Computer Communications 110 (2017) 48–58

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Real-time DDoS attack detection using FPGA[☆]

N. Hoque[a], H. Kashyap[b], D.K. Bhattacharyya[a,*]

[a] Department of Computer Science & Engineering, Tezpur University Napaam, Tezpur 784028, Assam, India
[b] Department of Computer Science Donald Bren School of Information and Computer Sciences, University of California Irvine, 3019 Donald Bren Hall Irvine 92697-3435, CA, USA

A R T I C L E I N F O

Article history:
Received 2 December 2016
Revised 1 May 2017
Accepted 28 May 2017
Available online 31 May 2017

Keywords:
DDoS attack detection
FPGA
Correlation measure

A B S T R A C T

A real-time DDoS attack detection method should identify attacks with low computational overhead. Although a large number of statistical methods have been designed for DDoS attack detection, real-time statistical solution to detect DDoS attacks in hardware is only a few. In this paper, a real-time DDoS detection method is proposed that uses a novel correlation measure to identify DDoS attacks. Effectiveness of the method is evaluated with three network datasets, viz., CAIDA DDoS 2007, MIT DARPA, and TUIDS. Further, the proposed method is implemented on an FPGA to analyze its performance. The method yields high detection accuracy and the FPGA implementation requires less than one microsecond to identify an attack.

1. Introduction

Distributed Denial of Service (DDoS) attack, also known as coordinated attack, is performed using a large number of compromised machines. Although DDoS attacks can be launched easily, they are catastrophic to websites and servers. Attackers use different architectures, such as agent-handler, IRC-based and web-based botnet to generate DDoS attacks. The main purposes of DDoS attacks are resource consumption and bandwidth utilization, such that victims cannot provide services to legitimate users.

DDoS attacks can be generated in application, transport, network and physical layers of TCP/IP framework using different protocols, such as ICMP, TCP, HTTP and UDP. A TCP SYN flooding attack overwhelms the target with SYN packets by exhausting the connection table, whereas ICMP and UDP-based flooding attack consumes network bandwidth by sending forged attack packets. Besides flooding DDoS attacks, a multivector DDoS [1] attack combines large UDP flood with slow HTTP GET flood such that a defense mechanism deployed at victim machine would deal with only UDP flooding, while the resources of the victim will be exhausted by HTTP floods.

Many techniques have been adopted to detect DDoS attacks, such as machine learning, knowledge-based, soft computing and statistical. Statistical techniques perform correct analysis of network traffic features during detection with lesser computations per packet. In order to detect DDoS attacks in real time, the detection mechanism should be able to detect attacks efficiently from a small set of relevant features. In the recent past, several correlation-based statistical measures have been proposed in the literature which can be used to detect DDoS attacks. However, such state-of-the-art correlation measures fail to provide high DDoS detection accuracy when using a small set of traffic features. Therefore, an effective correlation measure is needed to classify DDoS attack traffic in real-time even when using a small number of traffic features.

1.1. Motivation

A major limitation of most correlation-based statistical methods for DDoS detection is that due to shifting, scaling and shifting-and-scaling correlations [2] among network traffic features, they fail to overcome the occurrence of large number of false positives. Others, statistical methods impose a high computational overhead when a large number of features are included for analysis. Consequently, such methods fail to perform DDoS detection in real-time. Pearson's, Spearman's, and Kendall's correlation measures fail to identify the difference between a normal and an attack packet when shifting-and-scaling correlation exists among feature values of two packets. Moreover, these correlation measures are not suitable for analysis of network traffic in real-time with less number of features.

The intensity of DDoS attack traffic has been increasing and which is evident from recent Arbor network statistics (400 Gbps attack launched at CloudFare in 2014). Thus, a software-based

☆ Fully documented templates are available in the elsarticle package on CTAN.
* Corresponding author.
E-mail addresses: tonazrul@gmail.com (N. Hoque), kashyaph@uci.edu (H. Kashyap), dkb@tezu.ernet.in (D.K. Bhattacharyya).

http://dx.doi.org/10.1016/j.comcom.2017.05.015
0140-3664/© 2017 Elsevier B.V. All rights reserved.

solution, even when using networked and distributed systems, may not be adequate to handle massive attack traffic. Moreover, software-based attack detection systems themselves might be targeted or bypassed by the attacker. On the other hand, application specific hardware solutions are less susceptible to immediate attacks and are capable of providing significantly higher detection throughput in comparison to the software solutions. The proposed solution cannot be claimed to be less susceptible to attacks because it is not implemented fully on hardware.

1.2. Contribution

In this paper, a real-time DDoS detection method is proposed using an effective correlation measure referred to as *NaHiD*. The major contributions of this paper are mentioned below.

1. A robust correlation measure, referred to as *NaHiD* capable of handling shifting and scaling correlations among the features of two traffic objects.
2. Selection of a small subset of traffic features for DDoS detection without compromising detection accuracy.
3. A fast DDoS attack detection method implemented on software as well as hardware platforms. The hardware implementation uses a Field Programmable Gate Arrays (FPGA) device and requires less than one microsecond to classify an incoming traffic sample as attack or normal.
4. Evaluation of the proposed *NaHiD* and DDoS attack detection method in terms of detection time and accuracy on three benchmark network datasets.

1.3. Paper organization

The paper is organized as follows. In Section 2, the state-of-the-art statistical DDoS attack detection methods are discussed. The proposed DDoS detection method using an effective correlation measure is presented in Section 3. The framework proposed for DDoS attack detection using a combination of software and hardware modules is presented in Section 4. The experimental evaluation of the implemented method is discussed in Section 5, followed by our conclusions in Section 6.

2. DDoS attack detection: state-of-the-art

Distributed Denial of Service attack is an aggressive and menacing intrusive threats to online servers, websites, networks and clouds. The aim of DDoS attack is to exhaust resources and to consume bandwidth of a network system. Due to the coordinated nature of DDoS attack, an attacker can generate huge amount of attack traffic using a large number of compromised machines to crash a system or website [3]. Many organizations such as amazon, eBay, CNN and Yahoo were the victims of DDoS attacks in the recent past. Moreover, the intensity of DDoS attacks has been increasing day-by-day and recently a 400 Gbps massive DDoS attack took place at CloudFlare.[1] A DDoS attack is not only a major security threat for cloud data but it causes a significant economic loss. Economic Denial of Service (EDoS) attack can be occurred on cloud resources business whereas resource reservation for an application is vulnerable for DDoS attacks in clouds.

2.1. DDoS detection with different correlation measures

Many statistical measures, such as correlation, entropy, covariance, divergence, cross-correlation, and information gain have been used for network traffic analysis towards anomalous pattern identification [4]. Statistical measure uses multiple features extracted from network traffic. Features may be extracted either from individual network packets or from network traffic flows. Different feature selection and feature extraction methods have been discussed in literature [5–7]. Moreover, we have performed an empirical study of different proximity measures such as Euclidean distance, City-block distance, Mahalanobish distance, Person, Spearman and Kendall Correlations. From the empirical analysis of different proximity measures, compared to distance-based measures, statistical or correlation-based measures are found useful for network traffic analysis. Moreover, from the literature survey we found that statistical or correlation measures are effective for discriminating the attack behavior of network traffic from legitimate behavior. Therefor, we consider a statistical-based method for DDoS attack detection.

A rank correlation-based method, called Rank Correlation-based Detection (RCD) is proposed by Wei et al. [8] to detect reflector DDoS attacks. In this method, authors use Spearman's rank correlation among network traffic flows and claim that the correlation between two malicious flows must be remarkably strong compared to normal flows. An information theoretical approach using Kolmogorov complexity is used for detection of DDoS attacks. In this method, Kulkarni et al. [9] state that for some correlated strings, the joint complexity of random strings is smaller than the sum of the complexities of the individual strings and the joint complexity is inversely proportional to the amount of correlation. This technique is applied to correlate network traffic flows for detection of possible DDoS attacks flows. A DDoS detection method using chaos analysis of network traffic entropy is proposed by Xinlei Ma and Yonghong Chen [10]. The method is based on entropy variation between source IPs and destination IPs. Tan et al. [11] propose a multivariate correlation analysis (MCA) method using triangular area map structure to identify DoS attack traffic. MCA is used to determine geometric correlation between any two distinct features of a network traffic.

