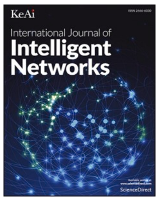
Contents lists available at ScienceDirect

International Journal of Intelligent Networks

journal homepage: www.keaipublishing.com/en/journals/international-journal-of-intelligent-networks



ARTP: Anomaly based real time prevention of Distributed Denial of Service attacks on the web using machine learning approach

P. Krishna Kishore [*], S. Ramamoorthy, V.N. Rajavarman

Department of Computer Science and Engineering, Dr. M.G.R Educational and Research Institute, Chennai, India

A R T I C L E I N F O

Keywords:
LLDoS data set
Denial of Service (DoS) attack
Distributed DoS (DDoS) attack
Detection of App-DDoS
Application layer DDoS (App-DDoS)

A B S T R A C T

Distributed Denial of Service (DDoS) attack is one of the most destructive internet network attacks, denying legitimate users access to resources and networks by maliciously blocking available computing resources. Intruders send a large number of packets to the network in order to create a crowding effect. Unlike a Denial of Service (DoS) attack, where a single compromised source generates all of the traffic, a Distributed Denial of Service (DDoS) attack generates traffic from multiple compromised nodes spread across multiple geographies. To address the challenges posed by the Distributed Denial of Service (DDoS) attack, several researchers proposed a variety of solutions for early detection and prevention of the attack. Effective solutions for the prevention and early detection of Distributed Denial of Service (DDoS) attacks, on the other hand, have yet to be developed, and the problem remains a prominent research focus area. This paper tries to present a novel and optimal solution for detecting Distributed Denial of Service (DDoS) attacks on internet networks more quickly and accurately. The proposed model is an anomaly-based real-time prevention model for web networks. The model is based on machine learning principles and can effectively counter new types of Distributed Denial of Service (DDoS) attacks. To demonstrate the efficiency, accuracy, model robustness, and relative of the proposed model, a simulation study was run on an LLDOS session log, and the results indicated that the model performed better than benchmark models found in the literature.

1. Introduction

With the increased use of the internet across industries and governments, the problems of failed communication are becoming more visible in computer networks. In particular, intruding into the network via compromised users is a common catastrophic threat encountering detection systems. Among the various types of DDoS attacks, the Application Layer DDoS (Application DDoS) attack, which uses HTTP-flood to send packet streams into the network, is quite common. Requests to access the web or application home page are frequently raised by this attack type. This chapter offers a solution for dealing with App-DDoS attacks in an efficient manner. HTTP-flood attacks are the result of repetitive HTTP request messages, which generate REQ messages that consume all of the host's resources [1,2]. The payload observed in this stream is in such a way that the host server is unable to differentiate it from instances of heavy flow from genuine users. Regardless of robust detection mechanisms, the host server cannot differentiate these attacks because the attackers use a distributed approach. Furthermore, if the attacker sends these request messages in multiple sessions rather than a single session, the identification process becomes more complicated.

Signature-based detection models [3] as well as anomaly-based detection models fail to detect HTTP-flood-based DDoS attacks. On the basis of signatures, signature-based approaches detect intrusion. However, no signatures are present in the case of HTTP-flood, so these IDS tools are unable to detect the attack. Though the attributes designed for training the detection system are compatible with evaluating the attack proneness of each request in the case of anomaly-based detection approaches, attack floods are a stream of requests. The majority of service providers are currently defending against the App-DDoS threat by limiting bandwidth usage. However, because the required bandwidth grows in proportion to the payload [4] limiting bandwidth usage is not an optimal solution [5]. The approach can have an impact on regular traffic flow as well as server performance in cases of crowding from genuine users. As a result, there is a strong need to develop an optimal defense strategy against App-DDoS threats that does not limit bandwidth usage.

