrusion detectors, fast growing DoS attacks and the variety of such intrusions necessitated detection mechanisms to reduce the false detection ratio, in reply to significant financial costs [27]. In essence, the research in anomaly detectors has moved towards *programmed* statistical based IDS during past few years (see Chapter 1 Section 1.2). The reasons can be enumerated as follows:

- Self-learning approach relies on a training phase. This entails an accurate, pre-organized and up-to-date dataset from network behavior as the training data. When it comes to network vantage points in which data rate exceeds million packets per second, gathering this data becomes hard and troublesome if not impossible. Moreover, the training time prevents changes to be functional

immediately and imposes a latency to the detector, deviating the real-time functionality.

- Self-learning approach uses AI (Artificial Intelligence) mechanisms such as ANN (Artificial Neural Network) [17] or SVM (Support Vector Machine) [9]. These algorithms work as a black box or their process is too complex to interpret. On the other hand it is important for an IDS to maintain the visibility.

- Self-learning anomaly detectors build models in terms of rules, networks, etc., to detect violations in the network. Checking incoming packets against these models inflicts notable memory and processor usage on the detector hosts, vantage or core routers in our case.

Programmed anomaly detection can be a state machine programmed as a routine in the device, or as a descriptive statistical approach which defines an attack as an abnormal and noticeable deviation of some statistic of the monitored network traffic workload [10]. The former, *default deny*, models the accurate signaling protocol into a finite state machine. The model should contain all legitimate cases that might occur during a connection, from the initiation phase to the termination. Figure 2.2 exemplifies the call setup request process in SIP message transaction [38]. While the model stands for all kinds of eligible processes, any deviation is marked as abnormal. Default-deny methods have very high precision; nevertheless, since they have to keep all state information about any single connection, they impose notable memory and processing usage on the router. This drawback prevents state-transition based IDS to scale up to high data rate network points. Examples of implementations based on this approach can be found in [19, 37, 42, 38]. Sengar et al. in [38] for example, proposes communicating extended finite machines which are tested in a simulated network and the result shows no false detection at all. Nonetheless, call request arrivals in the simulated network in [38] did not go beyond 6 calls per second, and at

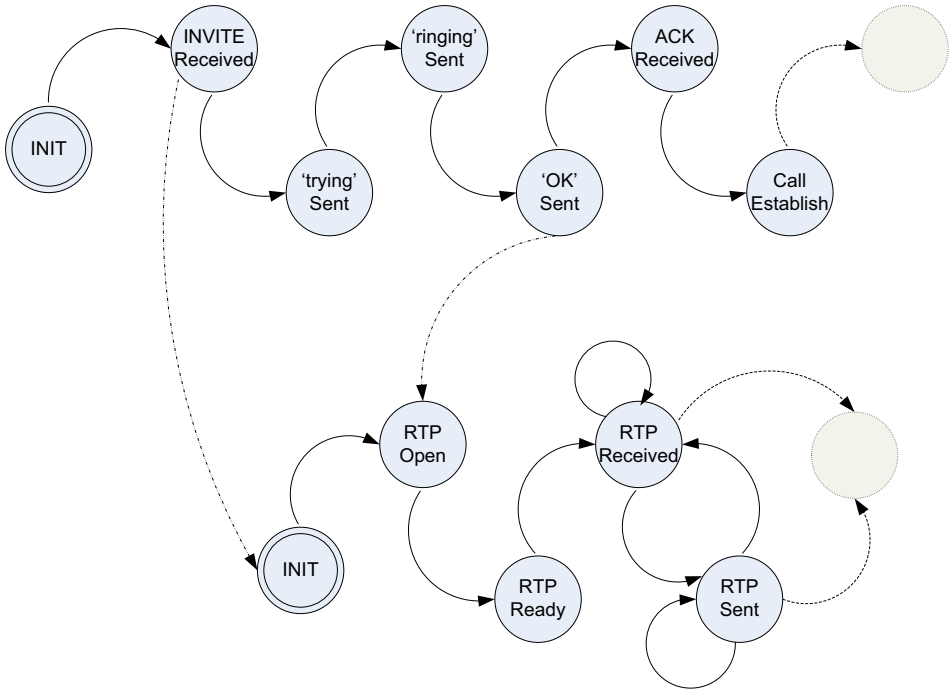


Figure 2.2: Call Establishment Finite State Machine in SIP [38]

the same time the CPU usage exceeds by 3.6% to run this IDS.

With the *descriptive statistical* anomaly detection approach, scalability is possible since data gathering and detection process reduce to easily accessible traffic or header data and predefined thresholds. The disadvantage, on the other hand, is that the precision for such systems is not satisfactory. The crucial difference between these systems lies on the choice of statistic selection.

In [8], network measurements are treated as *generic signals*, decomposed into distinct time series of average packet size per second. Wavelet analysis then is carried out on each time series to find the abrupt variability in localized high- and middle-spectral energies. This approach is complicated to trade off precision and scalability, and to implement on a plug-in module for a router.

The other method concentrates on easy accessible packet header data, which is already extracted in the router, to calculate simple statistics such as the number of session initiation or session termination signals. A threshold then is specified to find those connections with high signal imbalance. The idea here is to aggregate

multiple connections into groups and calculate simple statistics to avoid tremendous memory usage, while processing time has been reduced due to simple comparisons. A good example of this approach, Partial Completion Filters (PCF), is introduced by Kompella *et al.*, in [22]. This method is explained in the following section as the basis of this work.

2.3 Partial Completion Filters

Kompella *et al.*, [22] introduced a scalable statistical anomaly detection method based on *aggregation*, termed Partial Completion Filter (PCF), that avoids maintaining per-flow state and at the same time keeps false detection rate under control. There are two major challenges that any aggregative intrusion detection mechanism must deal with: *behavioral aliasing* and *spoofing*. Behavioral aliasing means aggregation of bad behaviors (e.g. open session without closing) to appear as innocent or vice versa. Spoofing means that an intelligent attacker studies the functionality of a system to send invalid packets to confuse the detector. The remedy is to improvise certain control mechanisms on the aggregation to effectively tackle the problem.

2.3.1 PCF introduction

PCF sits on a router at the edge point of a network, and aggregates connections based on some fields of each packet that can be easily extracted without any reassembly (e.g. source IP and destination IP addresses). PCF is mainly comprised of a hashing function (or several hashing functions in multi stage PCF [22, 14]) which hashes incoming connections into groups which are called *buckets*, and assigns a counter to each of them to hold the balance between connection establishments and terminations. Figure 2.3 presents a hashing scheme in PCF. In this example, all TCP packets originated from the same source IP address are aggregated into a bucket. Each bucket has been assigned a counter associated with a special feature. For example, the counter can be configured to increase for every SYN packet and decrease

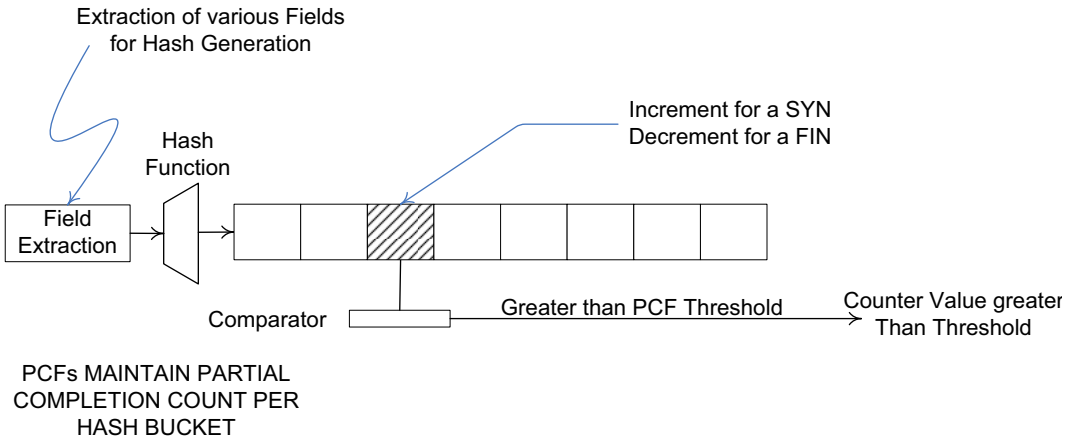


Figure 2.3: Partial Completion Filter

for every FIN packet passing through a monitoring point for TCP connections, or every $INVITE$ and BYE in the VoIP case.

Intuitively, under normal system traffic, SYN and FIN packets should be roughly balanced, meaning that the counter value of each bucket must be within a bound. This intuition has been proved right via mathematical analysis and experiments in [22]. The major concern of PCF is the errors caused by behavioral aliasing. Kompella *et al.* have shown that the errors resulted from behavioral aliasing are bounded by a Normal distribution. We have found that the errors in [22] can be under-estimated (see 3.2.1). The presented false positive rates are based on aggregated groups of connections, not individual ones. Therefore, the actual false detection ratio will be higher. This research introduces an Advanced PCF (APCF) to reduce error rate without sacrificing scalability in intrusion detection. PCF detailed Binomial behavior [22] is examined in the next section, before introducing the problem to solve. This makes our improvements easy to understand.

2.3.2 The Binomial Behavior of PCF

In their paper in 2004 and its successor in 2007 [22, 21], Kompella and Varghese have modeled PCF functionality in the network into Binomial behavior. Their reasoning is elaborated as follows. Let us assume that X_i is the value assigned (e.g., $+1$ for initiation and -1 for termination) for i^{th} incoming message, and the value of the PCF counter is maintained as $X = \sum_{i=0}^{n} X_i$. Then the value of PCF counter follows a Binomial distribution [22]. Since we expect the number of connection initiation messages to be equal to that of the terminations in a legitimate traffic scenario, the Binomial values of 1 and -1 are assigned with an equal probability of 0.5. Hence the mean of the Binomial distribution, $\mu_b = 0$, and the standard deviation of the Binomial distribution, $\sigma_b = 1$. According to the Central Limit theorem, for a large enough n, this Binomial distribution tends to a Normal distribution with $\mu_n = n\mu_b$ and $\sigma_n = \sigma_b\sqrt{n}$. Here, n is the number of valid messages[2] hashed to each bucket. Therefore, with confidence bounds of a and b, the following equation can be derived [22]:

$$P\left[a \leq \frac{X - \mu_n}{\sigma_n} \leq b\right] = P\left[a \leq \frac{X - n \cdot \mu_b}{\sigma_b\sqrt{n}} \leq b\right] = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-\frac{z^2}{2}} dz \tag{2.1}$$

With $a = -3$, $b = 3$, $\mu_b = 0$, and $\sigma_b = 1$ the above equation implies that $Pr\left[|X| \leq 3\sqrt{n}\right] = 0.9987$. For instance, if there are 3000 packets hashed to each bucket, then the equality indicates that the probability of a counter value lies in between 164 and -164 is 0.9987, given that all of the connections are benign.

[2]Initiation and termination messages are referred to as valid messages. In TCP they are SYN and FIN packets, in SIP they are INVITE and BYE/CANCEL messages. Packet/message in this thesis always means valid packets/messages.

2.4 Contribution

PCF relies on merely **counting** incoming initiation and termination signals. This way the functionality is extremely susceptible to behavior aliasing; a group of benign long connections when a TV show is airing, link noise and retransmission, and protocol deficiencies are examples that easily produce this mechanism. Moreover, PCF is extremely dependent on the duration of observation. The counter value is checked only after the duration is passed; if the duration is not longer than it should be, the attack impact might be counterbalanced by groups of termination signals. If the duration is too short, again attacks might remain undiscovered or innocent long connections might be labeled as intrusions. Furthermore, Kompella *et al.* do not provide a mechanism or a heuristic to find the proper PCF threshold or the observation durations.