Intrusion detection systems such as STAT [12] and Haystack [12] are built using statistical approaches. Ye et al. [13] apply Hotelling's T^2 test to detect two types of anomalies: (i) counter relationship anomalies and (ii) mean-shift anomalies. They use multivariate statistical analysis to detect anomalies by generating profile for normal traffic and use the normal profile for anomaly detection. The method is validated on small as well as large multi-day datasets. In large datasets, the method shows zero false alarm on normal traffic. A novel anomaly detection scheme based on Principle Components Analysis is proposed by Shyu et al. [14]. They apply PCA on the correlation matrix of normal traffic and use Mahalanobis distance to measure differences among traffic instances. The PCA-based classifier consists of a major component score that detects extreme observations with large values on selected original features and a minor component score that helps in detection of observations that do not conform to the normal correlation structure.

A method called Kernel-based Online Anomaly Detection (KOAD) is proposed by Ahmed et al. [15]. The algorithm incrementally constructs and maintains a dictionary of input vectors that define the region of normal behavior. As new traffic arrives, the dictionary is updated. This algorithm works sequentially and detects anomaly in real time. Feinstein et al. [16] present a method to detect DDoS attacks using entropy and packet attribute distributions. Entropy is computed on each network traffic sample. They observe that entropy values fall in a narrow range when the network is not under attack. But during an attack period, entropy values exceed the range in a detectable manner. Santiago-Paz et al. [17] propose a method based on entropy to classify anomalous traffic in an enterprise network. They apply Mahalanobis distance to describe an ellipse that characterizes network entropy which

[1] blog.cloudflare.com.

allows to determine whether a given actual traffic-slot is normal or anomalous.

2.2. Discussion

Even though a large number of anomaly-based DDoS attack detection techniques have been introduced, many issues remain to be addressed. The following are our observations on DDoS attack detection.

1. To provide real time detection, a DDoS detection mechanism must use a few relevant features only. Hence, selection or extraction of most relevant and optimal subset of features from network traffic for attack detection is a must.
2. Most statistical DDoS detection methods are dependent on multiple user defined thresholds and performance of such mechanisms is highly influenced by these thresholds. Such thresholds need to be updated dynamically to cope with changes in a network.
3. Most victim-end detection systems are not cost effective, and cannot provide real-time performance.
4. Moreover, entropy-based DDoS detection systems are effective in identifying anomalous traffic because entropy of traffic feature distributions provides useful information to measure distance among traffic groups.

2.3. Limitations of the related correlation measures

In literatures, though we have found many statistical DDoS detection mechanisms but most of them suffer from two major limitations.

1. Use of entropy, correlation or mutual information cause consumption of high computational time during DDoS attack detection. As a consequence, they cannot perform in real-time.
2. A real-time DDoS detection solution demands for minimum number of features to be used during traffic analysis, whereas correlation measures such as Spearman, Pearson, and Kendall are often fail to provide high detection accuracy over less number of features. An example to justify it is reported in the subsequent section w.r.t Table 2.
3. To reduce computational time during attack detection, Giseop No and Ilkyeun Ra use compression entropy for DDoS detection [18]. But, the method is found insensitive and produces high false alarm rate.

3. Proposed correlation measure

In this section, we introduce our correlation measure, called *NaHiD*, designed for network anomaly detection, towards DDoS detection. For any two objects X and Y, the proposed correlation measure considers standard deviation and mean of the two objects. First, it computes mean and then standard deviation for X and Y

meanX=mean value of X

meanY=mean value of Y

SDX=standard deviation of X

SDY=standard deviation of Y

NaHiD defines the correlation between the two objects X and Y as shown in Eq. (1).

$$NaHiD(X,Y) = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{|X(i)-Y(i)|}{||meanX-SDX|-X(i)|+||meanY-SDY|-Y(i)|} \quad (1)$$

where, n is the dimension of the two objects X or Y.

For simplicity, Eq. (1) can be written as:

$$NaHiD(X,Y) = 1 - \frac{1}{n}\sum_{i=1}^{n}D(i)$$

where,

$$D(i) = \frac{|X(i)-Y(i)|}{||meanX-SDX|-X(i)|+||meanY-SDY|-Y(i)|} \quad (2)$$

The proposed measure computes correlation based on deviation of the two objects from their mean values. The proposed measure computes two values; (i) absolute distance between X and Y and (ii) deviation of X and Y from their mean and standard deviation. The proposed correlation measure can be differentiated from the Pearson correlation, Spearman correlation and Kendall's correlation as follows.

- Similar to Pearson correlation, *NaHiD* considers standard deviation and mean value. However, unlike Pearson correlation, *NaHiD* computes absolute distance between the two objects.
- *NaHiD* computes the difference between each parameter value of the two objects, whereas Spearman correlation computes difference between the ranks for each parameter.
- Unlike *NaHiD*, the Kendall's correlation measure computes distance between number of concordant pairs and number of discordant pairs [19].

3.1. Proof of identity and symmetricity

NaHiD satisfies three basic properties of a metric such as: (i) identity and (ii) symmetricity. The proofs of these properties are given below.

(i) **Identity:** Correlation between an object and itself is always 1, i.e., for any object X, $NaHiD(X,X) = 1$.

Proof. $|X(i) - X(i)| = 0$ for $i = 1, 2, \cdots n$

From Eq. (1)

$$1 - \frac{1}{n}\sum_{i=1}^{n}\frac{0}{||meanX-SDX|-X(i)|+||meanX-SDX|-X(i)|}$$
$$= 1 - 0 = 1$$

Hence, $NaHiD(X,X) = 1$ □

(ii) **Symmetricity:** For any two objects X and Y, $NaHiD(X,Y) = NaHiD(Y,X)$.

Proof. The proposed measure satisfies the commutative law for both the numerator as well as denominator of $D(i)$.

For numerator: $|X(i) - Y(i)| = |Y(i) - X(i)|$

For denominator:

$||meanX - SDX| - X(i)| + ||meanY - SDY| - Y(i)| = ||meanY - SDY| - Y(i)| + ||meanX - SDX| - X(i)|,\ \forall i = 1, 2, \cdots n.$

Hence, $NaHiD(X,Y) = NaHiD(Y,X)$. □

Like Pearson correlation, *NaHiD* also does not satisfy the transitive dependency property. It is mostly due to the occurrences of shifting or scaling or shifting-and-scaling patterns in samples as well as in test objects. Detailed discussion on the shifting, scaling, and shifting-and-scaling patterns in numeric data objects is beyond the scope of this paper. However, Ahmed et al. [2] report a detailed discussion on these patterns.

The effectiveness of the measure is that it can analyze correlation value between any two samples with vary less number of parameters. However, as shown in Tables 1 and 2, Pearson, Spearman and Kendall's correlation measure yield identical correlation value even if the samples are uncorrelated. That is the Pearson, Spearman and Kendall correlation measure is not very effective to determine correlation values for low dimensional sample objects.

Table 1
List of five objects.

Object	F_1	F_2	F_3
O_1	365	2.52	0.9533
O_2	379	2.55	0.9709
O_3	345574	12.98	0.94
O_4	166453	12.7	0.9866
O_5	357663	12.79	0.94

F_1: Packet Rate, F_2: Variations of Source IPs
F_3: Entropy of Source IPS

Table 2
Correlation values of different object pairs.

Object pair	Pearson correlation	Spearman correlation	Kendall correlation	NaHiD VERC
O_1, O_2	1	1	1	0.9917
O_2, O_3	1	1	1	0.5600
O_3, O_1	1	1	1	0.5600
O_3, O_5	1	1	1	0.9924
O_5, O_2	1	1	1	0.5600

But, a real-time DDoS detection method should consider only a few traffic parameters during network traffic analysis. Therefore, the proposed correlation measure is tested for real-time DDoS attack detection using only a few traffic parameters, as discussed in the following section.

3.2. NaHiD *for DDoS detection using Variation, Entropy of Source IPs and Packet Rate:* NaHiD$_{VERC}$

The proposed correlation measure is used for DDoS attack detection in real-time. The measure considers three features for DDoS attack traffic classification. Features are extracted from raw network traffic samples of 1 second time window. From each time window, we extract three distinct features, viz., entropy of source IPs, variation index of source IPs and packet rate. These features will be extracted during pre-processing of network traffic as explained in Section 4. The measure is introduced for estimation of correlation between the samples of any two time windows by analyzing the three distinct features. This variant of the *NaHiD* measure is herein designated as *NaHiD$_{VERC}$*.