* Corresponding author.
E-mail addresses: krishna.boinapalli@gmail.com (P. Krishna Kishore), srm24071959@yahoo.com (S. Ramamoorthy), nrajavarman2003@gmail.com (V.N. Rajavarman).

https://doi.org/10.1016/j.ijin.2022.12.001
Received 13 October 2022; Received in revised form 13 December 2022; Accepted 22 December 2022
Available online 27 December 2022

According to Cloudflare's study, ransom DDoS assaults climbed by about a third between 2020 and 2021 and by 75% in Q4 2021 [6]. Moreover, the manufacturing industry had the greatest application-layer DDoS attacks, up 641% quarter over quarter. Attackers tried to block individuals from accessing online resources of various commercial and non-profit organizations, according to Yandex and Qrator Labs [7]. Attackers are adopting increasingly complex strategies, such as multi-vector assaults, in their DDoS attacks. Generally, the majority of DDoS attackers use multi vector DDoS attacks, which combine a variety of DDoS attacks into one [8].

As recent surveys on network applications, wireless networks, cloud computing, and big data illustrate, DDoS attacks are a particular form of network interference that has attracted academia's attention. The attack on seven South Korean banks to extort money instead of returning their function to normalcy is a recent example of cybercrime's evolution. On September 6 and 7, 2019, In Germany and some parts of Europe, Wikipedia was shot down by a DDoS attack. DDoS attacks became more widespread during the following few years, and Cisco expects that the overall number of DDoS attacks would double from 7.9 million in 2018 to over 15 million by 2023. Given that IT service downtime costs firms anywhere from $300,000 to over $1,000,000 per hour [9]. You can see how even a minor DDoS attack might have a significant financial impact. Various network layers, such as physical, transport, and application layers, can be submerged by DDoS attacks. Since one layer has a single failure point, if it is submerged by some DDoS attack, the entire network will go down simultaneously. The application Layer attack is more complicated and targets individual apps or utilities and exhausts network resources slowly [10,11].

Computer networks are currently undergoing a large-scale and complex evolution. DDoS attacks have continued to develop and grow over the years, with the presence of botnets, and the damage has gotten increasingly significant. DDoS attacks are primarily directed against network bandwidth and network resources, and they are easy to set up, direct, and launch. The least missing resource for an attacker in today's general environment is network resources. As a result, as long as devices can connect to the internet, they can be used as malicious actors. With the advancement of defense functions and attacker technology, new DDoS attack tactics appear one after the other. To successfully drain network bandwidth, the attacker just needs to transmit a high quantity of network data packets to the server at first. Network layer protocols such as UDP and ICMP are utilized for this. Networks and servers have improved their ability to detect DDoS attacks at the network layer over time. Servers have the ability to give better and more bandwidth. Despite this, large queries will use network capacity and cause server unavailability. However, when it comes to attacks, it appears that launching an attack has grown more difficult. As a result, the attacker shifts his focus to the transport layer, but the server begins to defend itself against these attacks over time. The patterns of these attacks can be discovered and properly distinguished across mediums. As network layer and transport layer defenses become stronger, attackers are moving to the application layer, resulting in a new wave of application layer DDoS attacks.

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 following are the portions of the paper: the first section is meant to serve as an overview. The Second section outlines the study that is linked to the research. Section 3 mainly focuses on the suggested algorithm. The simulation and implementation dataset environments are described in Section 4. Section 5 summarizes the findings and comments with a comparative analysis. Other conclusions and future work are stated in Section 6.

2. Related works

The recent growth of application layer DoS attacks have attracted a significant interest of a research community. Since application layer attacks usually do not disclose themselves at the network level, they avoid traditional network based detection mechanisms. As such, security community focused on specialized application-layer DoS attacks detection mechanisms.