The contribution of this thesis is to provide modifications to PCF in order to ameliorate these deficiencies. We introduce and discuss Advanced PCF in the following chapter. Specifically, besides signal counting, another counter is added to APCF that is used to measure malicious behavior in a group of signals. The concept of mere signal counting is replaced with a measure called Behavior Alarming Counter (BAC). This reduces the detector vulnerability against behavioral aliasing by a significant measure. Theoretical analysis has been provided to determine parameter values and thresholds. These parameters are leveraged as settings to adjust the detector to a given local network. Furthermore, we provide a simulation analysis to compare performance of both PCF and APCF the end of this work.

Chapter 3

Problem Statement, Proposed Solution and Methodology

The general context and scope of this work were given in the preceding chapters where descriptive statistical anomaly detection mechanisms were introduced and previous work including different anomaly detection approaches were described. The deficiencies of prior solutions as they relate to our problem space were also described. At the end of Chapter 2, an overview of Partial Completion Filters, as a remedy to the scalability problem was also presented. Particularly, the necessity of scalable DoS attacks detection without compromising precision was addressed.

In this chapter we elaborate our study and explain our approach to improve PCF methodology in order to take its false detection ratio under control, without sacrificing its simplicity. In Sections 3.1 and 3.2, the problem is clarified and the scope of this study is defined. Section 3.3 explains our improved methodology, APCF, and give the mathematical structure and analysis to relate the detector to the target network which it is intended to guard.

3.1 Scope and Assumptions

This research focuses on detecting partial completion DoS attacks at high link speeds (e.g. 1 Gb/s or more) with improved false detection ratio over its predecessors. To

this end, we overruled signature based intrusion detection approach because reassembly and dependency on known signatures are two of its constituents which prevent scalability. Stateful anomaly detection approach, similarly, is not a good candidate for scalability since it has to maintain state per each flow. We focus on stateless anomaly based IDS in this work that use aggregation approach introduced in [22] with PCF. The set of assumptions in our work is as follows:

- Intrusion signals are distributed randomly among the traffic, with exponential distribution. This helps us to model the detector behavior and obtain parameter analysis. Although this is not a 100% correct assumption, to our knowledge none of the anomaly detection approaches have postulated otherwise, given they presented any parametric analysis at all [6]. Ding et al. in [13] suggests that intruders tend to inject randomly generated flows for their attacks. Varghese in [14] and Moore et al. in [30] have postulated the same assumption for their studies.

- The detection mechanism sits on the routers, at egress or ingress points of a network.

- Certain statistical measures are available in the host network. These measures are used in our parametric analysis to adjust APCF for the host network. They include average number of incoming connection initiation packets, the probability of a particular bucket to contain intrusions, and the average number of packets aggregated to a particular group. We will explain these measures in detail in section Section 3.3.3.

- We aggregate connections into buckets by means of hashing functions, using certain extracted SIP header fields[1] such as *source IP address* and *destination*

[1] As mentioned before in Chapter 1, TCP header fields can be used as well as other easily extractable fields in any connection oriented protocol

IP address. We assume that these fields are already extracted by the router and are accessible.

- Collision is disregarded in this study. Hashing functions are assumed to be collision-free or hashing tables are large enough to avoid any collision. Since hashing attributes are not fixed, number of hashed buckets can be reduced to satisfy this assumption.

3.2 Problem Definition

Ever growing VoIP networks and ever increasing malicious intrusions over those networks have demanded intrusion detection mechanisms that investigate traffic more effectively [27, 25]. Necessary factors of this effectiveness encompass detecting attacks in their preliminary stage, investigating traffic in ingress and egress points of a network instead of endpoints, employability in high-speed routers, and having low false detection ratio. With these as axioms, we found that PCF [22] is a significant step towards a scalable IDS with false detection ratio under control. Its implementation is extremely easy for a hardware module and as a add-on to a router(it constitutes a counter and a comparator per each bucket, see Figure 2.3), and its aggregation approach allows the detector to scale up to the router speeds. Nevertheless, PCF suffers from drawbacks that are explained in following sections.

3.2.1 Error Estimation

We call a bucket as a "bad bucket" if its counter value falls outside a given bound[2]. Because PCF raises alarms for all flows hashed to a bad bucket, three types of error occur in PCF alarms: the *first type* occurs if all connections are *benign* in a bad bucket (false positive); the *second type* occurs when bad bucket containing intrusions is overlooked and ends up with a counter value that is less than the threshold (false negative); the *third type* occurs when a bad bucket *includes* both legitimate and attack

[2]There is no heuristic except experience in [22] to how to determine this threshold.

connections. This means that a bucket contains actual intrusions which caused the counter value to exceed the threshold, as well as it also contains benign connections. In the first two cases behavioral aliasing causes the counter value to exceed the threshold.

The probability that the first type of errors occur is smaller than $1 - 0.9987 = 0.0013$ if we set the bound to the counter value as $3\sigma_n$, as shown in Section 2.3.2. For the third type of errors, if we assume that there are b attacks and the total number of buckets is m, then the number of bad buckets should be h where $h \leq b$. Therefore, the number of legitimate connections mapped to a bad bucket by chance alone will be $\left(\frac{h}{m}\right) \cdot f$, that can be bounded by $\left(\frac{b}{m}\right) \cdot f$, where f is the total number of benign connections. The number of this type of errors is not negligible since f could be very large; The authors in [22] did not provide any experimental assessment for the second and third type of errors in terms of individual connections. Particularly for the false negative error, since $3\sigma_n$ is notably large (see Section 2.3.2), this type of error can not be ignored.

3.2.2 Adaptability and Parameter Heuristics

From Equation 2.1, it is obvious to note that the only influencing factor in PCF threshold is n, the number of valid messages hashed to a bucket. Hence, n is the only factor that can be leveraged to adjust PCF for a particular network. Besides, the threshold is very sensitive to variations of n; let us name PCF threshold as τ_{PCF}. With $\tau_{PCF} = 3\sigma_n$ and $\sigma_n = \sigma_b\sqrt{n}$ (Section 2.3.2), if n increases to 3000 from 1000 connection per bucket, τ_{PCF} will be increased by almost 69. Note that larger the PCF threshold, the bigger will be false negative and false positive error ratios. Therefore, PCF does not provide mechanisms to flexibly adjust itself to a network with certain attack probability, noise rate, etc.

The analysis in [22] gives the value for τ_{PCF} in a fully benign traffic; if there are intrusions, the analysis must be different (the reason is discussed and explained in

Section 3.3.3). The given threshold causes notable false positive and false negative errors, resulting in a low *efficiency*.

3.3 Solution and Methodology

PCFs help in detecting DDoS attacks scalably by aggregating connections into buckets and observing their behavior. This observation is aided by counters that keep track of the balance between call initiations and terminations in each bucket.

3.3.1 APCF Introduction

Our new detection method, the *Advanced PCF* (APCF) improves the detection efficiency by breaking the PCF counter into two different counters. The first one, *Behavior Alarming Counter* (BAC) counts a certain number of packets that arrive consecutively from the connections and are hashed to a certain bucket. It alarms a second counter called *APCF counter*, if the past stream of signals is found suspicious. APCF counter holds the number of alarms it has received from BAC. If the APCF counter value exceeds a preset threshold (τ_{APCF}), it recognizes the flows hashed to the bucket as intrusions. Figure 3.1 shows the general functional diagram of APCF. In this figure, connections are hashed into buckets for each of which there is an APCF counter. Dashed area shows the functionality of BAC along with APCF counter. Note that each bucket has its own APCF counter and BAC.

3.3.2 Behavior Alarming Counter (BAC)

BAC is a bin of n consecutive valid messages (n is called BAC *population*) and holds the balance between the connection initiation and connection termination signals. If the balance is more than a preset threshold τ_{BAC} (named BAC *alarm threshold* and stated as a percentage of n), BAC identifies that stream of packets as a suspicious one. BAC is a counter with the same idea as of PCF. It is incremented for every connection initiation and decremented for any connection termination packet. However, the difference between them is that BAC stops counting after every n packets, alarms

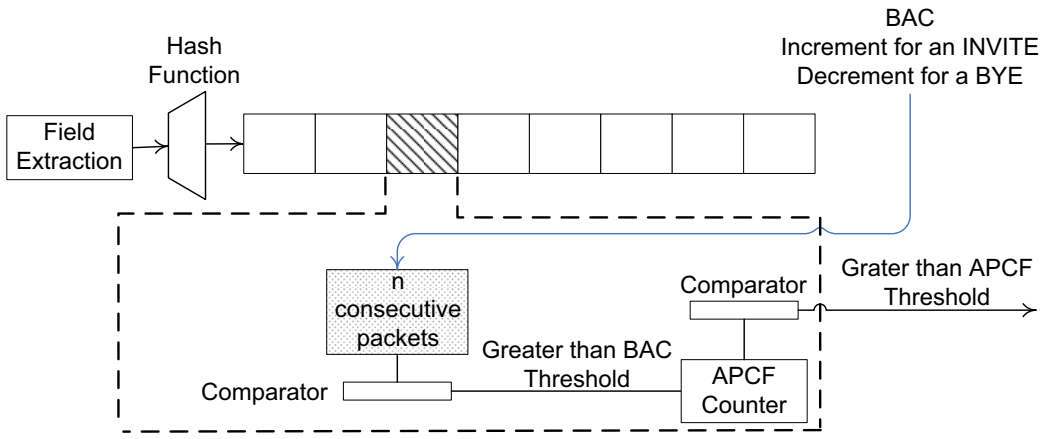


Figure 3.1: APCF Functional Diagram

APCF if BAC is more than the threshold, and resets itself to zero. This alarm causes APCF to increment its counter. A behavioral alarming counter with population of n and alarm threshold of τ_{BAC} is denoted as $BAC(n, \tau_{BAC})$. $BAC(n, \tau_{BAC})$ signals an alert to the APCF counter if the count exceeds $n.\tau_{BAC}$.

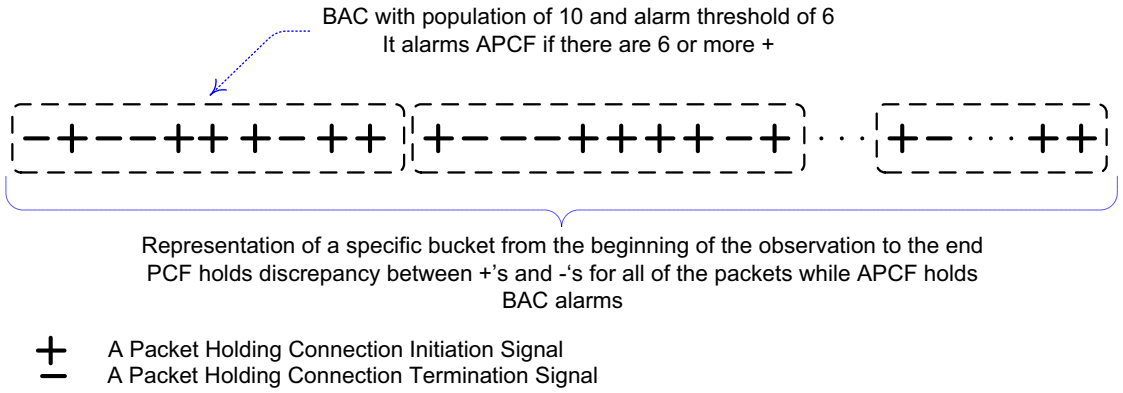


Figure 3.2: BAC Behavior

Figure 3.2 gives an insight into BAC behavior for a single bucket, assuming connection initiation packets are marked as "$+$" and connection terminations as "$-$".

Note that BAC resets its value to zero for the next window of n packets. If parameters are adjusted appropriately, the imbalance in the buckets will be certainly recognized in one or more packet groups. That is for any partial completion attack the attacker has to send groups of INVITE messages without proper ENDs following. As a reformulation, BAC is capable of detecting any imbalance between initiation messages and termination messages if its parameters are adjusted correctly. We investigate the parameter adjustment based on some given network features in the next section.