Let us consider five network traffic objects, viz., O_1, O_2, O_3, O_4, O_5, where O_5 and O_3 are normal and the others are attack objects. The values of the considered objects are shown in Table 1 and the correlation values among different pairs of objects are shown in Table 2. As shown in Table 2, the state-of-the-art correlation measures yield identical correlation values for (normal, normal) and (normal, attack) pairs. However, the proposed *NaHiD$_{VERC}$* correlation measure yields distinguishable variations for these pairs.

The proposed measure computes correlation among all extracted features and generates a normal profile during the analysis period. Similarly, a test profile will be generated for the observed network traffic during the testing period using the same correlation measure. If the distance between normal and observed traffic is greater than a certain threshold value, an alarm will be generated indicating that an attack has occurred. A framework of the proposed method is shown in Fig. 1.

We consider the following assumptions in our detection method.

1. If the entropy of source IPs within a short interval of time is very high and the packet rate is also very high, the attack probability is high.
2. If variation among source IPs is very high and the packet rate is also high, the attack probability is high.
3. There must be a notable difference between normal and attack traffic.

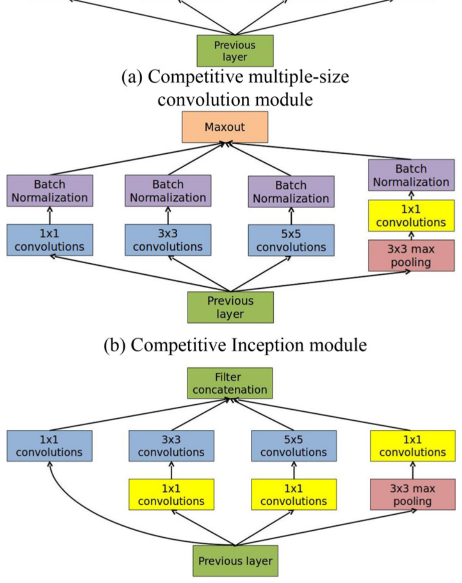


Fig. 1. Architecture of the DDoS detection system.

3.3. *How does* NaHiD$_{VERC}$ *overcome the limitations*

1. *NaHiD$_{VERC}$* considers only three distinct network traffic features to identify anomalous samples. The computational time of the measure is very low.
2. Unlike Pearson, Spearman and Kendall's correlation measure, the proposed correlation measure is very sensitive for any two unrelated traffic samples. This property is very important to analyze network traffic behavior. Because, in many situations an attacker may behave almost in a similar manner like a legitimate user.
3. The proposed measure is very effective to detect both low-rate and high-rate DDoS attacks.
4. The measure is easily implementable on hardware.

4. Proposed DDoS attack detection framework

A framework for detection of DDoS attacks, using the *NaHiD$_{VERC}$* measure is reported in Fig. 1. The framework aims to detect DDoS attacks in real time at the victim end. A combined software and hardware-based approach is adopted for implementation of the framework, in order to classify attack and normal traffic in real-time with detection accuracy.

The proposed framework consists of three major components, namely, a pre-processor, a dedicated hardware module for attack detection, and a security manager. Apart from those, the framework includes a router to capture network traffic and two databases, namely Log database and Profile database. The components enclosed inside the dotted box represent the real time components of the framework. Whereas, the components outside of this box operate in an offline manner to minimize the false alarm rate.

Raw network traffic are captured from a mirror port of the router as TCP/IP network layer packets, which are subsequently sent to the pre-processor module. The pre-processor groups the packets received during each 1 *second* time period into a 1 *second* window. The pre-processed traffic instance for the window is sent to the attack detection module, which classifies the instance as either normal or attack. The pre-processed traffic instance and its classification are stored in the Log database. The security manager maintains a normal profile and a threshold value in the Profile database, to be used by the attack detection module. The security manager incrementally recomputes the normal profile and the threshold based on their previous values and the updated classification logs, stored in the Profile database and the Log database, respectively. The major components of the framework are discussed below.

4.1. The pre-processor

Feature selection is an important pre-processing step for network traffic classification. One can use mutual information [6], correlation [20], rough sets [21] or fuzzy sets [22] for feature selection and extraction. A real-time anomaly detection system captures raw-network traffic from the network interface and analyzes this traffic. However, analysis of raw network traffic is time consuming. Moreover, all fields of packet header are not equally important for analysis. Therefore, only a few important features from network traffic are used for attack detection. In particular, three features, viz., entropy of source IPs, variation index of source IPs and packet rate are extracted over multiple 1 *second* time windows. To split the dataset, the *editcap* and the *tshark* tools are used, with the specific parameters. The entropy of source IPs, variation among source IPs and packet rate are calculated using custom scripts. We introduce each of these three below.

4.1.1. Entropy of source IPs

Network traffic analysis is performed to detect malicious traffic running through a network. During traffic analysis, one considers either packet header information alone or packet header and raw data information together. In both the schemes, TCP packet or IP packet header fields are analyzed to detect network level anomalies. Source IP, source port, destination IP, destination port, protocols, and flags of the TCP header are very useful to detect anomaly behavior of a network packet. We calculate entropy and variation among source IPs and packet rate for each sample traffic. Entropy of source IPs for each sample traffic is calculated using Eq. (3).

$$H(X) = -\sum_{i}^{n} p(x_i) \log_2 p(x_i) \tag{3}$$

where X is a random variable representing source IPs and n is the total number of possible values for the source IP.

4.1.2. Variation index of source IPs

The variation among source IPs is the rate of change of IP source addresses w.r.t time. We compute the variation index, i.e., V of source IPs using Eq. (4).

$$V(X) = \frac{\delta}{N} \tag{4}$$

where, δ=Number of changes of source IP in a given time window and N=Total number of source IPs in a time window.

If the source IP address changes frequently, the variation will be high. We compute entropy of source IPs, variation among source IPs and the packet rate for each network traffic sample. Observation of the behavior of high rate DDoS flooding attacks shows that flooding attacks can be generated by real attackers as well as zombies. If spoofed source IP addresses are used during TCP SYN flooding DDoS attack, the entropy of source IPs will be high, and the variation of source IPs w.r.t time window will also be high. This behavior of network traffic is similar to normal network traffic behavior. So, either the entropy of source IPs or variations of source IPs alone is not sufficient for effective identification of high rate DDoS flooding attacks. On the other hand, if the source IPs are real during UDP or ICMP flooding attacks, from certain IP addresses attack packets will be sent to the victim. If the randomness of the source IPs is high, entropy will also be high. Fig. 2 shows the difference of variations of source IPs between normal and attack traffic. Similarly, the entropy due to variation in source IPs between normal and attack traffic is shown in Fig. 3. The comparison of packet rate between normal and attack traffic on CAIDA2007 dataset is shown in Fig. 4.

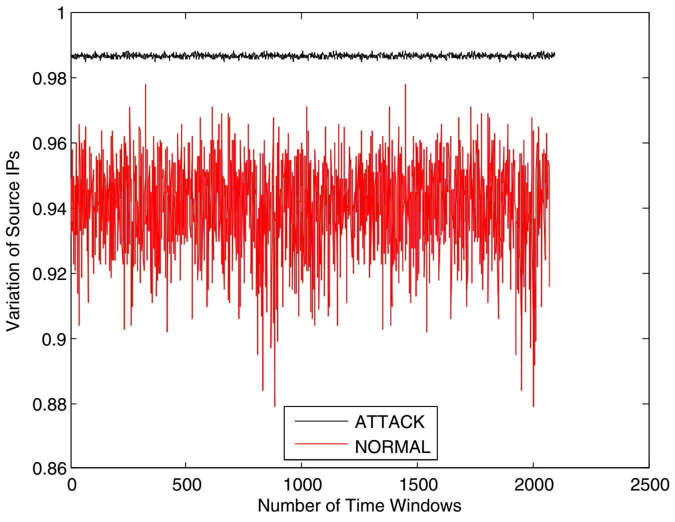


Fig. 2. Comparison of variation of source IPs on CAIDA2007.