According to a recent survey [12] few DDoS defense solutions for HTTP flood rely on application layer data. The DDoS shield [13] protects against HTTP floods by detecting session appearance times and adjusting time of arrival intervals. Another huge model, Using Page Access Lead to Fight HTTP Flood [14,15], investigates the relationship between information size and surfing time. Given the way criminals consolidate Botnets [16], these two methods are insufficient to deal with high rates of packet sending. In this exceptional case, there is an unnecessary burden on the client's component when a CAPTCHA-based probabilistic validation is used. In most cases, if clients are irritated, this equates to Denial of Service, especially when dealing with practical issues. The Hidden semi-Markov model (HSMM) [17] is used to identify the typical client access lead based on the instance of a deal request. Furthermore, the progress of incoming customers is evaluated using HSMM data. This technique generates numerous false alerts because the invoke prototype can change with the greatest frequency due to the remarkable looking at settings of real consumers; the frequency of mentioning is unquestionably not an important metric. Any situation in which clients use external internet connections, provide URLs to likely queries, or use a combination of browsers may result in phony alarms as a check on current concerns. Because it informs the typical client access direct employing representations of deals, the Hidden Semi-Markov Model (HSMM) must be used. The HSMM cutoff points were also employed as a scale to focus on the level of progress toward clients. Because the instance of mentioning is clearly not a basic evaluation, the business model can vary with high frequency due to the distinct analyzing settings of the authentic clients, there are a lot of unnecessary false alarms because of this model. As a result of this argument, clients will be able to pass URLs unmistakably to proper mentioning, will be able to tap external web interfaces, or will be able to employ a combination of actions, which will result in bogus alerts.

The audit of these current models reveals that they have a variety of requirements such as master's responsibility, anticipating static information, and limited assessments. Regardless, these are survey-based meetings. After a short period of time, a client can switch between different social events to perform a series of deals engineered by movement or the same. To that end, we propose a remarkable computational technique for thwarting an HTTP Flood-based DDOS attack. The proposal's goal is to determine whether the area of HTTP exchange is flooded or not, which is accomplished by various assessments wiped out from a level out time stretch rather than meeting. The proposed method's main goal is to evaluate exchange comparability using both common and flood data in order to increase the solicitation of conditional assessments. When compared to previous benchmarking methodologies, the proposed model's highlights come from the observed request stream over an extended period of time. This is due to the fact that the request stream that has been observed provides more accurate data. After this introduction, all subsequent sections of the paper will follow the same structure. Preparing to wrap up a project, review the ARTP model and conduct both a test study and an evaluation of the ARTP's performance are critical steps.

Hameed, S. and Ali, U [18]. described A framework named HADEC for detecting live high-rate DDoS attacks at network and application layers, such as TCP-SYN, HTTP GET, UDP, and ICMP. The detection server and the capture server are the two primary components of the framework. The server responsible for capturing live network traffic and transferring the log to the detection server for processing is the first step. If the source connection exceeds the set threshold, the detection

