Indian Journal of Science and Technology, Vol 9(27), DOI: 10.17485/ijst/2016/v9i27/87872, July 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Anomaly based Real Time Prevention of under Rated App-DDOS Attacks on Web: An Experiential Metrics based Machine Learning Approach

K. Munivara Prasad[1]*, A. Rama Mohan Reddy[2] and K. Venugopal Rao[3]

[1]Department of CSE, JNTUH, Hyderabad - 500085, Telangana, India; prasadkmv27@gmail.com
[2]Department of CSE, SVUCE, SV University, Tirupati - 517502, Andra Pradesh, India; ambatiramamohanreddy@gmail.com
[3]Department of CSE, GNITS, Hyderabad - 500008, Telangana, India; kvgrao1234@gmail.com

Abstract

To devise an Anomaly based Real Time Prevention (ARTP) of under rated App-DDOS attacks on Web for achieving fast and early detection. **Method:** We proposed a model based on machine learning approach that used to achieve the fast and early detection of the App-DDOS by multitude request flood. The proposed model ARTP is focused on defining set of metrics called "Re-quest chain length, request chain context, ratio of packet types, ratio of packet count, route context, router chain context and ratio of request intervals. The key factor of the proposal is unlike many of the bench marking models, which are considering requests or sessions as input to discover the anomalies, it considers set of requests are sessions in a time frame discovered to identify the anomalies of the metrics proposed. The experiments were carried out on bench marking LLDOS dataset and the performance analysis was done by the statistical analysis of the metrics like precision, recall, sensitivity and specificity. The process over-head also assessed in order to estimate the scalability and robustness of the proposal. **Findings:** The proposed model is highly significant in App-DDOS attack detection to adopt by current scenario of web applications with crowded requests that is phenomenally magnified to petabytes that compared to the past web request load in gigabytes.

Keywords: APP-DDoS, ARTP, Distributed Denial of Service, DDoS Atacks, HTTP Flooding, Intrusion Detection.

1. Introduction

Cyber malfunctioning activities from compromised users is a burning and serious act towards downgrading the computer communication, in particular of computer networks. One of such serious activity is Distributed Denial of Service that attacks web based networks, such that the potential web users unable to get the services from DDOS compromised web applications. The strategy of DDOS attack is that the host server of the web application is intentionally occupied by multiple sessions of multiple cooperative sources, such that no other user able to gain the access to that host server. In order to this, the attacker sends request packet flood of various types like SYN flood, UDP flood. The detection of such floods is very sensitive since the differentiation between user load and flood. This due to often user load resembles like flood, which essentially should not deny by the server. The recent familiar DDOS attack victims are explored in[1,2] and successful attack mitigating strategies explored in[3].

Among the existing attacks types, the most simple and effective way of App-DDoS attacks is utilizing the HTTP Flood to launch attack by requesting home page of the victim website repeatedly. In this paper our detecting schemes consider the App-DDoS attacks as anomaly browsing behavior.

In recent years, HTTP flood is one of DDOS attack observed. The HTTP flood is formed due to the abused HTTP requests, which generates request packet flood to occupy the target server resources[4,5]. The payloads observed at this flood such that the target servers unable to

*Author for correspondence

differentiate from the payload observed for crowded HTTP requests. This is due to the distributed strategy adopted by attacker that generates flood under multiple sessions.

The traditional signature based IDS tools[6] their inability to identify this multitude request flood, since these tools identifies the attacks based on the attack signatures, which is not practical in the case of request flood attack since no such signatures available under HTTP. Even the anomaly based IDS tools[6] also not compatible to identify these multitude request flood, since the features used to train the Anomaly based IDS[7-10] are compatible to assess the attack state of each request, but request flood is a stream of requests. The traditional strategy to prevent the App-DDOS attack is controlling bandwidth usage. The strategy of controlling bandwidth usage[11] is not optimal, since the desired bandwidth is always proportionate to the payload[12] Majority of time this controlling bandwidth usage would affect the performance of the server response to crowded users. In a gist often this strategy effects even the normal crowded traffic.

Henceforth it is obvious to confirm the vast research scope towards defining novel detection strategies to detect and prevent App-DDOS attacks[13].

This article aimed to define a machine learning strategy[14,15] to detect and prevent the App-DDOS attack. The main contributions of the paper aimed to identify the multitude request floods based on the stream of requests observed in time interval[16] rather in a session. A set of attributes defined to identify the state of a request stream is flood or fair. The machine learning[17] strategy called Anomaly based Real Time Prevention[18] of under rated App-DDOS attacks on Web that enables fast search is devised.