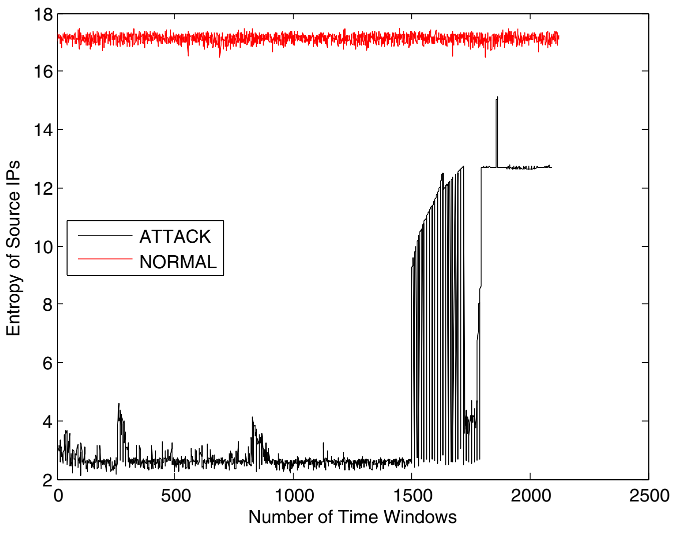


Fig. 3. Comparison of entropy of source IP on CAIDA2007.

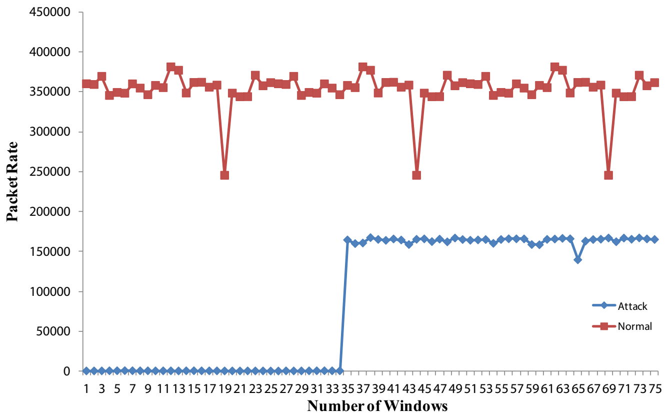


Fig. 4. Comparison of packet rate on CAIDA2007.

4.2. The attack detection module

As mentioned earlier, the main objective of the proposed DDoS attack detection framework is to detect attacks in real-time. This requires an attack detection module that provides high throughput and accuracy. High throughput is also essential for scalability, which is required in case of DDoS attacks. Software-based solutions are inefficient for real-time applications as they require large number of general purpose CPU cycles. Herein, a dedicated hardware architecture for the DDoS attack detection module is considered for implementation on FPGAs. Recent FPGA devices provide high performance and can accommodate large and complex logic. Nevertheless, the detection module may also be implemented as an Application Specific Integrated Circuit (ASIC). However, FPGAs provide dynamic adaptability, which is important for applications requiring frequent changes in their configurations, such as detection of frequently evolving DDoS attacks.

It should be noted that, the pre-processor and the security manager modules are implemented separately using software. Machines implementing these modules and the FPGA can communicate using the high speed I/O interfaces supported by the modern FPGAs, such as PCI and Ethernet.

The attack detection module receives the traffic instance from the pre-processor module. Additionally, it receives the normal profile and a threshold value from the Profile database created by the security manager. Each of the traffic instance and the normal profile are vectors consisting of three traffic features. The attack detection module first computes the *NaHiD_{VERC}* between the input traffic instance and the normal profile. The calculated correlation value is then compared with the threshold to classify the incoming traffic instance as either attack or normal. The classification result is stored in the Log database for offline analysis by the security manager. Further, an alarm is generated in case the instance is classified as an attack.

The operations performed by the detection module on the input values are presented below. Vectors *X* and *Y* represent the traffic instance and the normal profile.

$$(X[3], Y[3]) = Vectors\,to\,be\,measured$$
$$TH = Threshold$$
1. $M_X = \dfrac{X_1 + X_2 + X_3}{3}, \quad M_Y = \dfrac{Y_1 + Y_2 + Y_3}{3}$
2. $(M_X)^2 = M_X \times M_X, \quad (M_Y)^2 = M_Y \times M_Y$
3. $M_{X^2} = \dfrac{X_1^2 + X_2^2 + X_3^2}{3}, \quad M_{Y^2} = \dfrac{Y_1^2 + Y_2^2 + Y_3^2}{3}$
4. $SD_X = \sqrt{|M_{X^2} - (M_X)^2|}, \, SD_Y = \sqrt{|M_{Y^2} - (M_Y)^2|}$
5. $N_1 = |X_1 - Y_1|, \quad N_2 = |X_2 - Y_2|, \quad N_3 = |X_3 - Y_3|$
6. $D_1 = ||M_X - SD_X| - X_1| + ||M_Y - SD_Y| - Y_1|,$
   $D_2 = ||M_X - SD_X| - X_2| + ||M_Y - SD_Y| - Y_2|,$
   $D_3 = ||M_X - SD_X| - X_3| + ||M_Y - SD_Y| - Y_3|$
7. $NaHiD_{VERC}(X,Y) = |1 - \dfrac{\frac{N_1}{D_1} + \frac{N_2}{D_2} + \frac{N_3}{D_3}}{3}|$
8. $A \Leftrightarrow TH > NaHiD_{VERC}(X,Y)$

The features in the X and Y vectors are of size 11 bits. This length has been carefully selected by analyzing the precision required for a stable detection accuracy on the target dataset. Some of these operations are not efficient when implemented on hardware. Division by any number other than a power of two is costly. Steps 1, 3, and 7 contain division by three operations. The divisor is the number of features in each of the input vector. Therefore, if an additional fourth feature of all zeros is added to *X* and *Y*, the divisor will be changed to four, without any change to the dividend. Division by four can be implemented as 2 bit right shift on the wire and does not require any additional execution cycle. Therefore, steps 1, 3, and 7 are modified as depicted below.

1. $M_X = \dfrac{X_1 + X_2 + X_3}{4}, \quad M_Y = \dfrac{Y_1 + Y_2 + Y_3}{4}$
3. $M_{X^2} = \dfrac{X_1^2 + X_2^2 + X_3^2}{4}, \, M_{Y^2} = \dfrac{Y_1^2 + Y_2^2 + Y_3^2}{4}$
7. $NaHiD_{VERC}(X,Y) = |1 - \dfrac{\frac{N_1}{D_1} + \frac{N_2}{D_2} + \frac{N_3}{D_3}}{4}|$

Note that step 7 contains additional division operations with divisor value other than power of 2, such as $\frac{N_1}{D_1}$. This divisor can be any integer value of size 23 bits based on the input vectors. Therefore, it cannot be modified for efficiency. Further, all the subtraction operations in the proposed correlation measure are absolute difference. In order to reduce the area requirement by minimizing the number of subtractor components, the subtraction operations in steps 4 and 7 are performed as absolute difference. These subtraction results will not be affected as they are always positive.