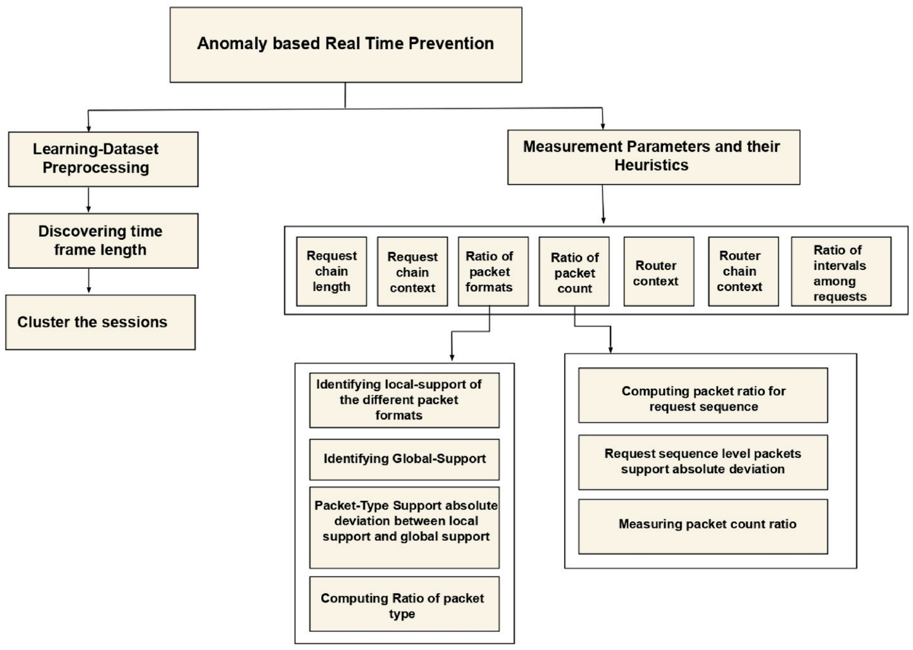

Anomaly based Real Time Prevention
Learning-Dataset Preprocessing
Discovering time frame length
Cluster the sessions
Measurement Parameters and their Heuristics
Request chain length
Request chain context
Ratio of packet formats
Ratio of packet count
Router context
Router chain context
Ratio of intervals among requests
Identifying local-support of the different packet formats
Identifying Global-Support
Packet-Type Support absolute deviation between local support and global support
Computing Ratio of packet type
Computing packet ratio for request sequence
Request sequence level packets support absolute deviation
Measuring packet count ratio

Fig. 1. ARTP representational architecture.

calculates incoming packets for UDP, ICMP, and HTTP. On the cluster nodes, the detection server runs a Hadoop cluster and starts MapReduce-based DDoS detection operations. Financial institutions, small and medium businesses, can benefit from the proposed detection because it is low-cost. They used MapReduce to create a counter-based DDoS detection system for four different types of flooding attack.

Sokolov, M. and Herndon, N [19]. proposed an Ensemble of Light Gradient Boosted Machines to anticipate malware attacks. The authors performed three Tests. Test A is the control experiment, they employed the entire dataset. Test B, authors used LGBM feature extraction with a threshold of 0.5% and reduced the number of features from 114 to 84 features. Test C, authors used Random Forest feature extraction with a threshold of 0.5% and reduced the number of features from 114 to 41 features. They showed that the suggested technique outperformed Automated Artificial Intelligence using a dataset given by Microsoft (Microsoft Kaggle’s Malware Prediction dataset). The architecture proposed is intended to identify malware with a good precision and less processing power.

A lightweight Version Number Attacks VNA detection model called ML-LGBM is suggested by Osman, M. et al. [20]. The construction of a large VNA dataset, a feature extraction technique, an LGBM algorithm, and the highest parameter optimization are all part of the ML-LGBM model’s effort. Extensive tests show that the ML-LGBM model has numerous benefits based on metrics, including accuracy, precision, F1-score, true negative rate, and false-positive rate. Furthermore, the ML-LGBM model has a shorter execution time and requires less memory, making it appropriate for IoT devices with limited resources.

In the literature [21], Bakhareva, N. et al. suggested “Attack detection in enterprise networks using ML methods,” which employs CatBoost tree, Logistic Regression, Linear SVC, and LGBM. They used the CICIDS2017 dataset for binary and multiclass classification. The authors used the “Flow Bytes/s" or “Flow Packets/s" columns and rounding values to five digits after the floating-point. The GPU was used for CatBoost and LGBM, while the CPU for the remainder. According to the authors, CatBoost beats others in terms of cross validation accuracy, F1 score, precision, recall, and AUC, but it comes at a cost in terms of execution times. As a consequence, new possibilities to apply CatBoost to other areas where Gradient Boosted Decision Trees (GBDTs) are beneficial in resolving cyber-security problems may emerge.

In [22], Sanjeetha, R. et al. suggested detection and mitigation of botnets in an SDN environment using ML. They used Mininet to simulate the Botnet and RYU controller and then used Python to create an auxiliary component of the CatBoost model that interacted with the controller through REST API. When discovering a DDoS attack in the network, the hosts responsible for the attack are recognized, and then they add flow rules into switches to mitigate the attack [23]. They made use of a Kaggle open dataset, including around 2.1 million items. Dataset is trained on several models to discover the least amount of training time. The author demonstrates that the proposed approach is 98% accurate [24].