There are few HTTP flood[19] related DDoS defensing strategies can be found the recent literature, which are mostly relied on application-layer information. DDoSshield[20] is one, which defense the HTTP Flood by analyzing the session arrival time intervals, request arrival time intervals. HTTP Flood Defensing by Page access behavior [21] is another significant model that analyzes the information size, browsing time and their correlation[22]. Since these two models are inadequate since the attackers involves botnet[23] in order to control the packet-sending rates. In this context a CAPTCHA based probabilistic authentication method[24] was devised that overloads the user role due to the task of solving graphic puzzles[25-27], this often reflects as denial of service if users aggravated.

The contribution from[28] utilizing the Hidden semi-Markov Model (HsMM)[29,30] that uses pattern of request order to notify the normal user access behavior. Further the thresholds obtained from HSMM used as a scale to assess the state of incoming users. This model ends with frequent false alarms, since the pattern of requests is not a considerable metric as the request pattern can change with high frequencies due to the divergent browsing contexts of the legitimate users. As evidence to this argument, the incident where users can pass URLs directly to relevant request, may click the external web links or may use combination of browsers, which may leads to false alarms.

The observations from the review of these existing models evincing that almost all of these are having different constraints such as expert's involvement, training on static data and limited metrics. But all of these are limited to assess session based. In practice a user can adapt multiple sessions to perform set of requests in order of sequence or parallel. In order to this here we propose an evolutionary computational strategy to prevent HTTP Flood based DDOS attack. The objective of the proposal is to assess the state of HTTP transaction is flood or not, which is done by multiple metrics extracted from an absolute time interval rather session. In order to magnify the search towards state of proposed metrics.

The proposed ARTP technique aims at assessing similarity of a transaction with Fair and as well as flood data that given for training. Unlike existing benchmarking approaches, the proposed model is extracting the features from the request stream observed in an absolute time interval rather from the user sessions. The features (see sec 3.1) that adopted in proposed model also unique under DDOS attack context.

The rest of this paper is organized as follows. In section 2 explores the model ARTP proposed here in this paper. Experimental study and performance analysis of the ARTP is done in section 3 that followed by section 4, which is concluding the article.

2. Anomaly based Real Time Prevention of under Rated DDOS by HTTP Flood Attacks on Web: An Experiential Metrics based Machine Learning Approach

2.1 Learning Dataset Preprocessing

Let CS be the cached user sessions $CS = \{s_1, s_2, ...s_{|C|}\}$ and each session is set of transactions given for

Training, such that each request is said to be transaction $\{t \exists t \in s_i \wedge s_i \in CS\}$ labeled as N (normal) or D (DDOS attack). The cached transactions CS is segregated into CS_N and CS_D, that contains requests labeled as N (normal) and labeled as D (DDOS attack) respectively. Further the heuristics of the metrics explored (in section 3.2) will be assessed on these datasets.

2.1.1 Discovering Time Frame Length

For each dataset CS (which is the aggregation of CS_N and CS_N), order the sessions in ascending order of their initiated time.

Let $SB = \{sb(s_1 \exists s_1 \in SC), sb(s_2 \exists s_2 \in SC), ...sb(s_{|SC|} \exists s_{|SC|} \in SC)\}$ as the set that represents the session begin time of all sessions belongs to SC.

Let $SE = \{se(s_1 \exists s_1 \in SC), se(s_2 \exists s_2 \in SC), ...se(s_{|SC|} \exists s_{|SC|} \in SC)\}$ as the set that represents the session end time of all sessions belongs to SC.

Let $SL = \{sl(s_1 \exists s_1 \in SC), sl(s_2 \exists s_2 \in SC), ...sl(s_{|SC|} \exists s_{|SC|} \in SC)\}$ as the set that represents the session life time of all sessions belongs to SC.

Session life time of a session $\{s_i \exists s_i \in CS\}$ is calculated as follows:

$$sl(s_i \exists s_i \in CS) = se(s_i \exists s_i \in CS) - sb(s_i \exists s_i \in CS)$$

Find session begin time absolute deviation[34] of SB

$$sbtAD = \frac{\sqrt{\sum_{i=1}^{|SB|} \left(\langle SB \rangle - sb(s_i \exists s_i \in SB)\right)^2}}{|SB|}$$

Here in the above equation,$|SB|$ is the size of SB and $\langle SB \rangle$ is the average of SB, Find session end time absolute deviation of SE

$$setAD = \frac{\sqrt{\sum_{i=1}^{|SE_N|} \left(\langle SE \rangle - se(s_i \exists s_i \in s_i \in SE)\right)^2}}{|SE|}$$

Here in the above equation, $|SE|$ is the size of SE and $\langle SE \rangle$ is the average of SE

Then the absolute time frame atf can be measured as follows

$$atf = \left(\langle SE \rangle + seAD\right) - \left(\langle SB \rangle + sbAD\right)$$