3.3.3 APCF Behavior Analysis

This section provides a theoretical analysis on the APCF behavior and a heuristic on how to adjust the parameters for a specific network configuration. First we discuss the probability for BAC to alarm an APCF in the absence of any attack traffic. In a benign traffic scenario, BAC value has a binomial distribution as BAC is updated with values of -1 and $+1$ with equal probability[3]. i.e. $P\left[BAC(n, \tau_{BAC}) = k\right] = \binom{n}{k} p^k q^{n-k}$. Therefore, BAC alarm probability with certain population size and alarm threshold (n and τ_{BAC}), **given that there is no intrusion**, can be calculated as,

$$P_{BAC} = P\left[BAC\left(n, \tau_{BAC}\right) \geq \alpha\right] = \sum_{k=\alpha}^{n} \binom{n}{k} \cdot \left(\frac{1}{2}\right)^n \tag{3.1}$$

where $\alpha = \lfloor n.\tau_{BAC} \rfloor$. APCF counter value is always updated with values of 0 and 1 and therefore, the APCF counter value follows a Binomial distribution. That is, APCF counter counts BAC alarms as 1 with success probability of P_{BAC} and no alarm as 0 with failure probability of $(1 - P_{BAC})$. This fact helps to bound the APCF counter for a non-attacked bucket, similar to PCF. As an example, according to Equation 3.1, $P[BAC(20, 0.75) \geq 15] = \binom{20}{15}0.5^{20} + \binom{20}{16}0.5^{20} + \cdots + \binom{20}{20}0.5^{20} =$

[3]In practice the number of initiation signals might be slightly larger than that of terminations because of the delay. This can be taken into consideration by changing the Bernoulli probability. For instance, the equal probability of 0.5 for both $+1$ and -1 can be shifted to 0.6 and 0.4 in favor of initiations

0.02069473. This means that $BAC(20, 0.75)$ alerts the APCF with relatively small probability of 0.0269473, given that it investigates every 20 packets for 15 or more invitations. This value is the success probability for the Binomial distribution[4] of the affiliated APCF counter value.

As the BAC threshold nears 1.0, the BAC alarm probability reduces and APCF becomes less sensitive. The reason is that, BAC alarms APCF when **larger** number of invite packets are received in a stream of n packets; therefore, APCF detects fewer buckets as intrusions. On the other hand, when the BAC threshold is smaller, the BAC alarm probability increases and sensitivity grows. Note that APCF counting units are not for individual connections, but for BAC alarms. In other words, if the total number of packets hashed to a bucket is C and BAC population is n, then the maximum possible value of APCF counter is $m = \frac{C}{n}$.

Similar to PCF, the Binomial distribution of APCF counter can be approximated by a Normal distribution with mean $\mu_{APCF} = P_{BAC} \cdot m$ and a standard deviation $\sigma_{APCF} = P_{BAC}(1 - P_{BAC})\sqrt{m}$. Substituting these values into Equation 2.1 yields a tighter alarming bound for APCF in a fully benign traffic scenario. For example consider a bucket with 3000 packets and a BAC population of $n = 20$. Thus, m would be 150, $P_{BAC} = P[BAC(20, 0.75) \geq 15] = 0.02069473$, $\mu_{APCF} = 3.10421$ and $\sigma_{APCF} = 0.24821$. Similar to Equation 2.1, it is easy to calculate the probability that APCF won't raise an alarm for a bucket will be $P[|X - \mu_{APCF}| \leq (3\sigma_{APCF})] = 0.9987$.

This result shows that if all the flows are not attacks, the APCF counter deviation will be less than $3\sigma_{APCF}$, or approximately 0.74463, with a probability of 0.9987. Hence, APCF has significantly smaller counter deviation than PCF (In this example, APCF counter lies between 0 and 4 instead of -164 and 164 in PCF). In fact, this bound depends on the BAC threshold and BAC population. How to find right values

[4]Note that for PCF this value always was 0.5

for these parameters is studied in the following section, 3.3.4.

3.3.4 APCF Parameter analysis

APCF introduces three controlling parameters as opposed to a single counter as in PCF. These parameters are APCF threshold, BAC population and BAC alarm threshold. Among them, the alarming probability of BAC (P_{BAC}) is derived from the population and alarm threshold. It means that if we have a proper value of P_{BAC}, we can find a relation between population and alarm threshold. Moreover, BAC alarm probability is the key variable to calculate a bound for APCF threshold as well as its mean and standard deviation in the non-attack situation.

Adjusting APCF parameters for a given network relies upon three given statistical attributes:

1. Average number of connection initiation packets or the *mean number of initiation packets* (M_{in}) in each bucket

2. Probability that APCF raises an alarm for a bucket or P_{ATTACK}, which can be considered as the probability that a bucket is an intrusion

3. Average number of packets in each bucket or C

If the traffic contains intrusions, then Equation 2.1 should be rewritten as:

$$P\left[a \leq \frac{X - P_{BAC} \cdot m}{P_{BAC}(1 - P_{BAC})\sqrt{m}} \leq b\right] = 1 - P_{ATTACK} \tag{3.2}$$

Consequently, if we relate P_{BAC} to some available network attributes, we can derive the relation between necessary parameters and adjust APCF accordingly. As mentioned previously in Section 3.1, we assume that all initiation and termination packets are scattered randomly in a stream of valid messages. If we assume that a group of initiation messages are together in an intrusion traffic, such attacks would be very easy to detect. Instead, the random assumption aims to target more intelligent

attacks that create signaling traffic with initiation and termination packets scattered randomly to evade detection. We point out that APCF will be more accurate if accurate models of signaling traffic are used. This traffic modeling is beyond the scope of this work.

Because BAC value is updated for every n packets, it can be approximated to a Poisson distribution(see Balls and Bins problem [28]) with $\mu_p = \frac{M_{in}}{m}$, where M_{in} is the average number of initiation packets received. Here we refer the BAC alarm probability as P'_{BAC} for this Poisson distributed BAC, to distinguish it from the Binomial distributed one. Subsequently, Equation 3.3 (Poisson Cumulative Distribution Function) gives the value of P'_{BAC} based on μ_p, and two BAC parameters (population and threshold).

$$\begin{aligned} P'_{BAC} &= \frac{\Gamma(n+1,\mu_p)}{n!} - \frac{\Gamma(\lfloor n \cdot \tau_{BAC} \rfloor + 1, \mu_p)}{\lfloor n \cdot \tau_{BAC} \rfloor !} \\ &= e^{-\frac{M_{in}}{m}} \sum_{k=\lfloor n \cdot \tau_{BAC} \rfloor}^{n} \frac{\left(\frac{M_{in}}{m}\right)^k}{k!} \end{aligned} \tag{3.3}$$

Assuming a and b in Equation 3.2 are symmetric, we can directly reformulate Equation 3.2, according to the Normal Distribution Function, as Equation 3.4:

$$P\left[b \leq \frac{X - P'_{BAC} \cdot m}{P'_{BAC}(1 - P'_{BAC})\sqrt{m}}\right] = \frac{P_{ATTACK}}{2} \tag{3.4}$$

We can find b by looking at the statistical tables for Z-ratios (*e.g.,* in [24]) and $P[\tau_{APCF} \leq X] = P_{ATTACK}/2$. Remember that X is the final value of the APCF counter, and P_{ATTACK} is a given control value. By substituting τ_{APCF} in Equation 3.4, we get:

$$b = \frac{\tau_{APCF} - P'_{BAC} \cdot m}{P'_{BAC}(1 - P'_{BAC})\sqrt{m}} \tag{3.5}$$

We know that P'_{BAC} can be derived from n and τ_{BAC}, and also $m = \frac{C}{n}$. As b is known from the table, Equation 3.5 leaves us with three unknown variables, τ_{APCF}, τ_{BAC} and n. However, the system should be reliable if there is no intrusion. Hence, τ_{APCF} must be a threshold that in no-intrusion case, no APCF alarm is raised. Therefore, τ_{APCF} can be bounded by Equation 3.2 with probability of 0.98 and $b = -a = 3$. This results in Equation 3.6 which leads to a relation between BAC population, BAC threshold and APCF threshold in a benign traffic scenario.

$$3 = \frac{\tau_{APCF} - P_{BAC} \cdot m}{P_{BAC}(1 - P_{BAC})\sqrt{m}} \tag{3.6}$$

We have to relate this result to the scenario in which intrusions are present. The key idea is that τ_{APCF} must be the same for both attacked and attack-free traffic. By substituting P_{BAC} and P'_{BAC} from Equations 3.1 and 3.3 into Equations 3.5 and 3.6, we come up with two equations that relate three APCF parameters with the affiliated network. The above analysis serves as a guideline heuristic to select APCF parameters for a given network.

Next chapter introduces some more error measures and analyzes the proposed method in a network setting using simulations.

Chapter 4

Analysis and Simulations

The simulation designed for this work aims to measure the performance of our proposed stateless IDS, APCF. We study the effectiveness of our analysis introduced in Section 3.3.3 to determine various parameters, as well as APCF efficiency compared to PCF. Section 4.1 describes the simulation platform, the network topologies and scenarios base on which we performed the simulation, as well as measurements used to assess PCF and APCF performance. Section 4.2 presents the simulation results and data we obtained along with analogies between PCF and APCF in terms of performance.

4.1 Simulation Platform and Architecture

Our simulation software is OMNET++, version 3.3 [4]. OMNeT++ is a public-source, component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. Its primary application area is the simulation of communication networks and because of its generic and flexible architecture, it has been successfully used in other areas like the simulation of IT systems, queuing networks, hardware architectures and business processes as well [4].

To accommodate OMNET++ simulation mechanisms with network standards and protocols, OMNET++ community has developed INET (Internet) framework.

INET is suited for simulations of wired, wireless and ad-hoc networks. It implements and supports many important network protocols such as IP, UDP/TCP, Ethernet, PPP, OSPF, RSVP-TE signaling, and 802.11. Basically, INET framework uses OMNET++ simulations concepts (such as queuing, timing, event handling, etc.) and implements protocol dependent features such as packet structures, signaling, routing, etc., for generic simulation of various Internet models.

We use INET to build small scale Internet like networks. Technical details about our simulated networks and platform are presented in Sections 4.1.2 and 4.1.3. However, before elaborating technical details of the network and scenarios, we introduce measurement factors in Section 4.1.1 that are used in this work to assess PCF and APCF functionality.

4.1.1 Measurement parameters

We used general classification terms for categorizing attacks in this work [15]. By the term *positive*, we refer to those single connections (<SIP, SP, DIP, DP>) detected as intrusions. *Negative* means a connection not detected as an attack. *False positives* (FP) are those innocent single connections that have been detected as intrusions and *true positives* (TP) are connections correctly detected as intrusions. APCFs detect a connection to be an attack by testing any incoming packet against the APCF counter of the bucket it is mapping to and declare it as an attack if the counter exceeds the APCF threshold.

The tool we used to visualize the APCF efficiency is Relative Operating Characteristics Diagram or ROC diagram [15]. ROC diagram is a good tool to obtain the behavior of a classifier when a specific classification parameter varies from a minimum value to a maximum value. Given that detectors are classifiers in nature, ROC curve shows how a particular detector behaves in terms of the ratio of true detections over false detections, when a specific feature (e.g. APCF threshold) varies. The larger the area beneath the curve, the better balance the detector holds between false positives

and false negatives [14]. Here are some definitions for the measures:

$$TP\ Rate(recall) = \frac{Truly\ detected\ attacks}{All\ the\ attacks} \tag{4.1}$$

Equation 4.1 defines *True Positive (TP) Rate*. It is also called *recall*, *hit ratio*, or *sensitivity* and determines what portion of intrusions has been detected.

$$FP\ Rate = \frac{Those\ connections\ falsely\ detected\ as\ attack}{All\ benign\ connections} \tag{4.2}$$

Equation 4.2 defines *False Positive (FP) Rate*. It determines what portion of benign connections have been falsely detected as intrusion. The two measures above are used to depict the ROC diagram shown in Figures 4.2 and 4.5.