In the implementation of the detection module, the following notations are used to simplify representation.

$$ax_1 = X_1 + X_2, \quad ay_1 = Y_1 + Y_2, \quad M_X = \frac{ax_1 + X_3}{4}$$
$$M_Y = \frac{ay_1 + Y_3}{4}, \quad mx_1 = X_1^2, \quad mx_2 = X_2^2$$
$$mx_3 = X_3^2, \quad my_1 = Y_1^2, \quad my_2 = Y_2^2, \quad my_3 = Y_3^2$$
$$(M_X)^2 = M_X \times M_X, \quad (M_Y)^2 = M_Y \times M_Y$$
$$amx_1 = mx_1 + mx_2, \quad amy_1 = my_1 + my_2$$
$$M_{X^2} = \frac{amx_1 + mx_3}{4}, \quad M_{Y^2} = \frac{amy_1 + my_3}{4}$$
$$V_X = |M_{X^2} - (M_X)^2|, \quad V_Y = |M_{Y^2} - (M_Y)^2|$$
$$SD_X = \sqrt{V_X}, \quad SD_Y = \sqrt{V_Y}$$
$$MSD_X = |M_X - SD_X|, \quad MSD_Y = |M_Y - SD_Y|$$
$$DX_1 = |MSD_X - X_1|, \quad DY_1 = |MSD_Y - Y_1|$$
$$DX_2 = |MSD_X - X_2|, \quad DY_2 = |MSD_Y - Y_2|$$
$$DX_3 = |MSD_X - X_3|, \quad DY_3 = |MSD_Y - Y_3|$$
$$D_1 = DX_1 + DY_1, \, D_2 = DX_2 + DY_2, \, D_3 = DX_3 + DY_3$$
$$N_1 = |X_1 - Y_1|, \quad N_2 = |X_2 - Y_2|, \quad N_3 = |X_3 - Y_3|$$
$$Q_1 = \frac{N_1}{D_1}, \quad Q_2 = \frac{N_2}{D_2}, \quad Q_3 = \frac{N_3}{D_3}$$
$$aQ_1 = Q_1 + Q_2, \quad aQ_2 = \frac{aQ_1 + Q_3}{4}$$
$$NaHiD_{VERC} = |1 - aQ_2|, \quad aT = NaHiD_{VERC} - TH$$

In terms of computation cycles, the module may be implemented as given below in Fig. 5. It requires 42 computation cycles to classify an incoming traffic instance as either attack or normal. The circles, the octagons, the ellipses, the stars, and the hexagon represent addition or absolute difference, multiplication, square root, division, and subtraction operations, respectively. Note that, in the first iteration, the module will require additional cycles for loading inputs. However, during the later iterations, the inputs can be loaded during the execution of the previous iteration.

Although the objective of the module is to detect attacks in real time, that is to provide the maximal throughput, the resource requirements should be minimal. This is due to the fact that FPGA resources are limited. Many widely used FPGAs have lesser resources than the recent ones. In order to map the module to the older devices, minimizing the resource requirements has been considered as an objective.

The implementation model presented in Fig. 5 requires five adders, three multipliers, one square root calculator, and one divider. The square root operation has been implemented using the cordic algorithm, provided as a Xilinx LogiCore IP. Similarly, the LogiCore IP for division operation is used to perform fractional division.

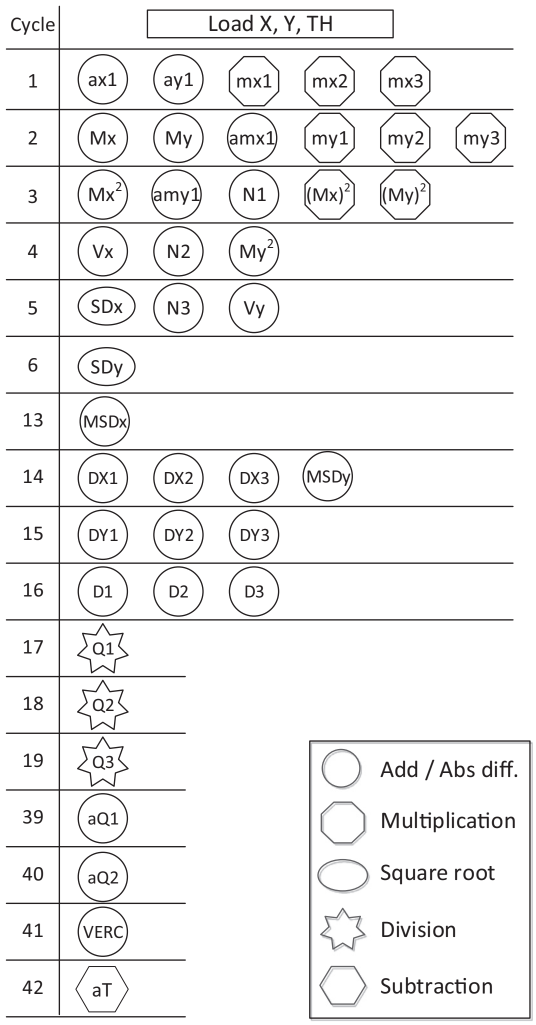


Fig. 5. Implementation model of the detection module.

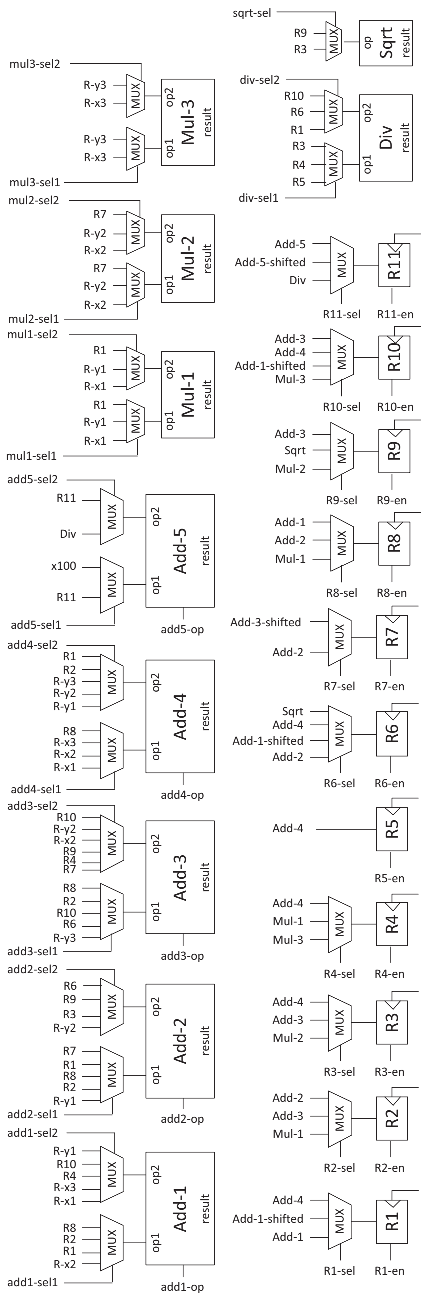


Fig. 6. Datapath design of the detection module implementation.

4.2.1. The control unit

In order to separate control and computation, the control unit and the datapath are developed separately. However, they are so intrinsically dependent on each other that it is difficult to discuss one before the other. The datapath consists of the operators for computation and the registers to store the inputs and the intermediate computation results. The control unit sends the multiplexer select signals used by the operations and the registers to select input.

The input vectors *X* and *Y* and the threshold value are used in many later computation cycles of each iteration. Therefore, these values are stored in dedicated registers as shown in Tables 3. Additionally, eleven registers, designated as R1–R11, are used to store the results of the intermediate computations. Fig. 3 depicts the contents of these registers throughout the computation cycles in each iteration. The control unit enables the write operations on these registers and selects the appropriate operation results to store in them using multiplexers.

The control unit is implemented as a finite-state machine (Mealy machine). It comprises of 21 states. The first iteration has one additional state, where the machine waits for the start signal. After receiving the start signal, the machine operates in a cyclic manner.

On the 42nd computation cycle, the $NaHiD_{VERC}$ value stored in the R11 register is compared with the input TH value. When $NaHiD_{VERC}$ is smaller than TH, the output denoting the classification result is raised high to signal an attack. This signal is accompanied by another output signal to denote the validity of the classification result.

4.2.2. The datapath

The datapath design of the detection module implementation is depicted in Fig. 6. The interconnections among the datapath components are not shown in the figure, in order to simplify representation. Add-1, Add-2, Add-3, Add-4, and Add-5 are the five adders of the design, which asynchronously perform addition and absolute difference operations. Add-1, Add-2, Add-3, and Add-4 have operand width of 23 bits. Whereas, Add-5 operates on inputs with 12.8 fixed point representation (i.e., 20 bits in size). The fixed point representation has been used since Add-5 operates on the outputs of the divider, consisting of integer and fractional parts. Mul-1, Mul-2, and Mul-3 are the three 11 × 11 multipliers used

Table 3
Register contents during the computation cycles.