2.1.1.1 Cluster the sessions by rmsd(SB_N) and rmsd(SE_N) distance

Finding K (count of centroids) value as follows:

Let K be the set of centroids and move session with least session begin time to the K,

For each session $\{s_i \exists s_i \in SB_N \wedge i = 2,3,...,|SB_N|\}$

Begin

$flag = true$

For each session $\{s_j \exists s_j \in K\}$

Begin

If $\left(\sqrt{(sb(s_i) - sb(s_j))^2} < sbtAD \| \sqrt{(se(s_i) - se(s_j))^2} < setAD\right)$ then

Begin

$flag = false$

End

End

if $(flag)$ then Begin

$K \leftarrow s_i$

End

End

End

Then apply K-Means[35] to find number clusters with sessions in approximately similar time frames

Let $C = \{c_1, c_2, ..., c_K\}$ be the set of clusters of size K

Then for each cluster $\{c_i \exists c_i \in C \wedge i = 1,2,3,....K\}$, find the time frame as the elapsed time between least session begin time and max session end time as follows

Foe each $\{c_i \exists c_i \in C \wedge i = 1,2,3,....K\}$ Begin

Let $SB_N(c_i) = \{sb_1, sb_2, ...sb_{|c_i|}\}$ be the ascending ordered set of session begin times of the sessions belongs to cluster c_i

Let $SE_N(c_i) = \{se_1, se_2, ...se_{|c_i|}\}$ be the descending ordered set of session end times of the sessions belongs to cluster c_i

Then the time frame $tf(c_i)$ of the cluster c_i is measured as follows:

$$tf(c_i) = \sqrt{(se_1 - sb_1)^2}$$

Then find the average of time frames length observed from all the clusters as follows

$$\langle tf(C) \rangle = \frac{\sum_{i=1}^{K} tf(c_i)}{K}$$

Further find Time Frame Absolute Deviation $tfAD$ observed from all the clusters as follows

$$tfAD = \frac{\sqrt{\sum_{i=1}^{K} \left(\langle tf(C) \rangle - tf(c_i)\right)^2}}{K}$$

Then fix the time frame *tf* as the sum of average of time frames length and Time Frame Absolute Deviation as follows.

$$tf = \langle tf(C) \rangle + tfAD$$

2.2 Metrics and their Heuristics to be Identified from the Training Data

2.2.1 Request Chain Length (RCL)

From the given set of cached transactions for training that are labeled as *N* or *D*, average length of requests usually observed from the clients that is said to be possible length of requests in a given time frame to label as attack or normal. The RCL can be measured as follows

- Partition the chain of transactions in CS_N as $TS(CS_N) = \{ts_1, ts_2, ...ts_{|TS(CS_N)|}\}$ such that ts_i contains the transactions observed in the i^{th} span of time frame.
- Find the Average chain of transactions length observed in all time frame shares discovered from CS_N as follows:

$$\langle TS(CS_N) \rangle = \sum_{i=1}^{|TS(CS_N)|} \{| ts_i | \exists ts_i \in TS(CS_N)\}$$

//Here in the above equation $\langle TS(CS_N) \rangle$ is the mean of transactions count observed for all time frames $TS(CS_N)$ and $|ts_i|$ is the count of transactions observed in the span of i^{th} time frame.

- Find the request chain length absolute deviation observed in all time frame groups discovered from CS_N as follows

$$clAD = \frac{\sqrt{\sum_{i=1}^{|TS(CS_N)|} \left(\langle TS(CS_N) \rangle - (| ts_i | \exists ts_i \in TS(CS_N)\right)^2}}{| TS(CS_N) |}$$

- Fix the value for metric called request chain length as the sum of Average chain of transactions and RMSD of chain of transactions length.

$$rcl(CS_N) = \langle TS(CS_N) \rangle + clAD$$

2.2.2 Request Chain Context (Order of Requests)

From the given set of cached transactions for training that are labeled as normal or attack, we segregate the order of requests in a given time frame that are labeled as normal and attack. Then we estimate the scope of a request order is normal or attack prone.

For the cached transactions set CS_N, prepare request pair set $rps_N = \{p_1, p_2, ..., p_{|rps_N|}\}$ such that each two requests in sequence as pair p_i find local support $ls_{ts_j}(p_i)$ of each pair $(p_i \exists p_i \in rps_N \wedge i = 1,2,.....,| rps_N |)$, which is representing the number of occurrences of pair p_i in time frame ts_j. Find the global support $gs(p_i)$ of each pair $(p_i \exists p_i \in rps_N \wedge i = 1,2,.....,| rps_N |)$, which is indicating the number of occurrences of a pair in all chain of requests of the CS_N.