Although two factors of true positive and false positive rates describe the detector behavior, they cannot be used to determine efficiency. The trade off between two measures is unknown, and they should be expressed in a unified inclusive factor that can be used as a measure to determine the best behavior of APCF. Furthermore, the important element of false negative error is not included in these factors. For these reasons, we use *precision* along with *recall* as a measure of *efficiency*. Precision is the portion of positively detected connections that are true intrusions. *Precision* is defined as [15]:

$$precision = \frac{truly\ detected\ attacks}{all\ detected\ (positives)} \tag{4.3}$$

In a real network scenario, it is very difficult (if not impossible) to measure the truly detected attacks. The reason is there is no absolute way to discover attacks amongst the traffic with 100% certanity. However, in our simulation, we can keep track of the generated true intrusions and use that count as a measure. In Kompella *et.al.*, [22], this measure is obtained by using a stateful detection system besides the

PCF detector.

Efficiency is a balance between the *precision* and *recall*. Efficiency (F_α or weighted F-measure) is a weighted measure which gives a different priority to precision and recall, using a *weight factor* α. It is defined as [15]:

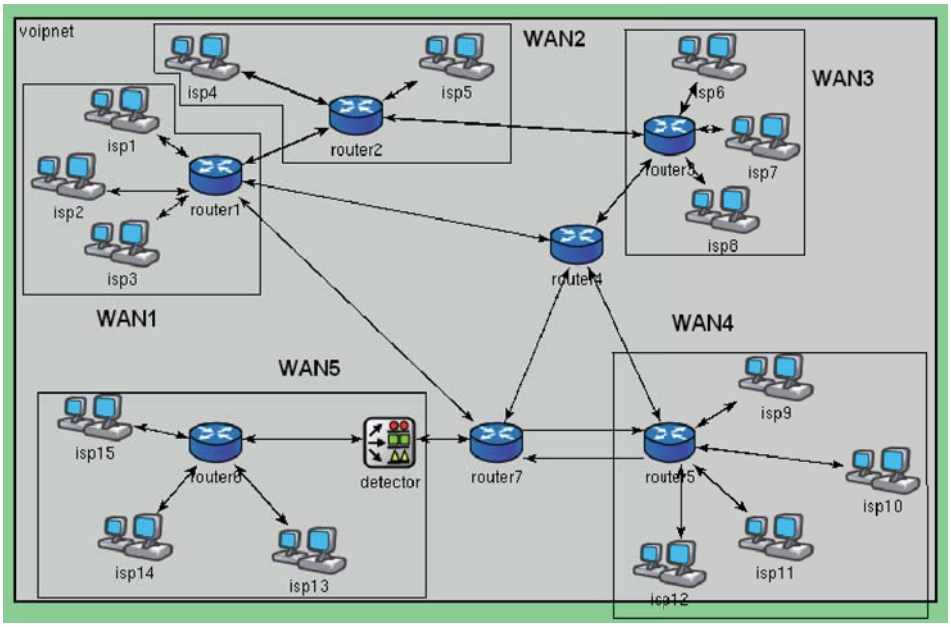
$$\text{Efficiency} = F_\alpha = \frac{(1+\alpha)\cdot(precision \times recall)}{\alpha \cdot precision + recall} \tag{4.4}$$

In our simulations, we used $\alpha = 1$ which is the *harmonic mean* of precision and recall, and gives the same weight to both of them. However, other F-measures can be used according to different false positive vs. false negative costs in various networks. For instance if it is critical for a network to serve legitimate users by any means and can overlook a few possible intrusions, precision must be set higher than recall. In this case α is more than 1. On the other hand, if it is vital to detect as many intrusions as possible and the network tolerates loosing some clients, recall is more important than precision; hence, α will be less than one to accentuate recall. The desired value of α, therefore, depends on the actual network requirements.

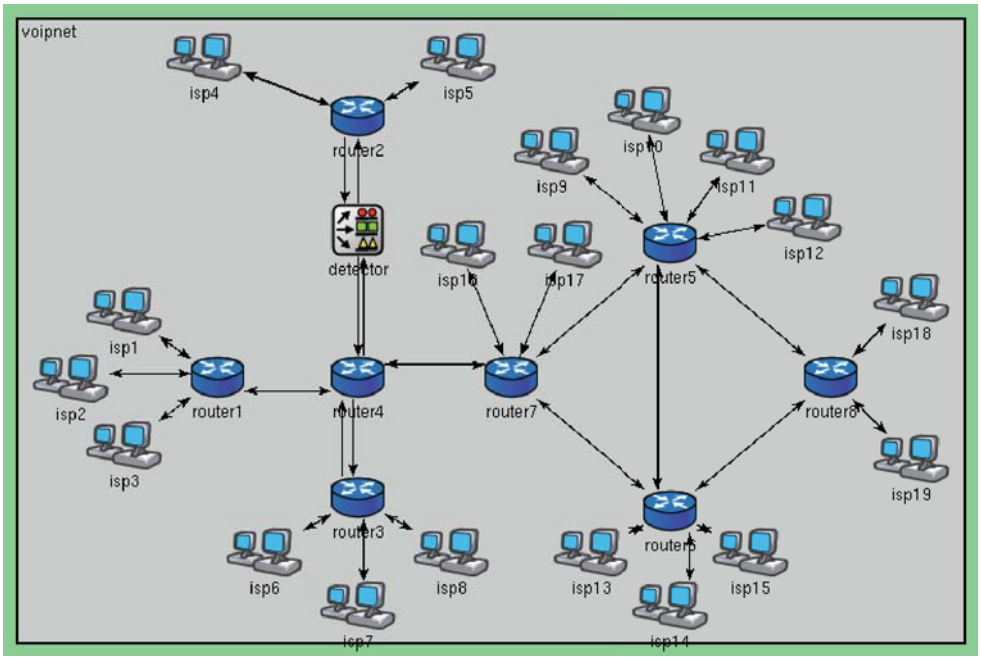
4.1.2 Network

Although there is no formal dependency between APCF functionality and the network topology, we developed two networks (both random networks) to test APCF and to make sure that there is no influence of topology on APCF even regarding their influence on the traffic. This is particularly true in our case due to the random distribution assumption we made that all invitation signals including intrusions are uniformly distributed(see Section 3.1). Figure 4.1 shows two topologies Network 1 and Network 2 that we chose for our simulations.

Since all technical features in both networks are identical except network topologies, hence forth we refer to both of them as the **network** unless otherwise explicitly mentioned. The network is divided into service providers (ISP) and routers. On



(a) Network 1: First network topology



(b) Network 2: Second network topology

Figure 4.1: Simulated network random topologies; two random networks

average each ISP is provisioned with 3000 clients. In other words, Network 1 connects 45000 peers through 7 routers, while Network 2 connects 57000 client through 8 routers. There are at least 200 active clients per ISP per second on the average in Network 1, and 350 in Network 2. Data streaming (RTP packets) is ignored to reduce the simulation time; SIP packets, OSPF packets and routing control packets constitute the main traffic. A manually set packet dropping rate is imposed on each network to compensate for network congestion due to the lack of RTP data stream.

The structure and configuration of each ISP is not taken into account. We assume them as LANs containing clients who are connected through possible routers and servers. Each ISP can contain smaller LANs or other ISPs. We are only interested in their inter-communication behavior, which reflects in the output traffic towards the edge routers. For instance in Figure 4.1, router1 is an edge router connecting WAN1 to other parts of the network by forwarding the traffic from WAN1 to the network and vice versa. Table 4.1 describes ISP attributes that are used in our simulations.

Table 4.1: ISP Attributes

Attribute	Description
num-client-to-call	The number of clients selected to initiate calls in every pickup period. For example if *num-client-to-call* is 50 and pickup period is 5 sec., for every 5 seconds, 50 clients will be selected randomly who initiate calls to random destinations.
call-pickup-time	Pickup period, determined randomly.
call-estbl-time	Each client, after being selected to initiate a call, waits for this amount of time to establish the connections. This is to eliminate synchronized bursts of calls.
call-time	The length of the call for each client, determined randomly.
intrusion-prb	The probability that any client in this particular ISP is an intruder, determined manually.
noise	The packet loss ratio, cumulatively, up to the edge router.

Each client in an ISP selects randomly one client who is in the same domain or

in a different one and establishes a connection for a call period which is also selected randomly from an exponential distribution. Each ISP has been assigned an attack probability with which they launch partial completion attacks on specified ISPs. Thereby we control the attack probability P_{ATTACK} mentioned in Section 3.3.4. The whole network is exposed to a drop/retransmission mechanism which is set randomly with a controllable rate, as mentioned above. Any user who is set to be an attacker is known and so we can keep track true intrusions to measure false and true positive ratios.

All routers use Open Shortest Path First (OSPF) algorithm for routing purposes. Since the OSPF routers provided by INET are virtually capable of handling any input rate, we added a random drop policy to the network manually. Based on this approach, any packet would be dropped randomly using an Exponential Distribution Function. The mean for generating this random number is manually set to 10% for high-attack scneario and 1% to low-attack scenraio.

The detector module contains both PCF and APCF functionality so that both methods can be compared in the same simulation setting. In Network 1, they guard WAN 5, containing ISP 13, ISP 14 and ISP 15. Any incoming and outgoing messages, as well as messages to a client in the same network will pass through router6, the edge router of WAN5. The separation between router6 and the detector module is for the simulation purpose, to maintain the software control over the detector as INET would not allow to use add-on module to the router. Indeed, the detector acts as a plug-in module to a router in reality and uses the already extracted header fields in the routing table. For Network 2, the detectors guard ISP 4 and ISP 5, both using another tier router as connecting means to Internet. Table 4.2 describes the detector module attributes.

All call related random variables are chosen from exponential distribution function. OMNET++ uses a Network Topology Description language called NED to

Table 4.2: Detector Attributes

Attribute	Description
n-BAC	BAC population (Section 3.3.2). Determined manually according to the analysis in Section 3.3.4.
p-BAC	BAC alarm probability (Section 3.3.2). Determined manually according to the analysis in Section 3.3.4.
APCF-thr	APCF threshold (Section 3.3)
PCF-thr	PCF threshold (Section 2.3)

define text based descriptions for the modules and use C++ objects to derive the functionality of each object.

4.1.3 Simulation Scenarios

It is worthwhile to emphasize that the detection ratios presented in this work are based on single clients and not on ISPs. These detection ratios are not provided for PCF in Kompella et. al. [22] and their evaluation was based on false positive and negative number of buckets.

Two scenarios are simulated for each network topology: Scenario 1 is a high attack scenario with attacks launched by more than half of the selected clients in the network; it means that *intrusion-prb* (see Table 4.1) for all ISPs is set to 50% on average. The error rate (*noise* in Table 4.1) is set to 10 percent for all WANs which is considerably a higher error rate[1]. As mentioned before, network error rate is mostly caused by the packet drops in router queues.

Scenario 2 is a low attack scenario, in which the *intrusion-prb* is set to 20% on average for less than half of the ISPs and the rest of the network generates regular traffic with a low error rate of 1 percent. In this scenario, half of the ISPs have *intrusion-prb* of 0 and the rest have 20% attack probability. Details about settings of these two cases is shown in Table 4.3. In both scenarios APCF aggregation is

[1]APCF parameters can be adjusted individually for each bucket to get a best detection ratio. Without loosing generality, we assigned the same intrusion probability and noise ratio to every bucket for simplicity in this work.

based on **source ISP IP address** and **destination ISP IP address** hashing, and PCF aggregation is two level hashing based on **source ISP IP address** for the first level and **destination ISP IP address** for the second level. We do not use ports as hashing arguments since SIP always uses port 5060 for signaling.