Cycle	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
1											
2	ax_1	mx_1	mx_2	mx_3			ay_1				
3	M_x	amx_1					M_y	my_1	my_2	my_3	
4				M_x^2	N_1	M_{x^2}		amy_1	M_y^2		
5			V_x	N_2						M_{y^2}	
6			N_3						V_y		
13						SD_x					
14		MSD_x							SD_y		
15	DX_1	DX_2				DX_3		MSD_y			
16								DY_1	DY_2	DY_3	
17	D_1					D_2				D_3	
39											Q_1
40											aQ_1
41											aQ_2
42											VERC

in the implementation. Modern FPGA devices contain dedicated multiplier blocks or DSP slices to perform large multiplication operations efficiently in one cycle.

As discussed earlier, division by any number other than a power of two is costly in terms of hardware implementation. A LogiCore IP is used to perform the division operations in the implementation. The divisor IP imposes a latency of 22 computation cycles, which is two more than the sum of the sizes of the integer and the fractional part of the division result. Therefore, the latency can be adjusted based on the precision required for attack detection. However, the divisor provides a bandwidth of one division per clock cycle.

In this implementation, the LogicCore IP for Cordic algorithm is configured to perform square root calculation on integers of size 22 bits. The core generates square root results of size 12 bits. It provides a latency of 8 cycles and generates a new result in every cycle.

Fig. 6 does not include the registers used to store the input data received by the attack detection module. Six registers, each of size 11 bits, store the normal profile and the traffic instance vectors (three-dimensional) received as inputs. The threshold value is stored in another register of size 8 bits. Apart from that, the other eleven registers depicted in Fig. 6 have varied width, depending on the width of the data to be stored in them. The R5, R7, and R11 are 11, 12, and 20 bits wide, respectively. All other registers are of size 24 bits.

These registers store different results based on the register enable and the multiplexer select signals received from the control unit on the rising edge of the clock. It may be noted that, certain registers store the shifted version of an operation result. For instance, R1 stores both the Add-1 result and the shifted Add-1 result. The operation result is right-shifted by 2 bits to perform division by four on the wire.

4.3. The security manager

As mentioned earlier, the security manager operates offline. Therefore, it can perform detailed analytics on the attack detection logs using techniques from machine learning and statistics. Further, feedbacks from human experts may be used to validate the analytics results. The major tasks performed by the security manager are discussed below.

4.3.1. Update of the normal profile and the threshold

The security manager initially calculates the normal profile that best represents all the normal traffic instances in the training dataset. Additionally, it also finds the optimal threshold value that provides the highest classification accuracy for the attack and the normal instances in the training set. These values are initially loaded into the Profile database. However, normal traffic behavior as well as the DDoS attack patterns change over time. Consequently, the initial normal profile may significantly differ from normal traffic in a real network. Similarly, the initial threshold may not be able to detect new types of DDoS attacks. Therefore, these two values need to be dynamically updated in regard to new attack and normal traffic observed during deployment. The update process is performed incrementally based on the previous values stored in the Profile database and the traffic and classification logs stored in the Log database. The updated normal profile and the threshold are written back to the Profile database, which are subsequently used by the h/w detection module.

4.3.2. Validation of classification results

Although the normal profile and the threshold are updated dynamically, false classification by the detection module is possible due to frequent changes in attack and normal patterns over time. False positives as well as false negatives by the detection module are costly, particularly in case DDoS attacks. Since the security manager operates in offline, sophisticated analytics methods and human experts may be used to validate classification results. In case of a false classification by the detection module, the normal profile and the threshold are further adapted, in order to improve classification accuracy for new types of legitimate traffic.

5. Experimental results

In order to evaluate the effectiveness of the proposed DDoS detection method, an experiment was carried out on a workstation with 12GB main memory, 2.26 GHz Intel Xeon processor and 64-bit Windows 7 operating system. The proposed DDoS detection framework was initially implemented and evaluated in software. Additionally, to evaluate the throughput of the proposed method, a prototype of the hardware based attack detection module was implemented on a Xilinx Virtex-5 FPGA device (XC5VLX50T).

5.1. Experimental evaluation

To validate the detection method we use three network intrusion datasets, viz., CAIDA DDoS 2007 [23],[2] DARPA[3] and TUIDS [24]. The performance of our method in terms of accuracy is shown in Figs. 7–9 for CAIDA, TUIDS and DARPA datasets respectively.

[2] www.caida.org/data.

[3] www.ll.mit.edu/ideval/data.

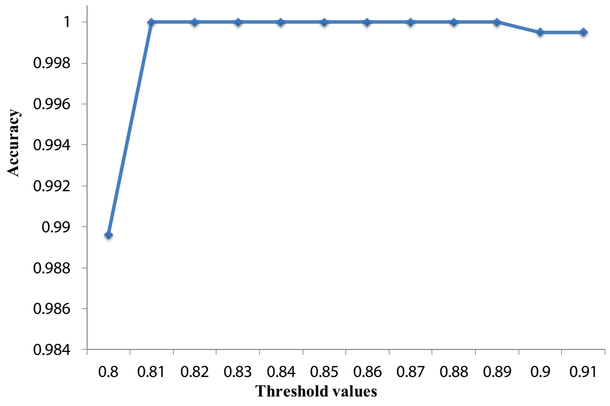


Fig. 7. Performance analysis on CAIDA DDoS dataset.

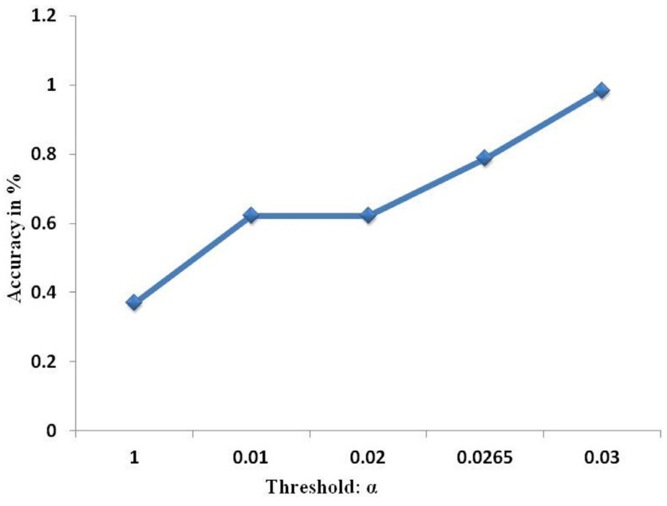


Fig. 8. Performance analysis on TUIDS dataset.

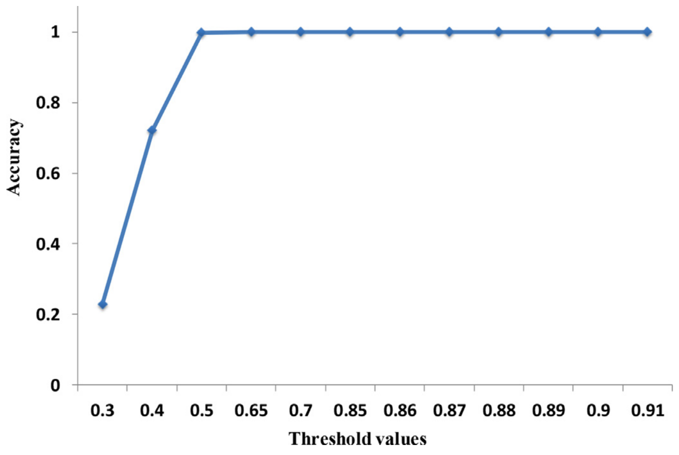


Fig. 9. Performance analysis on DARPA dataset.

5.1.1. Result analysis

Experimental results are plotted in three graphs for the three datasets, viz., CAIDA, TUIDS and DARPA. As shown in Fig. 7, in CAIDA dataset we observe that detection rate is very high, i.e., 100% with low false positive rates for different values of α within the range 0.81–0.89. Fig. 8 shows detection rate upto 99% for the TUIDS dataset and finally we observe 100% detection accuracy for different values of α on the DARPA dataset as shown in Fig. 9. Though the proposed method gives high detection accuracy and low false positive rates for the CAIDA and DARPA datasets, it shows relatively low detection rate and high false positive rate for the TUIDS dataset.

Our method detects DDoS attacks based on the deviation of the attack profile from the normal profile. The method detects attacks when the deviation is greater than a user defined threshold value α. In our experiment, we use different values of α to detect attacks and we observe that the value of α should be in the range of 0.026–0.820 for different datasets.