Find the request pair support absolute deviation of the local supports of the each pair from the global support as follows

For each pair $\{p_i \exists p_i \in rps_N \wedge i = 1,2,.....,| rps_N |\}$

$$rpsAD = \sqrt{\frac{\sum_{j=1}^{|TS(CS_N)|} \left(gs(p_i) - ls_{ts_j}(p_i)\right)^2}{| TS(CS_N) |}}$$

Then measure the request chain context $rcc(p_i)$ of each request pair p_i can be measured as follows

For each pair $\{p_i \exists p_i \in rps_N \wedge i = 1,2,.....,| rps_N |\}$

$$rcc(p_i) = \frac{\sum_{j=1}^{|TS(CS_N)|} ls_{ts_j}(p_i)}{| TS(CS_N) |} + rpsAD$$

2.2.3 Ratio of Packet Types (Packets in a Fixed Time Fame)

The ratio of packets of type observed in a given time frame under the chain of requests labeled as normal and under chain of requests labeled as attack. These assessed ratios further used to define the packet ratio threshold for each type such as udp, icmp and tcp-SYN under normal and attacked transactions.

2.2.3 .1 Finding local support of the divergent packet types (support of packet types in the individual span of time frames)

Let consider the different packet types as a set $PT = \{pt_1, pt_2, ...pt_{|PT|}\}$

For each time frame $\{ts_i \exists ts_i \in TS(CS_N)\}$ Begin

Let $P(ts_i) = \{p_1, p_2, ..., p_{|P(ts_i)|}\}$ be the set of packets observed in the span of time frame ts_i ^{ts_i}

For each packet type $\{pt_j \exists pt_j \in PT \wedge j = 1,2,.....,| PT |\}$ Begin

$$c_{ts_i}(pt_j) = \sum_{k=1}^{|P(ts_i)|} \{1 \exists t(p_k) \equiv pt_j\}$$

//Here $t(p_k)$ is function verifies the packet p_k type, $c_{ts_i}(pt_j)$ is number of packets of type pt_j observed in the span of time frame ts_i

$$ls_{ts_i}(pt_j) = \frac{c_{ts_i}(pt_j)}{|P(ts_i)|}$$

//local support $ls_{ts_i}(pt_j)$ of the packet type pt_j in the span of time frame ts_i

End

End

2.2.3.2 Finding Global Support (support of each packet type from the all cached transactions CS_N)

For each packet type $\{pt_j \exists pt_j \in PT \wedge j = 1,2,.....,|PT|\}$ Begin

$$c(pt_j) = 0$$

//Initializing the number of packets $c(pt_j)$ of type pt_j found in complete dataset

For each time frame $\{ts_i \exists ts_i \in TS(CS_N)\}$ Begin

Let $P(ts_i) = \{p_1, p_2, ..., p_{|P(ts_i)|}\}$ be the set of packets observed in the span of time frame ts_i

$$c(pt_j) = c(pt_j) + \sum_{k=1}^{|P(ts_i)|} \{1 \exists t(p_k) \equiv pt_j\}$$

End

$$gs(pt_j) = \frac{c(pt_j)}{\sum_{k=1}^{|TS(CS_N)|} |P(ts_k)|}$$

// $gs(pt_j)$ is the global support of the packet type pt_j, $\sum_{k=1}^{|TS(CS_N)|} |P(ts_k)|$ is to find the total number of packets in all time frame spans. $|P(ts_k)|$ is the total number of packets available in time frame span ts_k.

2.2.3.3 Packet Type Support absolute deviation between local support and global support of the packet types

For each packet type $\{pt_j \exists pt_j \in PT \wedge j = 1,2,.....,|PT|\}$ Begin

$$ptsAD(pt_j) = \frac{\sqrt{\sum_{i=1}^{|TS(CS_N)|} \left(gs(pt_j) - ls_{ts_i}(pt_j)\right)^2}}{|TS(CS_N)|}$$

2.2.3.4 Measuring Ratio of packet type (rpt)

Then the ratio of packet type $rpt(pt_j)$ of packet type pt_j can be measured as follows

$$rpt(pt_j) = \frac{\sum_{i=1}^{|TS(CS_N)|} ls_{ts_i}(pt_j)}{|TS(CS_N)|} + ptsAD(pt_j)$$

2.2.4 Ratio of Packet Count

For given set of cached transactions for training, the numbers of packets received for each time frame are used to identify the ratio of packet count observed under chain of requests labeled as normal and also for chain requests labeled as attack.