Application DDoS attacks are shown in the diagram from Fig. 1 using anomaly-based Real Time Prediction (ARTP). By utilizing pretty far, plans should be developed based on time span rather than interest level to identify whether the traffic contains assault packages [25]. The concept is put to the test using the benchmark dataset LLDOS. The sequence became easier, and the attained most significant region accuracy differed from prior AI actions [26]. The results are satisfactory, but more study may be required [27].

However, the following weaknesses in the detection of application-layer DDoS attacks remain.

1. The accuracy of attack detection must be improved, and the number of false alarms generated during the detection process must be reduced.
2. The majority of research focuses on detecting high-rate DDoS attacks. Only a tiny proportion of researchers are concerned about low-rate DDoS. Only a few researches have presented strategies for detecting two types of DDoS attacks: high-speed and low-speed DDoS attacks. The detection range should be increased as well.
3. Parts of the data sets now in use are out of date and cannot be meaningfully compared or evaluated.

3. Proposed methodology

Application-DDoS attacks on the host server are detected using the suggested K-Means Algorithm of Machine Learning (ML)-based intrusion detection system (IDS) model. Instead of monitoring floods inside a single session, the model recognizes a huge number of request floods as request flows within a certain time range. The Anomaly-based-Real-Time-Prevention (ARTP) model uses a set of features to detect request flooding. The proposed model is thoroughly examined in the following sections. A block diagram of the architecture is shown in Fig. 1.

3.1. *Learning-dataset preprocessing*

Let us SL represent the session log $SL = \{s_1, s_2, s_3, ...s_{|L|}\}$ and each session is thought of as a collection of requests used for training. Each request is represented as a record $\{r\exists r \in s_i \wedge s_i \in SL\}$ and is labelled as B (Benign) or F (HTTP Flood). The SL is further classified as SL_B containing B type records and SL_F containing F type records. Both SL_B andSL_F session logs are used to assess the assessment parameters.

3.1.1. *Determining the length of the time period*

Sequence the sessions in ascending order of their start time for each session log SL.

Assume that belongs to the set $S_{bt} = \{bt(s_1\exists s_1 \in SL), bt(s_2\exists s_2 \in SL), ..., bt(s_{|SL|}\exists s_{|SL|} \in SL)\}$ that represents the time at which the session begins for all SL sessions.

Assume that belongs to the set $S_{et} = \{et(s_1\exists s_1 \in SL), et(s_2\exists s_2 \in SL), ..., et(s_{|SL|}\exists s_{|SL|} \in SL)\}$ that specifies the end time of the SL session for the entire session.

Assume that belongs to the set $S_{lt} = \{lt(s_1\exists s_1 \in SL), lt(s_2\exists s_2 \in SL), ..., lt(s_{|SL|}\exists s_{|SL|} \in SL)\}$ that shows how long a session has been active in Second Life.

In (Eq 3.1), the session $\{s_i\exists s_i \in SL\}$ life time is calculated as follows:

$$lt(s_i\exists s_i \in SL) = et(s_i\exists s_i \in SL) - bt(s_i\exists s_i \in SL) \qquad \text{(Eq.1)}$$

Determine the absolute deviation [12] of S_{bt} the session start time.

$$ad(S_{bt}) = \frac{\sqrt{\sum_{i=1}^{|S_{bt}|} (\langle S_{bt}\rangle - bt(s_i\exists s_i \in S_{bt}))^2}}{|S_{bt}|} \qquad \text{(Eq.2)}$$

In (Eq (2)), $|S_{bt}|$ is the total number of S_{bt} sessions remaining and $\langle S_{bt}\rangle$ represents the average time S_{bt} remaining in the session. Determine the absolute difference S_{et} between this time and now.