	P_{ATTACK}	Pkt drop	APCF Hash	PCF Hash
Scenario 1	50%	0.1	<SRC IP, DEST IP>	<SRC IP>,<DEST IP>
Scenario 2	20%	0.01	<SRC IP, DEST IP>	<SRC IP>,<DEST IP>

Table 4.3: Two simulated scenarios

4.1.4 Simulation Runs

Both network topologies were tested against two simulation scenarios introduced in previous section. For the first network(i.e., Network 1), Figure 4.1(a), all simulation runs are repeated 8 times with different seeds. Due to time constraints, simulation runs for the second network (i.e, Network 2), Figure 4.1(b) are repeated with 4 runs for each simualation with different seeds. However, we applied *t-test* [7] on all sets of data for 8 seeds and 4 seeds and found out that for each data set the difference among data for different seeds is not statistically significant for 95% confidence interval.

To produce ROC and efficiency diagrams, APCF threshold is varied from 300 down to 0. As a reformulation, the difference between two consecutive points in the ROC diagram is just in their APCF threshold. This process is repeated for various BAC thresholds, 60%, 70%, 80% and 90%. Experience showed us that there is no significant discrepancy between thresholds in the middle, say 70%, 75%, 77% etc. Thus, we eliminated those to avoid confusion in the diagram. As we increase the APCF threshols, both false and true postive ratios decrease monotonically until they become zero. After that as APCF threshold continues to increase these ratios remain at zero. Therefore we eliminated all the zero outcomes after the first one for each BAC threshold. The same process holds for PCF. We vary PCF threshold from 1000 down to 0 and assessed its false positive, false negative and efficiency per network

topology per scenario.

Since BAC population is the free variable for which there is no analytical assessment, we repeated the simulations with $n = 15$, and $n = 35$. We aimed to show that our parameter analysis stands sound with different BAC population, and that even with different BAC population the efficiency in the best case is better than that of PCF. Best case is obtained by calculations we have provided in Section 3.3.4.

4.2 Simulation Results

4.2.1 Parameter Calculation

This section gives the calculated APCF parameters, based on the analysis given in Section 3.3.4. Table 4.4 presents the theoretical value for APCF that should result in the acceptable outcome for the particular simulated network. Note that these calculations are all based on $n = 25$ for the BAC population.

	P_{ATTACK} in average	M_{in} or Mean # of initiation pkts	C or Average # of pkts in each bucket	τ_{BAC}	τ_{APCF}
Network 1 Scenario 1	50%	22091	42840	20	14
Network 1 Scenario 2	20%	17537	29675	18	38
Network 2 Scenario 1	50%	37920	79175	20	8
Network 2 Scenario 2	20%	27336	51594	19	15

Table 4.4: Calculated parameters for APCF for simulated networks, based on parametric analysis

Care should be taken when calculating these parameters for APCF. τ_{BAC} should not be less than 50% because we are not interested in *lack of* INVITE messages, but their excess. With $\tau_{BAC} = 100\%$ both P_{BAC} and P'_{BAC} in Equations 3.6 and 3.5 become 0, making the fraction infinity. This means setting full (100%) BAC threshold alarms the APCF counter whenever all 25 incoming messages are INVITE, which is

not an appropriate assumption to find intrusions. With full BAC threshold APCF fails to detect majority of intrusions, causing a significant false negative error. For these reasons we calculated both Equations 3.6 and 3.5 for the BAC thresholds from 1 to 25 and calculated the difference between τ_{APCF} . The smallest difference between 1 and 25 excluding the two end points is presented in Table 4.4.

Table 4.5 gives the threshold calculations for PCF. According to Section 2.3.2, PCF threshold depends only on C, the average valid messages hashed to each bucket. The τ_{PCF} shows the threshold which the PCF counter will not exceed with a probability of 0.9987 for a traffic scenario without any intrusion.

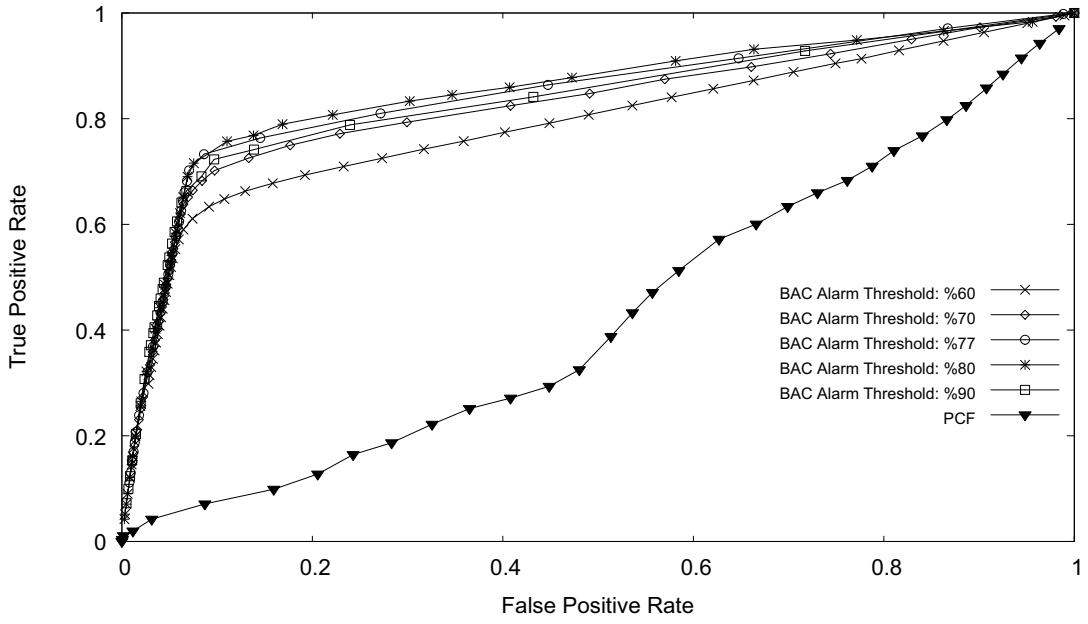
	C or Average # of packets in each bucket	τ_{PCF}
Network 1 Scenario 1	42840	620
Network 1 Scenario 2	29675	516
Network 2 Scenario 2	79175	844
Network 2 Scenario 2	51594	681

Table 4.5: Calculated PCF threshold for simulated networks

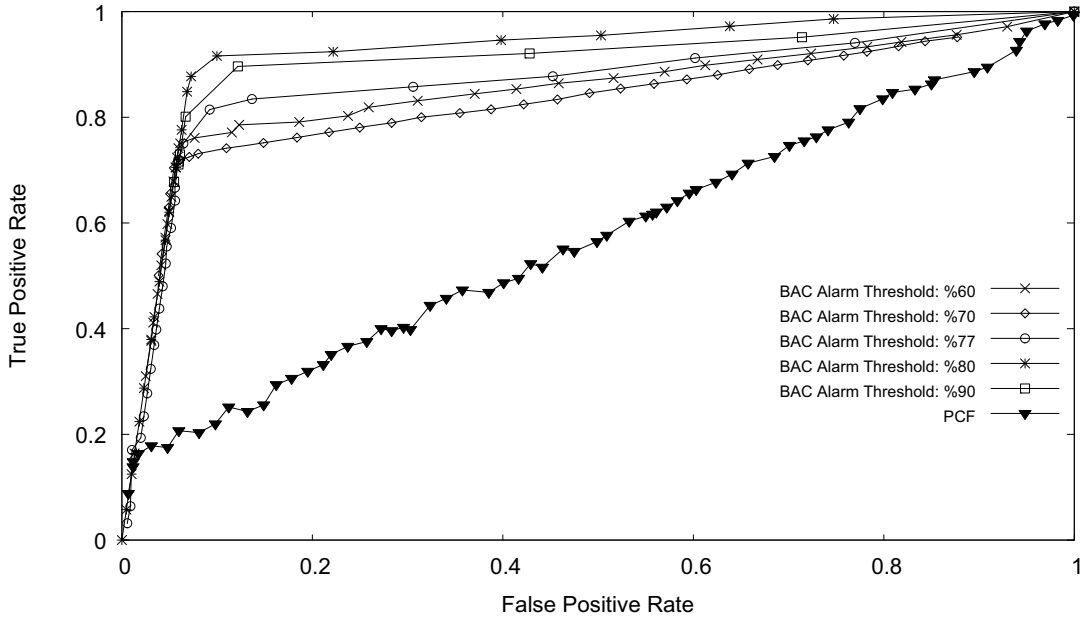
4.2.2 Simulation Results

First Scenario, High Attack Density

Figure 4.2 demonstrates how APCF and PCF detection ratios increase as their thresholds increase. Different BAC thresholds have been tested to assess the validity of our parameter analysis. Figure 4.2(a) represents APCF and PCF behaviors in Network 1 (Figure 4.1(a)) in massive attack scenario (Scenario1). Figure 4.2(b) shows APCF and PCF behaviors in the Network 2 (Figure 4.1(b)) in the same scenario.



(a) Network 1



(b) Network 2

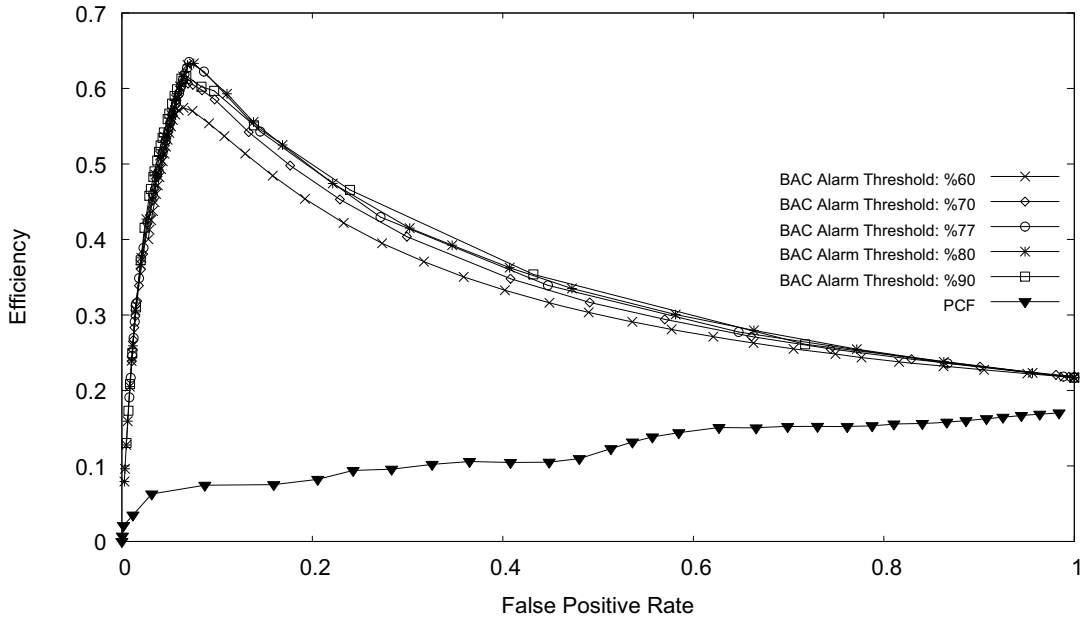
Figure 4.2: ROC diagrams for different BAC alarm thresholds, high attack scenario

When APCF and PCF thresholds are large enough[2](near the origin) both false positive and true positive ratios are almost zero; this is because the counter can not exceed these larger thresholds and prevents incoming connections to be declared as intrusion. As the threshold decreases, both ratios grow until they tend to one because BAC alarms now reach the APCF threshold and more incoming packets are declared as intrusion. Remember that true positive detection rate and false positive detection rate are *monotonically increasing* functions; when the threshold is smaller, more incoming connections are declared as intrusion and essentially more intrusions along with more innocent connections are marked as attacks.