2.2.4.1 Measuring ratio of packets for time frame

For each $\{ts_i \exists ts_i \in TS(CS_N) \wedge i = 1,2,....|TS(CS_N)|\}$ Begin

$$rp(ts_i) = \sum_{i=1}^{|TS(CS_N)|} \frac{|P(ts_i)|}{\sum_{k=1}^{|TS(CS_N)|} |P(ts_k)|}$$

// $rp(ts_i)$ is the ratio of packets for ts_i, $|P(ts_i)|$ is total number of packets in time frame. ts_i and $\sum_{k=1}^{|TS(CS_N)|} |P(ts_k)|$ is the total number of packets of all time frames.

2.2.4.1.1 Time Frame Level Packets Support absolute deviation

$$tflpsAD = \frac{\sqrt{\sum_{j=1}^{|TS(CS_N)|} \left(1 - rp(ts_j)\right)^2}}{|TS(CS_N)|}$$

2.2.4.1.2 Measuring Ratio of packet count

$$rpc(TS(CS_N)) = \frac{\sum_{i=1}^{|TS(CS_N)|} rp(ts_i)}{|TS(CS_N)|} + tflpsAD$$

2.2.5 Router Context

From the given set of cached transactions for training that are labeled as normal or attack, we estimate the scope of each router involved.

For the cached transactions set CS_N, let $rs_N = \{r_1, r_2, ..., r_{|rs_N|}\}$ be the set of routers observed involved in request and response exchange. Find local support $ls_{ts_j}(r_i)$ of each router $(r_i \exists r_i \in rs_N \wedge i = 1,2,.....,|rs_N|)$, which is representing the number of occurrences of router r_i in time frame ts_j. Find the global support $gs(r_i)$ of each router $(r_i \exists r_i \in rs_N \wedge i = 1,2,.....,|rs_N|)$, which is indicating the number of occurrences of a router in all spans of time-frames $\{ts_i \exists ts_i \in TS(CS_N) \wedge i = 1,2,.....|TS(CS_N)|\}$.

Find the Local Support Absolute Deviation of the each router from the global support as follows

For each pair $\{r_i \exists r_i \in rs_N \wedge i = 1,2,.....,|rs_N|\}$

$$lsAD(r_i) = \frac{\sqrt{\sum_{j=1}^{|TS(CS_N)|} \left(gs(r_i) - ls_{ts_j}(r_i)\right)^2}}{|TS(CS_N)|}$$

Then measure the router context $rc(r_i)$ of each router r_i can be measured as follows

For each pair $\{r_i \exists r_i \in rs_N \wedge i = 1,2,.....,|rs_N|\}$

$$rc(r_i) = \frac{\sum_{j=1}^{|TS(CS_N)|} ls_{ts_j}(r_i)}{|TS(CS_N)|} + lsAD(r_i)$$

2.2.6 Router Chain Context

From the given set of cached transactions for training that are labeled as normal or attack, we segregate the order of routers in a given time frame that are labeled as normal and attack. Then we estimate the scope of a request order is normal or attack prone.

For the cached transactions set CS_N, prepare router pair set $hps_N = \{hp_1, hp_2, ..., hp_{|hps_N|}\}$ such that each two requests in sequence as pair hp_i find local support $ls_{ts_j}(hp_i)$ of each pair $(hp_i \exists hp_i \in hps_N \wedge i = 1,2,.....,|hps_N|)$, which is representing the number of occurrences of pair hp_i in time frame ts_j. Find the global support $gs(hp_i)$ of each router pair $(hp_i \exists hp_i \in hps_N \wedge i = 1,2,.....,|hps_N|)$, which is indicating the number of occurrences of a router pair in all chain of routers of the CS_N.

Find the local support absolute deviation of the each router pair from the global support as follows

For each pair $\{hp_i \exists hp_i \in hps_N \wedge i = 1,2,.....,|hps_N|\}$

$$lsAD(hp_i) = \frac{\sqrt{\sum_{j=1}^{|TS(CS_N)|} \left(gs(hp_i) - ls_{ts_j}(hp_i)\right)^2}}{|TS(CS_N)|}$$

Then measure the router chain context $hcc(hp_i)$ of each router pair hp_i can be measured as follows

For each pair $\{hp_i \exists hp_i \in hps_N \wedge i = 1,2,.....,|hps_N|\}$

$$hcc(hp_i) = \frac{\sum_{j=1}^{|TS(CS_N)|} ls_{ts_j}(hp_i)}{|TS(CS_N)|} + lsAD(hp_i)$$

2.2.7 Ratio of Intervals between Requests

For a given set of cached transactions for training, the elapsed time between any two requests in sequence of same session will be considered as access time. The access time observer for all pairs of requests in sequence of same session of an each time frame will be considered to assess the ratio of intervals between requests labeled as normal and labeled as attack.