Table 4
Comparison with other methods on CAIDA dataset.

Method	DR	FPR	FNR
NFBoost [25]	97.2%	8.1%	NA
SNMP-based method [26]	99.4%	1.80%	0.60%
Proposed method	99.95%	0%	0.004%
DR: Detection Rate, FPR: False Positive Rate			
FNR: False Negative Rate, NA: Not Available			

Table 5
Comparison with other methods on DARPA 2000 dataset.

Method	DR	FPR	FNR
RBFBoost [27]	99.4%	3.7%	NA
Optimized traffic matrix [28]	98.7%	NA	NA
Chaos theory [29]	94%	0.45%	NA
Proposed method	100% (max)	0%	0.18%

5.1.2. Detection accuracy using dynamic threshold

Our detection method performs DDoS attack detection using both static and dynamic thresholding approach. The static thresholding-based detection is more effective in known type of DDoS attack detection. In contrast, the dynamic thresholding-based approach is more suitable for unseen DDoS attack type detection, in addition to known types. It does not require detection threshold as user input. In our experiment, a dynamic threshold is computed from the normal profile itself. To compute the dynamic threshold, we update the normal profile incrementally. The profile stores normal traffic statistics such as maximum correlation (Mx_c), mean correlation (Me_c) and minimum (Mn_c) correlation values of the normal samples. A test sample S_t is classified either as normal (N) or as an attack (A) using the following equation.

$$S_t = \begin{cases} A & \text{if } (Mx_c - Co_c) > \theta > (Co_c - Mn_c) \\ N & \text{if } (Mx_c - Co_c) > \theta < (Co_c - Mn_c) \end{cases}$$

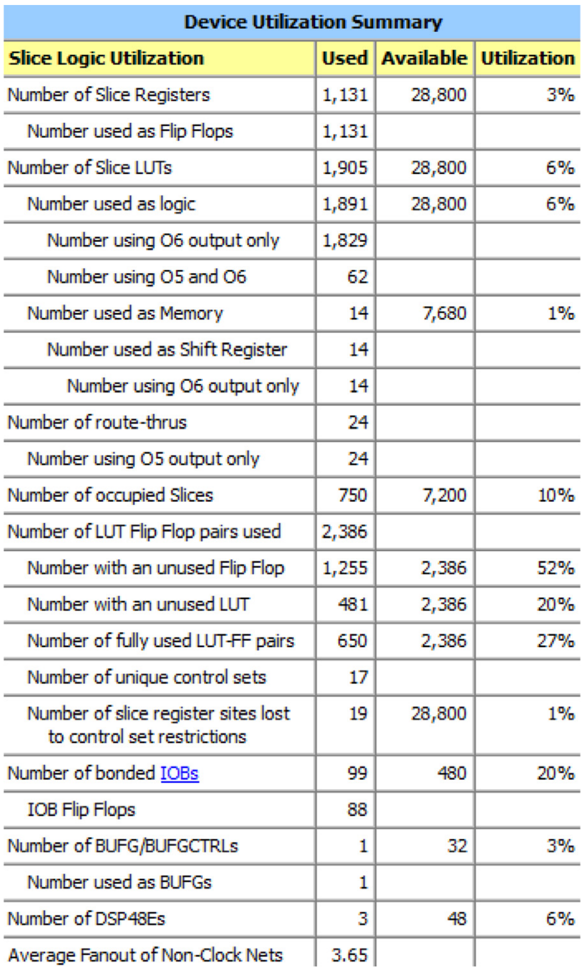
where, $\theta = (Me_c - Co_c)$, Co_c is the correlation value between mean of all normal samples and the test sample S_t. Since, we compare a test sample with the mean of normal samples, so we compute the detection threshold from the mean sample only. If the correlation between the test sample and the mean of normal samples is very close then the method detects the sample as normal. Otherwise, the sample is marked as attack. The proposed detection method with dynamic thresholding gives 99% detection accuracy on CAIDA dataset and 100% accuracy on DARPA dataset, the method yields 100% detection accuracy.

5.1.3. Comparison with existing methods

The proposed DDoS attack detection method is compared with some existing methods as shown in Tables 4 and 5 for CAIDA and DARPA 2000 datasets, respectively. From the comparison, we observed that the proposed DDoS attack detection method yields high detection accuracy on CAIDA as well as DARPA 2000 datasets.

5.2. Resource requirements and throughput of the detection module

The attack detection module of the proposed framework is implemented on a Xilinx Virtex-5 FPGA using the Xilinx ISE tools. The logic for the detection module is specified in VHDL and



Device Utilization Summary			
Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1,131	28,800	3%
Number used as Flip Flops	1,131		
Number of Slice LUTs	1,905	28,800	6%
Number used as logic	1,891	28,800	6%
Number using O6 output only	1,829		
Number using O5 and O6	62		
Number used as Memory	14	7,680	1%
Number used as Shift Register	14		
Number using O6 output only	14		
Number of route-thrus	24		
Number using O5 output only	24		
Number of occupied Slices	750	7,200	10%
Number of LUT Flip Flop pairs used	2,386		
Number with an unused Flip Flop	1,255	2,386	52%
Number with an unused LUT	481	2,386	20%
Number of fully used LUT-FF pairs	650	2,386	27%
Number of unique control sets	17		
Number of slice register sites lost to control set restrictions	19	28,800	1%
Number of bonded IOBs	99	480	20%
IOB Flip Flops	88		
Number of BUFG/BUFGCTRLs	1	32	3%
Number used as BUFGs	1		
Number of DSP48Es	3	48	6%
Average Fanout of Non-Clock Nets	3.65		

Fig. 10. Device utilization statistics for the implemented detection module on Virtex-5 FPGA.

synthesized. The synthesized design is placed and routed using the Xilinx Place&Route tools. The post Place & Route results for the detection module implementation are discussed below.

Fig. 10 depicts the post Place&Route utilization statistics of the implementation on the target FPGA. The implementation occupies 750 Slices, which is only 10% of the available Slices on the target device. Slice registers and Slice LUTs utilizations are 3% and 6% of the available resources, respectively. Moreover, the implementation does not require any Block RAM available in FPGAs. Therefore, the implementation can be considered as a compact solution for DDoS attack detection.

In Virtex-5 FPGAs, DSP Slices are used to implement fast multiplication. The three 11X11 multipliers in the datapath of the design are implemented using the three DSP48Es utilized in the implementation, depicted in Fig. 10.

According to the Place&Route report, the best achievable minimum clock period for the implementation is 8.42 ns, on the target FPGA. Which translates to the maximum achievable frequency of 118 MHz. Since the implementation is capable of classifying a new traffic instance in every 42 clock cycles, it provides throughput of 1 traffic instance per 354 ns. In the proposed solution, each traffic instance is created from a 1 s time window. Therefore, the implemented DDoS attack detection module easily fulfills the real time objectives of the proposed solution and also leaves significant amount of time for traffic capture and pre-processing.

The software implementation of the proposed method on Matlab requires 296 µs on average to classify a traffic sample as attack or normal, when three network traffic features are used. Therefore, the FPGA based implementation of the proposed DDoS attack detection method is able to achieve better speed up during attack detection (836 times faster) compared to the software based implementation.

The implementation model of the detection module presented in Fig. 5 is sequential. From cycle 6 to cycle 12, only the cordic component is used and similarly from cycle 17 to cycle 38, only the divider component is used. During these periods, other components of the design can be used to perform the operations for the next iteration. Therefore, pipelined implementation of the detection module can be considered using more resources for further performance improvement. However, a pipelined implementation of the detection module may not improve the overall throughput of the DDoS attack detection framework. This is due to the fact that the other components of the framework, such as the pre-processor, are implemented in software and are much slower than the hardware-based detection module. As suggested by the experimental results, the implemented detection module requires only 354 ns to detect DDoS attacks in a 1 s traffic window, which is negligible compared to the window size and the time required by the software components.