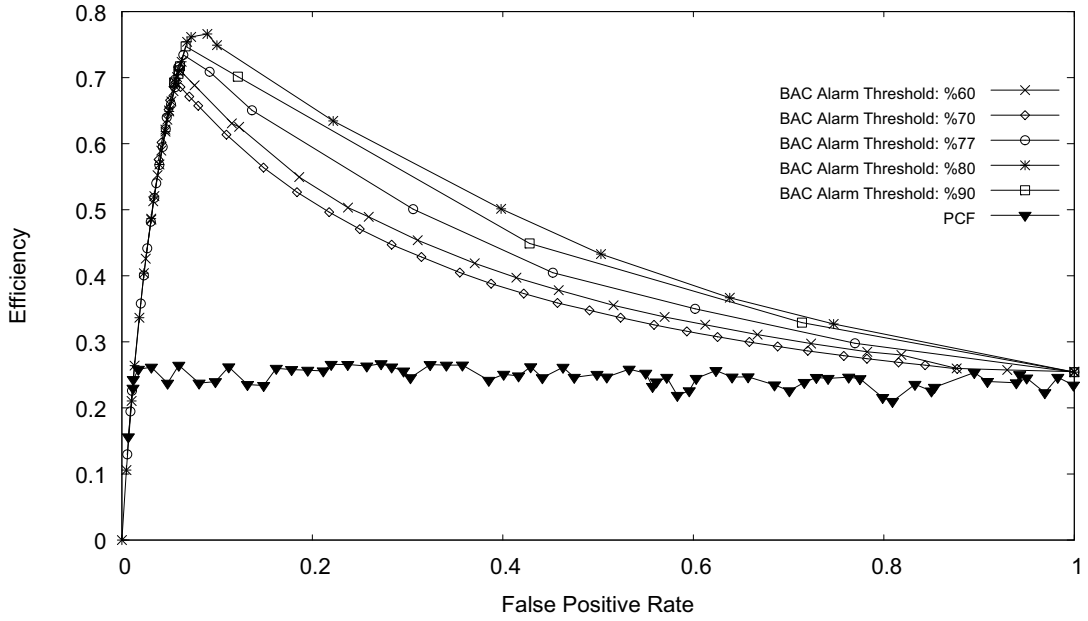
As it is observable, higher BAC thresholds produce sparser ROC diagrams. This is because of higher τ_{BAC}s having smaller range of τ_{APCF} variation. For example for $\tau_{BAC} = 0.9$, true positive and false positive rates drop to zero for $\tau_{APCF} = 40$, whereas for $\tau_{BAC} = 0.6$ these rate do not tend to zero unless τ_{APCF} gets close to 300.

Figure 4.3 depicts APCF and PCF behaviors in both networks in the high attack scenario in terms of efficiency. The diagram is produced similar to ROC diagram, but with *y-axis* to be efficiency instead of true positive rate. Each simulation run (each point in the ROC diagram) has a unique false positive ratio.

[2]The largest τ_{APCF} for all P_{BAC}s is 300, and for PCF is 1000 in our simulation



(a) Network 1



(b) Network 2

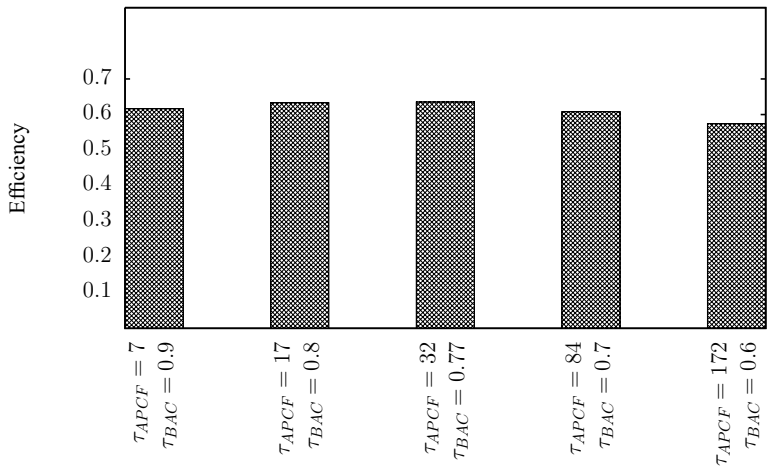
Figure 4.3: Efficiency diagrams for different BAC alarm thresholds, high attack scenario

According to the ROC diagram, APCF outperforms PCF regardless of the value of BAC alarm threshold. APCF behaves differently than PCF at the beginning of the diagram, when τ_{APCF} is larger. True positive ratio (recall) acceleration is more than that of false positive ratio in APCF, leading to higher efficiencies as efficiency approaches its peak in Figure 4.3. This is the point for which the best balance between true positives and false positives has been reached, and the efficiency is the largest.

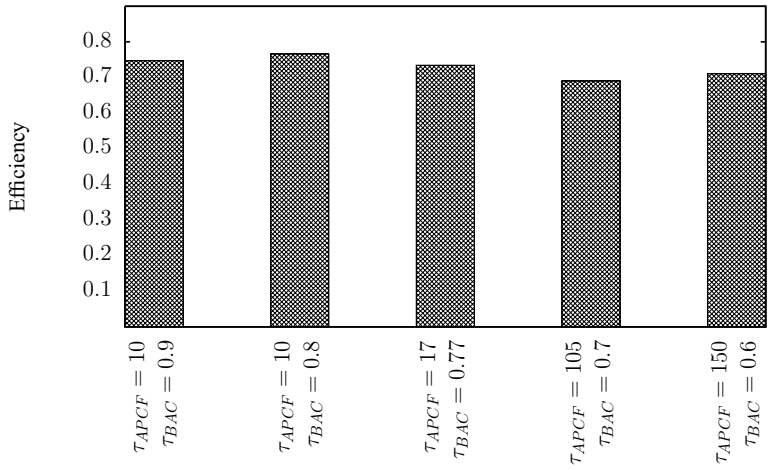
We explain this behavior as follows. Since BAC is sensitive to abnormal behavior of streams (i.e., more initiation signals than BAC thresholds for a given BAC population), it tolerates the behavioral aliasing caused by packet drops or retransmission or even small customer bursts. The translation of these confounding events into BAC alarms smoothen APCF counter behavior. On the other hand, attacks are translated into consecutive numerous BAC alarms due to their aggressive behavior. They have to come together, and maintain their integrity for a longer period. These two intrinsic features lead APCF to be more sensitive to intrusion than behavioral aliasing.

Both Figures 4.2 and 4.3 suggest that, in both networks, BAC alarm threshold variation does not impose a significant discrepancy in APCF results in this scenario. Nevertheless, results from simulations confirm our calculations for parameters in Table 4.4. The calculated BAC alarm threshold for Network 1, first scenario is 20; with BAC population of 25, the BAC alarm threshold is 80%. As Figure 4.3(a) suggests, $\tau_{BAC} = 77\%$ has the highest efficiency along with $\tau_{BAC} = 80\%$. For the second network topolgy (Network 2), 80% for BAC threshold is the best choice according to the simulation; Table 4.4 gives us the same amount.

Figure 4.4 demonstrates best efficiencies for each BAC threshold and affiliated APCF threshold. The straight line stands for PCF efficiency for each network. In both networks all efficiencies exceed PCF efficiency.



(a) Network 1



(b) Network 2

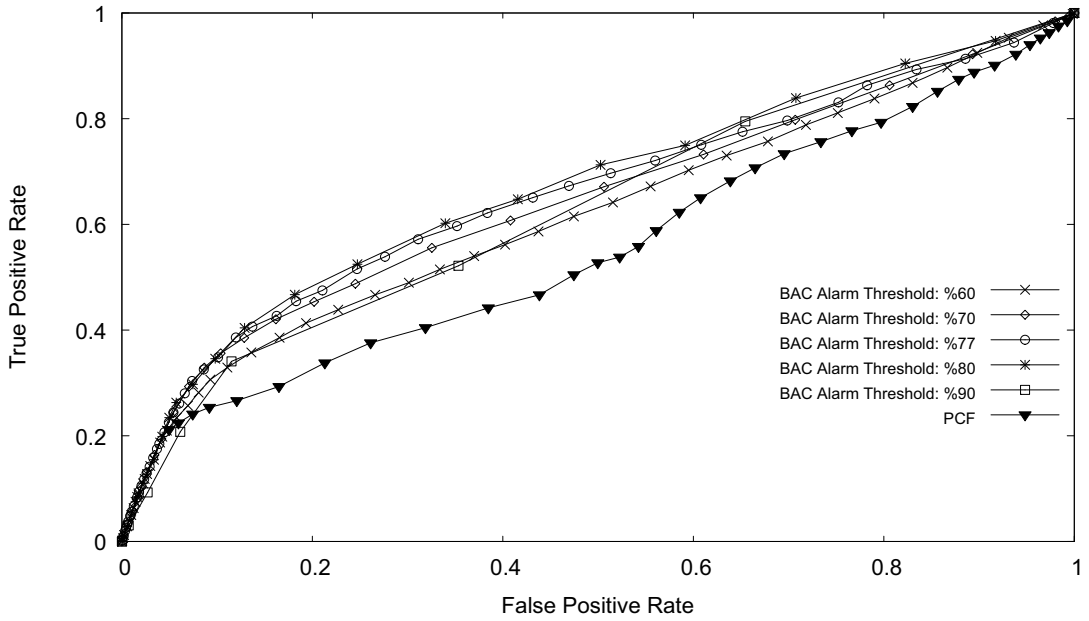
Figure 4.4: Efficiency vs. BAC and APCF thresholds, high attack scenario. The efficiency for PCF for the first network is 0.17223 and for the second network is 0.26694, both cases lower than APCF efficiencies

It is immediate to realize that best efficiencies are not notably different for various BAC alarm thresholds; Diagrams 4.4(a) and 4.4(b) display this very clearly. But still it is important to know which τ_{BAC} to choose for a particular network to obtain the corresponding P_{BAC} and essentially to calculate τ_{APCF}; because as Figure 4.3 and 4.2 show, the result is dramatically different for each τ_{APCF}. Our parametric analysis match the simulation result with an acceptable accuracy. BAC alarming threshold is suggested correctly, and APCF threshold is within 5 percent accuracy in the worst case.

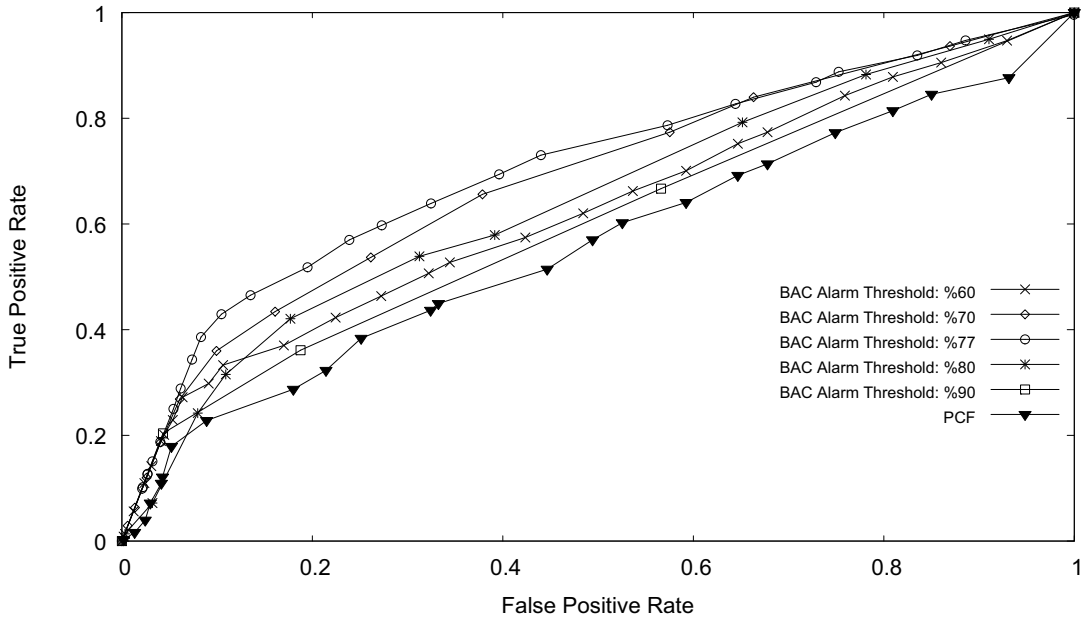
High intrusion density is rather easy for APCF to detect. High P_{ATTACK} causes small difference between different τ_{BAC}. However, low attack density is more challenging. We present the simulation results and analysis for the second scenario with low attack density in the following part.