Find the local mean $lm(ts_i)$ of the intervals between requests in sequence of each time frame $\{ts_i \exists ts_i \in TS(CS_N)\}$ as follows:

For each time frame $\{ts_k \exists ts_k \in TS(CS_N) \wedge k = 1,2,3,.....|TS(CS_N)|\}$

$$lm(ts_k) = \frac{\sum_{j=1}^{|I(ts_k)|} i_j}{|I(ts_k)|} \quad // \; I(ts_k) \text{ set of intervals found between}$$

the pair requests in sequence from ts_k

Find the global mean $gm(CS_N)$ of the intervals between requests in sequence of CS_N as follows.

$$gm(CS_N) = \frac{\sum_{j=1}^{|I(CS_N)|} i_j}{|I(CS_N)|} \quad // \; I(CS_N) \text{ set of intervals found}$$

between the pair requests in sequence from CS_N CS_N

Find the interval absolute deviation iAD as follows

$$iAD = \frac{\sqrt{\sum_{j=1}^{|TS(CS_N)|} (gm(CS_N) - lm(ts_j))^2}}{|TS(CS_N)|}$$

Then find the ratio of interval $ri(CS_N)$ as follows

$$ri(CS_N) = \frac{\sum_{i=1}^{|TS(CS_N)|} lm(ts_i)}{|TS(CS_N)|} + iAD$$

3. Experimental Results and Performance Analysis

The experiments were done in the context of assessing the detection accuracy and scalability, robustness and process complexity of the proposed ARTP.

3.1 Dataset

The LLDOS 2.0.2-scenario two[36,37] is used to generate the App-DDOS attack requests of aggregate time interval of 1800 seconds. Similarly the request said to be normal is also picked for the same aggregate interval time. The overall sessions observed in the given time are 1200 and 740 of App-DDOS attack and normal requests respectively. The metric values observed from the given data are listed below in Table 1.

The total data considered for experiments were partitioned into 70% and 30% for training and testing respectively. The detection accuracy is assessed using the statistical metrics[38] called precision, sensitivity (true positive rate), specificity (true negative rate) and accuracy. The total number of absolute time frames observed from aggregate time interval given is 666, which is measured as follows.

Table 1. The values for metrics devised observed from the experimental data

Metric	Value Observed	
	App-DDOS	Normal
Absolute Time Frame (see sec 3.1.1)	2700 milliseconds	2700 milliseconds
Request chain length (see sec 3.2.1)	176	124
Number of Request Chain Context Patterns (see sec 3.2.2)	25	14
Ratio of packet types (see sec 3.2.3)	0.8574899	0.568959
Ratio of Packet Count (see sec 3.2.4)	0.8373156	0.509015
Ratio of Router Context (see sec 3.2.5)	0. 396215	0.734201
Number of Router Chain Context Patterns (see sec 3.2.6)	6	4
Ratio of intervals between requests (see sec 3.2.7)	0.178284	0.226098

$$|atf| = \frac{ati}{atf}$$

// Here in the above equation, *ati* is aggregate time interval (1800*1000) in milliseconds considered for experiments and *atf* (2700 milliseconds) is absolute time frame discovered.

The 70% of the $|atf|$ (444 time frames) are used for training and the 30% time frames (222) are used for testing. Among 222 time frames used for testing, the 135 are of App-DDOS attack scenarios and rest are attack free scenarios.

3.2 Performance Analysis

The ARTP detection statistics are explored in Table 2. The precision indicates the attack predication capability, sensitivity observed from the experiments indicates the attack detection rate, specificity indicates the normal request scenario detection rate and Accuracy is the rate of detection accuracy for both normal and attack scenarios. The value observed for specificity is lesser than the value observed for sensitivity, which indicates that the possibility of an attack scenario identified as normal is much lesser than the possibility of a normal scenario identified as attack. The rate of identifying an attack as normal scenario is 0.126 (1-sensitivity) and the rate of identifying a normal scenario as attack is 0.184 (1-specificity), which is much greater than the earlier. Hence it is obvious to quote that the proposed ARTP is much sensitive towards attack detection than the normal.

In order to identify the process complexity, training was done at divergent count of time frames given for testing. The observed results indicating that the time

Table 2. Statistical performance metrics and observed values

TP	118
FP	16
TN	71
FN	17
Precision	0.880597
Sensitivity	0.874074
Specificity	0.816092
Accuracy	0.851351

complexity and space complexity observed for training is linear shown in Figure 1 and 2. The significance of metrics towards App-DDOS attack detection is also elevated, which is explored in Figure 3.

All of the training cases in Figure 1, the completion time are lesser than the expected completion shown in the error bars in Figure 1, which is an aggregate of the completion time for each time frame involved. Hence the process complexity is said to be linear. And the Figure 2 indicating the linearity in space usage towards training using divergent number of time frames given.