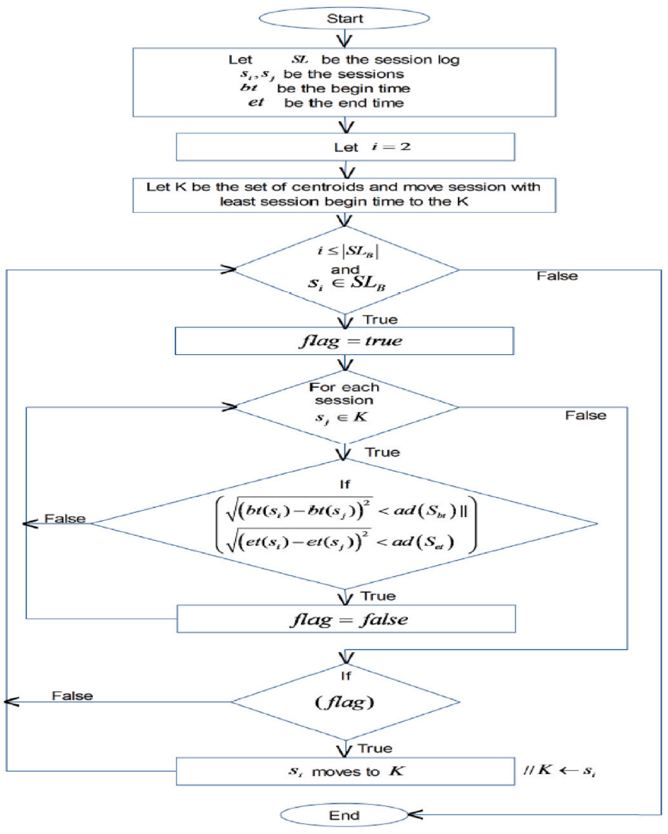
$$ad(S_{et}) = \frac{\sqrt{\sum_{i=1}^{|S_{et}|} (\langle S_{et}\rangle - et(s_i\exists s_i \in s_i \in S_{et}))^2}}{|S_{et}|} \qquad \text{(Eq.3)}$$

Equation (3) can now be used to calculate *atf* the total time frame S_{et}, which can then be used in Equation (4) to determine range $|S_{et}|$ and $\langle S_{et}\rangle$ standard deviationS_{et}:

$$atf = (\langle S_{et}\rangle + ad(S_{et})) - (\langle S_{bt}\rangle + ad(S_{bt})) \qquad \text{(Eq.4)}$$

3.1.2. *Cluster the sessions*

Obtaining the value of K (number of centroids) as shown in Fig. 2:



Start
Let SL be the session log
s_i, s_j be the sessions
bt be the begin time
et be the end time
Let $i = 2$
Let K be the set of centroids and move session with least session begin time to the K
$i \le |SL_B|$ and $s_i \in SL_B$
False
True
$flag = true$
For each session $s_j \in K$
False
True
If $\left(\sqrt{(bt(s_i)-bt(s_j))^2} < ad(S_{bt}) \,||\, \sqrt{(et(s_i)-et(s_j))^2} < ad(S_{et})\right)$
False
True
$flag = false$
If $(flag)$
False
True
s_i moves to K
// $K \leftarrow s_i$
End

Fig. 2. Finding the value of K (Count of centroids) as represented by a flowchart.

Then, using K-Means Algorithm [12], determine the number of clusters with periods in roughly comparable time frames.

Let $CL = \{cl_1, cl_2, cl_3, ..., cl_K\}$ be the size K of set of clusters.

It's best to use the following example to illustrate how to determine how much time has elapsed between the beginning and ending times of each individual session in your cluster $\{cl_i\exists cl_i \in CL \wedge i = 1, 2, 3, 4, ..., K\}$.

$\overset{|K|}{\underset{i=1}{\forall}}\{cl_i\exists cl_i \in CL\}$ StartLet $bt_B(cl_i) = \{bt_1, bt_2, bt_3, ..., bt_{|cl_i|}\}$ sessions belonging to the cluster, ordered from earliest to latest, should be stored as a list of their respective session start times CL_i

Let $et_B(cl_i) = \{et_1, et_2, et_3, ..., et_{|cl_i|}\}$ is represents cluster end times are listed in reverse chronological order cl_i.