Second Scenario, Low Attack Density

Figure 4.5 demonstrates ROC diagrams for APCF and PCF in both networks, for the second scenario. Second scenario is a network in which half of ISPs do not contain any intrusions, and the rest have no more than 20% of their traffic as DoS attacks. This situation is challenging in essence; attacks are not sharp in terms of traffic attributes, but still exist and are potentially harmful.



(a) First network



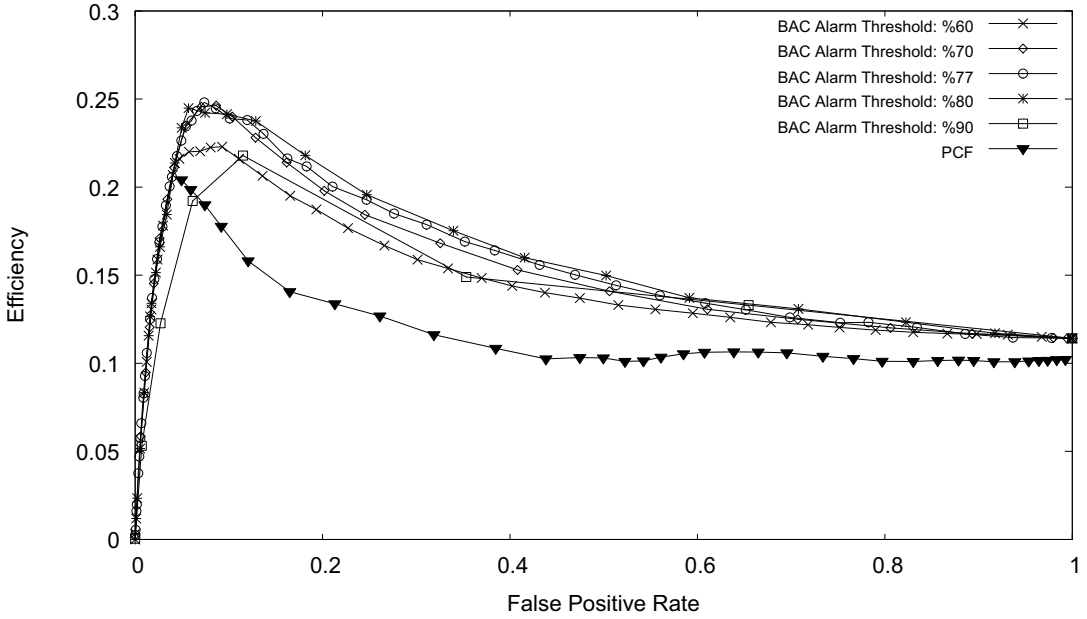
(b) Second network

Figure 4.5: ROC diagrams for different BAC alarm thresholds, low attack scenario

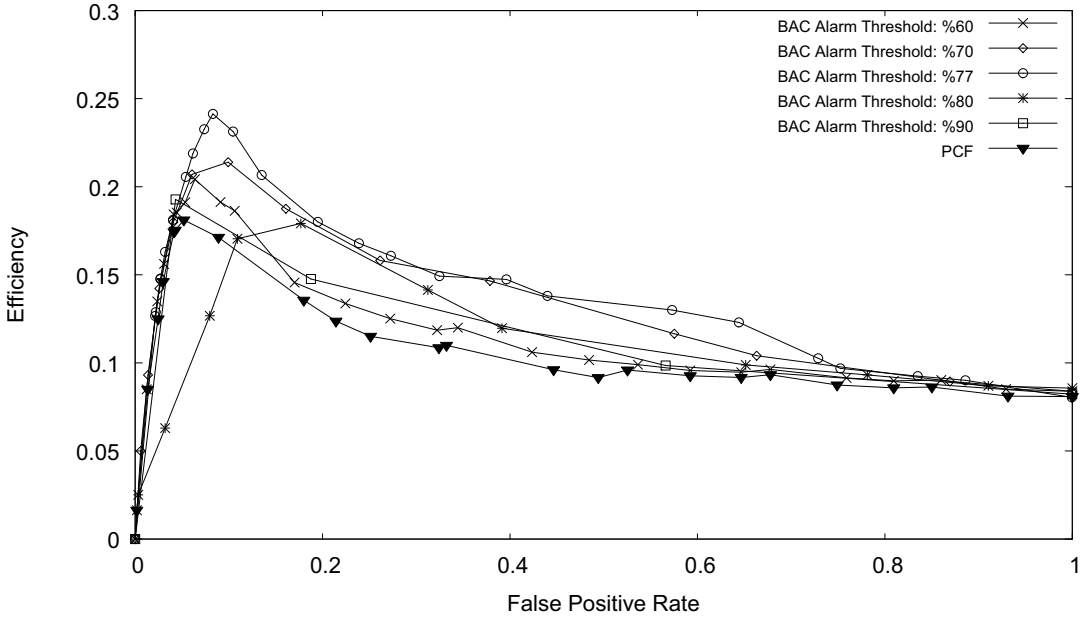
In this scenario, both PCF and APCF are rather low in efficiency; they have to sacrifice either the precision or recall due to intense blend between intrusion traffic and innocent traffic. The reason is that attacks are spread randomly throughout the network, including the WAN our detector intends to protect.

Figure 4.6 depicts efficiency diagrams for PCF and APCF in the second scenario. Figures 4.5(a) and 4.6(a) suggest that the best efficiency is attained with τ_{BAC} to be 77% or 80% for Network 1, the first network topology. The analysis suggests 18 for BAC alarming threshold in this case, which is $\tau_{BAC} = 72\%$.

PCF behavior is rather linear in the low attack scenario. The reason lies in the growing rate for true positive and false positive ratios. As Figure 4.5 presents, although true positive rate is always ahead of false positive, they grow with almost identical pace.



(a) Network 1

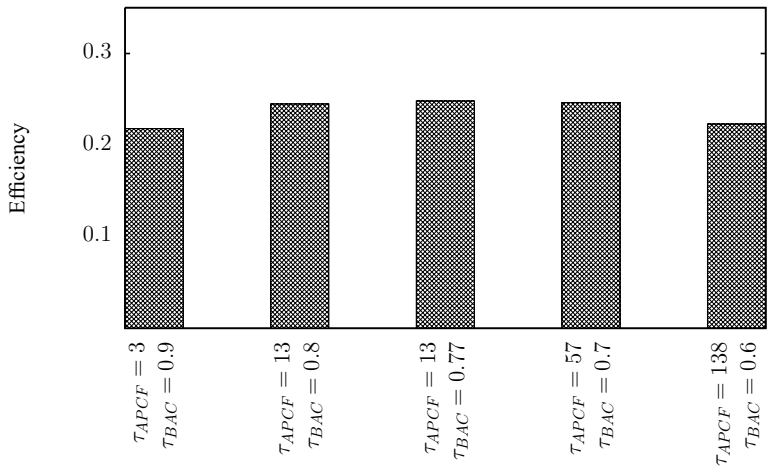


(b) Network 2

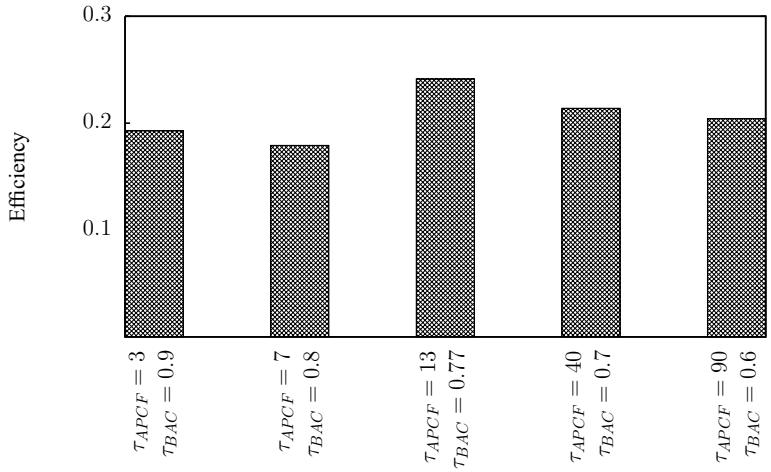
Figure 4.6: Efficiency diagrams for different BAC alarm thresholds, low attack scenario

We interpret this PCF linear behavior as follows. Since the attack ratio is low, mere counting of messages causes the growth of false positive ratio to increase drastically. As PCF threshold is incremented, the number of discovered bad buckets which were unknown before are almost the same as the number of benign buckets which are falsely categorized as an attack. Behavioral alarming system which is used in BAC results in finding *groups* of messages and their *occuring pattern*, rather than merely counting signals. This distinction between attacks and innocent traffic leads to a more effective detector, specially for massive throughput routers that switch millions of packets.

To test the parametric analysis (Section 3.3.4) for this second scenario, we provide the following diagram similar to the previous section, 4.2.2. Figure 4.7, analogous to Figure 4.4, depicts the best APCF thresholds for different τ_{BAC}s.



(a) Network 1



(b) Network 2

Figure 4.7: Efficiency vs. BAC and APCF thresholds, low attack scenario. The efficiency for PCF for the first network is 0.20412 and for the second network is 0.18113, both cases lower than APCF efficiencies

4.2.3 BAC Population Variation

In the previous section we presented our simulation results to show APCF's better performance than its predecessor, PCF. Also we showed that the theoretical analysis can be used as heuristics to adjust the detector for a particular network. Albeit, we did not discuss the BAC population (referd to as n) which was left out of our analysis as a free variable.

In Section 4.2.2 no specific assumption has been imposed to the simulation. $n = 25$ for the BAC population was an intuitive good value for a group of messages to be judged. Table 4.6 shows that even with different BAC population, the parameters can be estimated with our parametric analysis with appropriate precision. Data for this table is obtained from Network 1 and first scenario environment settings. Note that for Network 1, scenario 1, PCF best efficiency is constant and equals to 0.17223.

	Data from simulation				Predicted	Efficiency
	Best Efficiency	FP rate	TP rate	τ_{APCF}	τ_{APCF}	
$\tau_{BAC} = 60\%$						
$n = 15$	0.49382	0.06075	0.48882	475	N/A	N/A
$n = 35$	0.51651	0.07437	0.52277	395	N/A	N/A
$\tau_{BAC} = 70\%$						
$n = 15$	0.54417	0.05431	0.53784	225	N/A	N/A
$n = 35$	0.56612	0.06536	0.58482	200	207	0.53371
$\tau_{BAC} = 80\%$						
$n = 15$	0.57958	0.06369	0.61919	70	N/A	N/A
$n = 35$	0.48739	0.09271	0.66583	55	N/A	N/A
$\tau_{BAC} = 90\%$						
$n = 15$	0.56936	0.09329	0.69965	50	47	0.59883
$n = 35$	0.42346	0.10128	0.60127	11	N/A	N/A

Table 4.6: Simulated results vs. predicted parameters for the first scenario and the first network

Predicted τ_{APCF} is the value calculated from the parametric analysis (see Section 3.3.4). The analysis gives specific BAC alarm threshold which in this case is 90% for $n = 15$ and 70% for $n = 35$. For all other BAC thresholds there is no

assessment through the parametric analysis, as the table cell for those are left as **not available**.