The metrics devised under ARTP are evenly significant towards attack detection accuracy is shown in Figure 3. More specific, the ratio of packet types is the metric, which is observed to be with fewer influences towards detection accuracy that compared to other metrics. The overall performance analysis is evincing that the

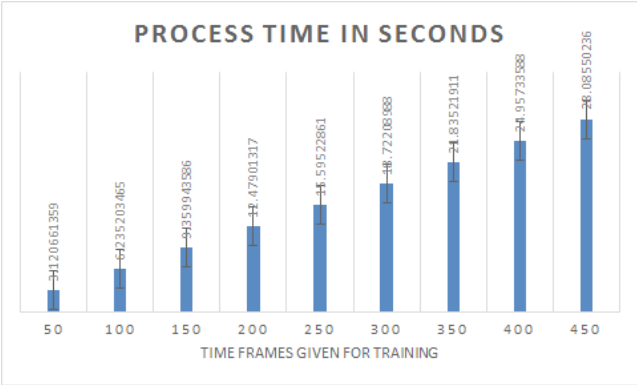

Figure 1. Process completion time for divergent number of time frames given for training.

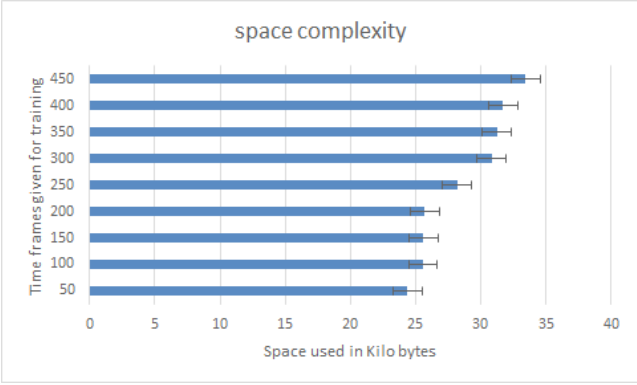

Figure 2. Space used towards training phase observed for divergent count of time frames.

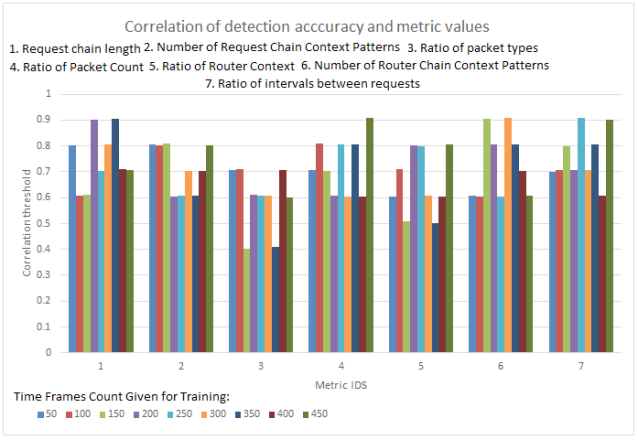

Figure 3. The correlation between metrics and detection accuracy observed.

metrics devised under ARTP are significant and magnified the accuracy of App-DDOS attack detection.

4. Conclusion

Machine learning strategy called Anomaly based Real Time Prevention (ARTP) of under rated App-DDOS attacks is devised here in this article. The overall contribution of the paper is in three levels. The initial contribution is define feature metrics to identify the request stream behavior is of attack intension or not. Unlike traditional approaches, the feature metrics were assessed on the stream of requests observed in an absolute time interval rather in a session. Further ARTP uses the threshold values observed for defined metrics to learn the behavior of request stream is a flood or not. The third contribution of the proposal is to test the performance of the ARTP on benchmarking LLDOS dataset. The devised ARTP amplified the detection accuracy with minimal process complexity and maximal speed. The exploration of the results concluding that the devised metrics are promising and significant to learn the request stream state from training records dataset. Further the threshold values learnt from training are used to estimate the request stream in an absolute time interval is an App-DDOS attack or not. ARTP is observed to be robust and is with minimal process complexity and maximal speed. Hence the model devised here in this paper is significantly minimized the computational overhead and retains the maximal prediction accuracy. In future the research can be driven in the direction of feature optimization and adopting a compatible evolutionary strategy towards learning the stream behavior from anomalies.