The cluster's cl_i time frame $tf(cl_i)$ is then measured using the formula (Eq (5)):

$$tf(cl_i) = \sqrt{(et_1 - bt_1)^2} \qquad \text{(Eq 5)}$$

```
Loop 1: $\overset{|SL_B|}{\underset{i=2}{\forall}}\{s_i\exists s_i \in SL_B\}$ start
    $flag = true$
    For every period $\{s_j\exists s_j \in K\}$ start
        If $\left(\sqrt{(bt(s_i)-bt(s_j))^2} < ad(S_{bt}) \,||\, \sqrt{(et(s_i)-et(s_j))^2} < ad(S_{et})\right)$ then start
            $flag = false$
        End
    End
    if $(flag)$ then start
        $K \leftarrow s_i$
    End
End
```

Then, as defined in (Eq (6)), Find out the average length of the time frames observed in all clusters:

$$\langle tf(CL)\rangle = \frac{\sum_{i=1}^{K} tf(cl_i)}{K} \qquad \text{(Eq 6)}$$

For each cluster, calculate the Absolute Time Frame Deviation $ad(tf)$ and (Eq (7)) as shown in the formula:

$$ad(tf) = \frac{\sqrt{\sum_{i=1}^{K} (\langle tf(CL)\rangle - tf(cl_i))^2}}{K} \qquad \text{(Eq 7)}$$

Then, as defined in (Eq (8)): based on the average length of previous time frames *tf* and the Time Frame Absolute Deviation, adjust the length of the time frame:

$$tf = \langle tf(CL)\rangle + ad(tf) \qquad \text{(Eq 8)}$$

3.2. Metrics and their heuristics

3.2.1. Request chain length (RCL)

Data that is used for training is marked as either B or F and the length of records recognized from clients is measured by taking the longest request contained in a specific time to order as legitimate or attack-prone. The maximum length of requests contained by a precise time is used to calculate the mean length of records recognized from clients. The following formula is used to calculate the RCL value:

- Separate the record sequence $RS(SL_B) = \{rs_1, rs_2, rs_3, \ldots\ldots, rs_{|RS(SL_B)|}\}$ so that it only contains the records identified within & the span of the request sequence.
- Determine the average length of the record chain identified across all request sequence shares observed from SL_B the following:

$$\langle RS(SL_B)\rangle = \sum_{i=1}^{|RS(SL_B)|} \{|rs_i| \exists rs_i \in RS(SL_B)\} \qquad \text{(Eq 9)}$$

In (Eq (9)), $\langle RS(SL_B)\rangle$ represents the average of the amount of records recognized for all request sequences $RS(SL_B)$ and $|rs_i|$ is the amount of records recognized in the span of $|al|$ the sequence request.

- As shown in (Eq (10)), sequence groups were discovered SL_B.

$$ad(rcl) = \frac{\sqrt{\sum_{i=1}^{|RS(SL_B)|} (\langle RS(SL_B)\rangle - (|rs_i| \exists rs_i \in RS(SL_B))^2}}{|RS(SL_B)|} \qquad \text{(Eq 10)}$$

- Set the metric defined in (Eq (11)) to the following value:

$$rcl(SL_B) = \langle RS(SL_B)\rangle + ad(rcl) \qquad \text{(Eq 11)}$$

3.2.2. Request chain context (order of requests)

The model is trained on a predetermined set of cached records, which are then used to evaluate the authenticity of a given set of requests made within a given time period. The scope of the request is estimated to be genuine or suspicious in the following stage.

In order to ensure that any two requests in a series as a pair $rp_B = \{rp_1, rp_2, rp_3, \ldots\ldots., rp_{|rp_B|}\}$ find local support for each pair, create a request pair-set specifically for caching records SL_B so that any two requests as a pair find local support $ls_{rs_j}(rp_i)$ for each pair$(rp_i \exists rp_i \in rp_B \wedge i = 1, 2, \ldots.., |rp_B|)$. Each pair of requests rp_i in a series can now find support in the local area for the other request in the series rs_j. To find the total number of times a pair rp_i is requested, find the global support SL_B for each pair. Each pair's $(rp_i \exists rp_i \in rp_B \wedge i = 1, 2, \ldots.., |rp_B|)$ global support should be determined $gs(rp_i)$.