Repeating the simulation with BAC population of 15 and 35 shows that parameters can be predicted regardless of the value of BAC population. Also APCF efficiencies exceed PCF efficiencies for all cases. However, further investigation can find heuristics for this parameter as well through extending analysis presented in Chapter 3.

Chapter 5

Conclusion

5.1 Summary

In this work, we introduced a scalable approach to detect Denial-of-Service (DoS) intrusions in lower levels of the network, say Tier-2 routers. Our research was an improvement over a previous scalable attack detection approach, Partial Completion Filters (PCF) [22], in terms of efficiency and flexibility. PCF uses a counter to count up incoming connection initiation messages and count down connection termination ones in order to track the balance. PCF also uses aggregation of multiple connections into hashed buckets to reduce the memory usage.

Although PCF has significant breakthrough as an aggregation approach in reducing memory usage and processing load, it suffers from flexibility and low efficiency. PCF has a single parameter, the counter threshold that merely depends on the number of connections in each bucket. It makes it hard to adjust PCF for a particular network with intrinsic factors such as *attack probability* of certain subnets, consequently reducing PCF flexibility. Furthermore, relying only on the initiation/termination balance causes notable false positive and false negative.

This research introduced Advanced Partial Completion Filters (APCF) to tackle flexibility and efficiency deficiencies of PCF. APCF has changed the nature of PCF counter from keeping initiation/termination (e.g., INVITE/BYE in VoIP) balance

to counting small intrusive behaviors in definite groups of incoming messages. To this end, APCF uses extra functionality named Behavior Alarming Counters (BAC) which alarms APCF whenever it encounters an intrusive behavior among a group of consecutive messages. APCF simply counts these alarms and alerts whenever they exceed a certain threshold, τ_{APCF}.

There are two parameters that have influence on the functionality of BAC, BAC population (n_{BAC}) and BAC alarm threshold (τ_{BAC}). The former is the number of consecutive messages that BAC investigates together, and the later is the maximum number of connection (session) initiation messages among those n_{BAC} messages that is tolerable to BAC. For instance if BAC population is 20 and BAC threshold is 17, then BAC investigates every 20 consecutive messages and alarms APCF counter whenever more than 17 of them are session initiation signals. For better understanding of BAC behavior refer to Figure 3.2.

In this study we also analyzed APCF theoretically in order to obtain values for parameters and relate them to a particular network. Statistical measures for a network including attack probability, number of initiation messages hashed to each bucket and all messages hashed to a bucket are taken into account to attain the parametric analysis provided in Section 3.3.4.

To compare PCF and APCF functionality, we used a measure called *efficiency* which relies on precision and recall, two factors that cover true positive, false positive and false negative rates. This measure along with ROC diagrams enabled us to compare PCF and APCF in different situations. This work considers Voice over IP networks (VoIP) as an example of vulnerable networks against partial completion DoS attacks, and provides a simulation testbed to test the validity of the contribution. The simulation is based on OMNET++ [4] with two different random networks with different topologies and traffic loads. For each network, two scenarios were designed, a massive attack scenario with more than 50% of intrusion rate, and low attack scenario

with less than 20% of intrusion rate. Provided results confirm that APCF performs better than PCF in terms of efficiency and ROC behavior. They also support the APCF parametric analysis.

In summary, the major contributions of this study are:

- Reducing PCF false positive and false negative detection ratios, as a scalable intrusion detector, by introducing an improved version called Advanced-PCF, APCF.
- Analytical evaluation of APCF parameters, given that certain statistical information is available for a particular network.
- Designing and developing a simulation testbed which is used to compare PCF and APCF performance in this work. This testbed can also be used as a future tool to expand research in this area.

Some of the work in this thesis will be published at IEEE Globecom 2008 Conference:

- A.Hamidi, S.Ganti, K.Wu, "An Aggregative Approach for Scalable Detection of DoS Attacks", *IEEE Globecom 2008 Conference, 30 November - 4 December,2008 NewOrleans, USA*

5.2 Future Work

For future work, the theoretical analysis will be expanded to consider more statistical data from the network, and to encompass BAC population; BAC population is considered as a free variable in our analysis in this study and parametric analysis is done for two other attributes, APCF threshold and BAC threshold. This can be done by approaching the parameter calculation in other ways. The distribution of attack INVITE signals amongst usual messages in a bucket is considered as a *Ball and Bins* problem. This approach leads to a Poisson distribution assumption for the messages

in the bucket, because we assume that the number of session initiation messages in a bucket is already known. If this assumption is dismissed, Poisson Distribution will be no longer in effect. Other statistical information (e.g., the frequency of initiation messages) that are easier to attain can lead to other equations which are easier to solve.

We want to perform a thorough study on the traffic distribution in VoIP networks, including different situations of normal and attacked. This work relies on the assumption that the traffic, including initiation and termination messages and intrusion signals, are distributed randomly in the scale of each bucket. Although, to our knowledge, other similar works assumed the same, a comprehensive model brings will bring out more reliable parametric analysis.

Real experiments will be of priority for this work in future. Simulations provided in this study confirm our findings and improved performance of APCF over PCF; nevertheless, important factors such as memory consumption and processing time scalability are only assessed through real experiments. PlanetLab is an excellent platform for APCF to be deployed and tested against actual Internet traffic. Furthermore, APCF hardware implementation and its challenges which have remained unattended in the present study can be works of future study items. Besides real experiments, considering real network trace data that is available in research community can be used for further analysis of APCF performance.

Bibliography

[1] *Check Point Software Technologies, Ltd.*

[2] *Cisco Systems.*

[3] *NetScreen Technologies, Inc.*

[4] *OMNET++*. Discrete Event Simulation Tool.

[5] D. Anderson, T. Frivold, and A. Valdes. Next-generation Intrusion Detection Expert Systems (NIDES), A Summary. *Computer Science Labratory, SRI International*, May 1995.

[6] S. Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. Technical Report 99-15, Chalmers University, Mar. 2000.

[7] J. Banks, J. Carson, B. Nelson, and D. Nicol. *Discrete-Event System Simulation.* Prentice Hall, Upper Saddle River, New Jersey 07458, USA, 2005.

[8] P. Barford, J. Kline, D. Plonka, and A. Ron. A Signal Analysis of Network Traffic Anomalies. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 71–82, New York, NY, USA, 2002. ACM.

[9] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[10] G. Carl, G. Kesidis, R. Brooks, and R. Suresh. Denial-of-service Attack-detection Techniques. *Internet Computing, IEEE*, 10(1):82–89, Jan.-Feb. 2006.

[11] P. K. Chan and M. V. Mahoney. Modeling Multiple Time Series for Anomaly Detection. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 90–97, Washington, DC, USA, 2005. IEEE Computer Society.

[12] E. Chen. Detecting DoS Attacks on SIP Systems. *1st IEEE Workshop on VoIP Management and Security*, pages 53–58, April 2006.

[13] Y. Ding and G. Su. Intrusion Detection System for Signal Based SIP Attacks Through Timed HCPN. *The Second International Conference on Availability, Reliability and Security*, pages 190–197, April 2007.

[14] C. Estan and G. Varghese. New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice. *ACM Transactions on Computer Systems*, 21(3):270–313, 2003.

[15] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers, 2003.

[16] T. M. Gil and M. Poletto. MULTOPS: A Data-Structure for Bandwidth Attack Detection. pages 23–38, Aug. 2001.

[17] M. H. Hassoun. *Fundamentals of Artificial Neural Networks*. The MIT Press, 55 Hayward Street Cambridge, MA 02142-1493, USA, 1995.

[18] K. Ilgun. USTAT: a Real-time Intrusion Detection System for UNIX. *IEEE Computer Society Symposium on Research in Security and Privacy Proceedings*, pages 16–28, May 1993.

[19] K. Ilgun, R. Kemmerer, and P. Porras. State Transition Analysis: a Rule-based Intrusion Detection Approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, Mar 1995.

[20] Y. Jou, F. Gong, C. Sargor, X. Wu, S. Wu, H. Chang, and F. Wang. Design and Implementation of a Scalable Intrusion Detection System for the Protection of Network Infrastructure. *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, 2:69–83, 2000.

[21] R. R. Kompella, S. Singh, and G. Varghese. On Scalable Attack Detection in the Network. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 187–200, New York, NY, USA, 2004.

[22] R. R. Kompella, S. Singh, and G. Varghese. On Scalable Attack Detection in the Network. *IEEE/ACM Transactions on Networking*, 15(1):14–25, Feb. 2007.

[23] S. Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, 1995.

[24] R. J. Larsen and M. L. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Prentice-Hall, Upper Saddle River, NJ 07458, third edition, 2001.

[25] Lawrence, A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson. CSI/FBI Computer Crime and Security Survey. *Computer Security Institute*, 2006.

[26] K. Levchenko, R. Paturi, and G. Varghese. On the Difficulty of Scalably Detecting Network Attacks. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 12–20, New York, NY, 2004.

[27] P. B. C. P. Ltd. 2007 Global NGN IP VoIP - Analyses Statistics and Forecasts. Technical report, June 2007.

[28] E. U. M. Mitzenmacher. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 40West 20th Street, NewYork, NY 10011-4211, USA, 2005.

[29] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[30] D. Moore, G. M. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pages 2–2, Berkeley, CA, USA, 2001. USENIX Association.

[31] Z.-S. Pan, S.-C. Chen, G.-B. Hu, and D.-Q. Zhang. Hybrid Neural Network and C4.5 for Misuse Detection. *International Conference on Machine Learning and Cybernetics*, 4:2463–2467, Nov. 2003.

[32] V. Paxson. Bro: a System for Detecting Network Intruders in Real-time. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium, 1998*, pages 3–3, Berkeley, CA, USA, 1998. USENIX Association.

[33] V. Paxson. An Analysis of Using Reflectors for Distributed Denial-of-service Attacks. *SIGCOMM Computer Communication Review*, 31(3):38–47, 2001.

[34] P. A. Porras and P. G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proc. 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, 1997.

[35] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association.

[36] H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, and J. Rosenberg. *SIP:Session Initiation Protocol*, June 2002. RFC 3621.

[37] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based Anomaly Detection: a New Approach for Detecting Net-

work Intrusions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 265–274. ACM Press, 2002.

[38] H. Sengar, D. Wijesekera, H. Wang, and S. Jajodia. VoIP Intrusion Detection Through Interacting Protocol State Machines. *International Conference on Dependable Systems and Networks, 2006. DSN 2006.*, pages 393–402, 2006.

[39] A. Shevtekar and N. Ansari. Do Low Rate DoS Attacks Affect QoS Sensitive VoIP Traffic? *IEEE International Conference on Communications, ICC '06*, 5:2153–2158, June 2006.

[40] D. Sisalem, J. Kuthan, and S. Ehlert. Denial of Service Attacks Targeting a SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms. *Network, IEEE*, 20(5):26–31, Sept.-Oct. 2006.

[41] N. Tuck, T. Sherwood, B. Calder, and G. Varghese. Deterministic Memory-efficient String Matching Algorithms of Intrusion Detection. In *Proceedings of the IEEE Infocom Conference*, pages 333–340, 2004.

[42] G. Vigna and R. Kemmerer. NetSTAT: a Network-based Intrusion Detection Approach. *14th Annual Computer Security Applications Conference Proceedings*, pages 25–34, Dec 1998.

[43] S. Vuong and Y. Bai. A Survey of VoIP Intrusions and Intrusion Detection Systems. *The 6th International Conference on IEEE Advanced Communication Technology*, 1:317–322, 2004.

[44] H. Wang, D. Zhang, and K. G. Shin. Detecting SYN Flooding Attacks. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1530–1539, June 2002.

[45] J. Zar. VoIP Security and Privacy Threat Taxonomy-Public Release 1.0. *Voice over IP Security Alliance (VOIPSA)*, October 2005.