5. References

1. Udhayan J, Anitha R. Demystifying and rate limiting ICMP hosted DoS/DDOS flooding attacks with attack productivity analysis. IEEE International Conference on Advance Computing; 2009. p. 558-64.
2. Chun-Tao Xia X-HD-F-C. An algorithm of detecting and defending CC attack in real time. International Conference on Industrial Control and Electronics Engineering; 2012. p. 1804-6.
3. Lee SM. Distributed denial of service: Taxonomies of attacks, tools and counter measures. Proceedings of the International Workshop on Security in Parallel and Distributed Systems; San Francisco. 2004. p. 543-50.
4. Byers S, Rubin AD, Kormann D. Defending against an internetbased attack on physical world. ACM Transactions on Internet Technorogy. 2004; 239-54.
5. Estevez-Tapiador JM. Detection of web-based attacks through Markovian protocol parsing. 10th IEEE Symposium on Computers and Communications; 2005. p. 457-62.
6. Jyothsna VP. A review of anomaly based intrusion detection systems. International Journal of Computer Applications. 2013; 26-35.
7. Abraham A, Jain R, Thomas J, Han SY. D-SCIDS: Distributed soft computing intrusion detection system. J Network Computer. 2007; 30(1):81-98.
8. Sundaram A. An introduction to intrusion detection. The ACM Student Magazine. 1996; 2(4):3-7.
9. Li YX. An efficient intrusion detection system based on support vector machines and gradually feature removal method. Expert Systems with Applications. 2012; 424-30.
10. Mell RB. Intrusion detection systems. Intrusion Detection System. NIST Special Publication; 2001.
11. Chie Ishida YA. Forecast techniques for predicting increase or decrease of attacks using Bayesian inference. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing; Victria, Canada. 2005. 1088-92.
12. Cacheda RA. QoS requirements for multimedia services. Resource Management in Satellite Networks. Springer; 2007. p. 67-94.
13. Saravanan A, Irfan Ahmed MS, Sathya Bama S. Policy approval engine - A framework for securing web applications and web user. Indian Journal of Science and Technology. 2016 Jan; 9(4).DOI:10.17485/ijst/2016/v9i4/84341
14. Hassan MM. Current studies on intrusion detection system, genetic algorithm and fuzzy logic. International Journal of Distributed and Parallel Systems. 2013; 35-48.
15. Lane T. Machine learning techniques for the computer security. Purdue University; 2000.
16. Goodman NR. Statistical analysis based on a certain multivariate complex Gaussian distribution. Annals of Mathematical Statistics. 1963; 152-77.
17. Bhatti DG. Conceptual framework for soft computing based intrusion detection to reduce false positive rate. International Journal of Computer Applications. 2012; 44(13):1-3.
18. Bauer DS. NIDX - An expert system for real-time network intrusion detection. Proceedings of the Computer Networking Symposium; 1998. p. 98-106.
19. Abbasvand S, Nasser S, Hashemi S, Jamali S. Defense against SYN-flooding attacks by using game theory. Indian Journal of Science and Technology. 2014 Oct; 7(10):1618–24.
20. Ranjan S. DDoS shield: DDoS-resilient scheduling to counter application layer attacks. IEEE/ACM Trans Netw. 2009; 26-39.
21. Yatagai T. Detection of HTTP-GET flood attack based on analysis of page access behavior. Proceedings IEEE Pacific RIM Conference on Communications, Computers and Signal Processing; 2007. p. 232-5.
22. Sindhu SS. Decision tree based light weight intrusion detection using a wrapper approach. Expert Systems with Applications. 2012; 129-41.
23. Shevtekar A. Is it congestion or a DDoS attack? IEEE Commun Lett. 2009; 546-8.
24. Kandula S. Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds. Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation. 2005; 287-300.
25. Katar C. Combining multiple techniques for intrusion detection. Int J Comput Sci Network Security. 2006; 208–18.
26. KDD cup99. Available from: kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
27. Kennedy J. Particle swarm optimization. Encyclopedia of Machine Learning. 2010; 760-6.
28. Chen WH. Application of SVM and ANN for intrusion detection. Comput Oper Res. 2005; 32(10):2617–34.
29. Chen Y. Feature deduction and intrusion detection using flexible neural trees. 2nd IEEE International Symposium on Neural Networks. 2005; 32(10):2617-34.
30. Xie Y. A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. IEEE/ACM Trans Netw. 2009; 54-65.
31. Yang XS. Cuckoo search via Levy flights. World Congress on Nature and Biologically Inspired Computing. 2009; 210-4.
32. Stolfo WL. Data mining approaches for intrusion detection. Proceedings of the 7th USENIX Security Symposium; 1998.
33. Real R. The probabilistic basis of Jaccard's index of similarity. Systematic Biology. 1996; 380-5.
34. Leys CL. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. Journal of Experimental Social Psychology. 2013; 5(3):764-6.

35. Hartigan JA. Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. 1979; 100-8.
36. MIT, MI. Darpa intrusion detection evaluation. Available from: https://www.ll.mit.edu/ideval/data/1998data.html
37. MIT MI. 2000. Available from: https://www.ll.mit.edu/ideval/data/2000data.html
38. Powers DM. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. 23rd International Conference on Machine Learning; Pitsburg. 2006.