Fig. 11 depicts a simulation of the detection module using Xilinx Isim simulator. The waveforms demonstrate classification of two traffic instances by the detection module. The threshold, normal (att1-att3), and traffic (att1-att3) waveforms represent the inputs to the detection module. The simulation assumes a 10 ns clock period. Out of the two traffic instances, the first is classified as a normal instance (is_attack signal is low) at 655 ns of the simulation. The second instance is classified as an attack (is_attack signal is high) at 1075 ns of the simulation, after 42 cycles. When the classification result is available at the output port, the *result_valid* signal is raised for each of instances. The

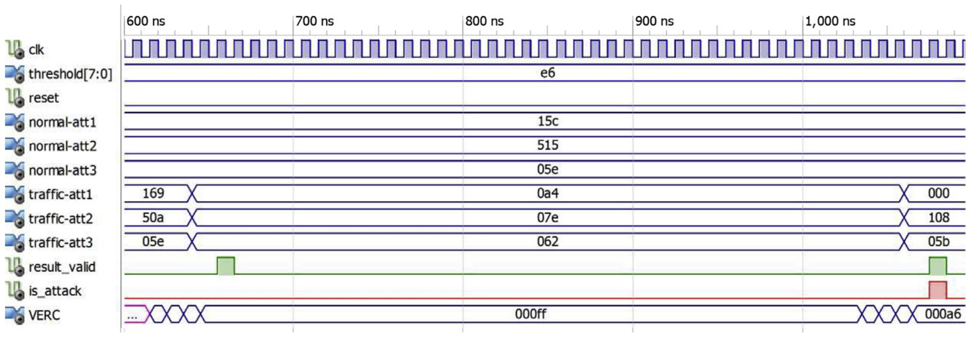


Fig. 11. Simulation waveforms demonstrating the operation of the DDoS attack detection module.

computed $NaHiD_{VERC}$ value for the traffic instance by the detection module is also depicted.

6. Conclusion

The state-of-the-art correlation measures, such as Pearson, Spearman, and Kendall's are not sensitive enough to detect DDoS attacks with high detection accuracy. With reduced number of features, these correlation measures do not perform well. This paper presents an effective correlation measure which is sensitive and capable of handling both shifting and scaling correlations between a pair of samples.

The proposed $NaHiD_{VERC}$ dynamically maintains a normal traffic profile and computes its correlation value with the incoming traffic sample during operation. An attack alarm is generated when the computed correlation value is smaller than a user defined threshold.

$NaHiD_{VERC}$ is implemented on software as well as on hardware using FPGA. It is able to achieve an attack detection accuracy of 100% over benchmark datasets. The FPGA implementation of the detection method imposes a minimal footprint on the available FPGA resources and requires 354ns to classify a traffic sample as attack. It also achieves better speed up during attack detection as compared to the software-based implementation.

However, at present $NaHiD_{VERC}$ considers the DDoS detection as a 2-class problem. We are working towards *n*-class DDoS detection by considering additional types of DDoS attacks as suggested by Mirkovic [30]. Work is going on to extend the work towards development of a distributed defense solution for detection crossfire attack in less time.

References

[1] Akami, Prolexic Quarterly Global ddos Attack Report q1 20, Technical Report, Prolexic, 2014.
[2] H.A. Ahmed, P. Mahanta, D.K. Bhattacharyya, J.K. Kalita, Shifting-and-scaling correlation based biclustering algorithm, IEEE/ACM Trans. Comput. Biol. Bioinf. (6) (2014) 1239–1252.
[3] N. Hoque, D.K. Bhattacharyya, J.K. Kalita, Botnet in ddos attacks: trends and challenges, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2242–2270.
[4] J. Wang, I.C. Paschalidis, Statistical traffic anomaly detection in time-varying communication networks, IEEE Trans. Control Netw. Syst. 2 (2) (2015) 100–111.
[5] N. Hoque, D.K. Bhattacharyya, J.K. Kalita, Ffsc: a novel measure for low-rate and high-rate ddos attack detection using multivariate data analysis, Secur. Commun. Netw. 9 (3) (2016).
[6] N. Hoque, D. Bhattacharyya, J. Kalita, Mifs-nd: a mutual information-based feature selection method, Expert Syst. Appl. 41 (14) (2014) 6371–6385.
[7] N. Hoque, H. Ahmed, D. Bhattacharyya, J. Kalita, A fuzzy mutual information-based feature selection method for classification, Fuzzy Inf. Eng. 8 (3) (2016) 355–384.
[8] W. Wei, F. Chen, Y. Xia, G. Jin, A rank correlation based detection against distributed reflection DoS attacks, Commun. Lett. IEEE 17 (1) (2013) 173–175.
[9] A. Kulkarni, S. Bush, Detecting distributed denial-of-service attacks using kolmogorov complexity metrics, J. Netw. Syst. Manage. 14 (1) (2006) 69–80.
[10] X. Ma, Y. Chen, Ddos detection method based on chaos analysis of network traffic entropy, Commun. Lett. IEEE 18 (1) (2014) 114–117.
[11] Z. Tan, A. Jamdagni, X. He, P. Nanda, R.P. Liu, A system for denial-of-service attack detection based on multivariate correlation analysis, Parallel Distrib. Syst. IEEE Trans. 25 (2) (2014) 447–456.
[12] P. Ning, S. Jajodia, Intrusion detection techniques, The Internet Encyclopedia (2003).
[13] N. Ye, S.M. Emran, Q. Chen, S. Vilbert, Multivariate statistical analysis of audit trails for host-based intrusion detection, Comput. IEEE Trans. 51 (7) (2002) 810–820.
[14] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang, A Novel Anomaly Detection Scheme Based on Principal Component Classifier, Technical Report, DTIC Document, 2003.
[15] T. Ahmed, M. Coates, A. Lakhina, Multivariate online anomaly detection using kernel recursive least squares, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, IEEE, 2007, pp. 625–633.
[16] L. Feinstein, D. Schnackenberg, R. Balupari, D. Kindred, Statistical approaches to ddos attack detection and response, in: DARPA Information Survivability Conference and Exposition, 2003. Proceedings, 1, IEEE, 2003, pp. 303–314.
[17] J. Santiago-Paz, D. Torres-Roman, P. Velarde-Alvarado, Detecting anomalies in network traffic using entropy and mahalanobis distance, in: Electrical Communications and Computers (CONIELECOMP), 2012 22nd International Conference on, IEEE, 2012, pp. 86–91.
[18] G. No, I. Ra, An efficient and reliable ddos attack detection using a fast entropy computation method, in: Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on, IEEE, 2009, pp. 1223–1228.
[19] H. Abdi, The kendall rank correlation coefficient, in: Encyclopedia of Measurement and Statistics, Sage, Thousand Oaks, CA, 2007, pp. 508–510.
[20] L. Yu, H. Liu, Feature selection for high-dimensional data: A fast correlation-based filter solution, in: ICML, 3, 2003, pp. 856–863.
[21] R.W. Swiniarski, A. Skowron, Rough set methods in feature selection and recognition, Pattern Recognit. Lett. 24 (6) (2003) 833–849.
[22] W. Lee, D. Xiang, Information-theoretic measures for anomaly detection, in: Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on, IEEE, 2001, pp. 130–143.
[23] P. Hick, E. Aben, K. Claffy, J. Polterock, The caida ddos attack 2007 dataset, 2007.
[24] P. Gogoi, M.H. Bhuyan, D. Bhattacharyya, J.K. Kalita, Packet and flow based network intrusion dataset, in: Contemporary Computing, Springer, 2012, pp. 322–334.
[25] P.A.R. Kumar, S. Selvakumar, Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems, Comput. Commun. 36 (3) (2013) 303–319.
[26] J. Yu, H. Lee, M.-S. Kim, D. Park, Traffic flooding attack detection with snmp mib using svm, Comput. Commun. 31 (17) (2008) 4212–4219.
[27] P.A.R. Kumar, S. Selvakumar, Distributed denial of service attack detection using an ensemble of neural classifier, Comput. Commun. 34 (11) (2011) 1328–1341.
[28] S.M. Lee, D.S. Kim, J.H. Lee, J.S. Park, Detection of ddos attacks using optimized traffic matrix, Comput. Math. Appl. 63 (2) (2012) 501–510.
[29] A. Chonka, J. Singh, W. Zhou, Chaos theory based detection against network mimicking ddos attacks, IEEE Commun. Lett. 13 (9) (2009) 717–719.
[30] J. Mirkovic, P. Reiher, A taxonomy of ddos attack and ddos defense mechanisms, ACM SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53.