$\overset{|rp_B|}{\underset{i=1}{\forall}} \{rp_i \exists rp_i \in rp_B\}$

$$ad(rp) = \sqrt{\frac{\sum_{j=1}^{|RS(SL_B)|} \left(gs(rp_i) - ls_{rs_j}(rp_i)\right)^2}{|RS(SL_B)|}} \qquad \text{(Eq 12)}$$

Then, as shown in (Eq (13)), consider how the chain of requests $rcc(rp_i)$ affects each individual request rp_i: $\overset{|rp_B|}{\underset{i=1}{\forall}} \{rp_i \exists rp_i \in rp_B\}$

$$rcc(rp_i) = \frac{\sum_{j=1}^{|RS(SL_B)|} ls_{rs_j}(rp_i)}{|RS(SL_B)|} + ad(rp) \qquad \text{(Eq 13)}$$

3.2.3. Ratio of packet formats (packets in a fixed time fame)

In a given period of time, it shows the percentage of authentic or vulnerable packets. Thereafter, these ratios are used to determine the threshold levels for each sub-format, such as UDP, ICMP, or TCP-SYN, for records that are either genuine or vulnerable.

Packet-Format Absolute support deviation between packet formats with local and global support: $\overset{|PF|}{\underset{j=1}{\forall}} \{pf_j \exists pf_j \in PF\}$ Begin

$$ad(pf_j) = \sqrt{\frac{\sum_{i=1}^{|RS(SL_B)|} \left(gs(pf_j) - ls_{rs_i}(pf_j)\right)^2}{|RS(SL_B)|}} \qquad \text{(Eq 14)}$$

3.2.4. Computing ratio of packet format (rpf)

Then, using (Eq (15)), compute the packet-format ratio $rpf(pf_j)$:

$$rpf(pf_j) = \frac{\sum_{i=1}^{|RS(SL_B)|} ls_{rs_i}(pf_j)}{|RS(SL_B)|} + ad(pf_j) \qquad \text{(Eq 15)}$$

3.2.5. Count of packets

3.2.5.1. Computing packet ratio for request sequence.

$\overset{|RS(SL_B)|}{\underset{i=1}{\forall}} \{rs_i \exists rs_i \in RS(SL_B)\}$ Begin

$$pr(rs_i) = \sum_{i=1}^{|RS(SL_B)|} \frac{|P(rs_i)|}{\sum_{k=1}^{|RS(SL_B)|} |P(rs_k)|} \qquad \text{(Eq16)}$$

In (Eq (16)), $pr(rs_i)$ is the packet ratio forrs_i, $|P(rs_i)|$ is the total number of packets in the request sequencers_i, and $\sum_{k=1}^{|RS(SL_B)|} |P(rs_k)|$ is represents the total number of packets present in each request sequence.

3.2.5.2. Request sequence level packets support absolute deviation

$$ad(pr) = \frac{\sqrt{\sum_{j=1}^{|RS(SL_B)|} \left(1 - pr(rs_j)\right)^2}}{|RS(SL_B)|} \qquad \text{(Eq 17)}$$

3.2.5.3. Measuring packet count ratio

$$pcr(RS(SL_B)) = \frac{\sum_{i=1}^{|RS(SL_B)|} pr(rs_i)}{|RS(SL_B)|} + ad(pr) \qquad \text{(Eq 18)}$$

3.2.5.4. Router context.

The algorithm predicts the scope of every router in the process based on the group of cached records used for training that have been classified as genuine or vulnerable.

Let $RO_B = \{ro_1, ro_2, ro_3, \ldots\ldots., ro_{|RO_B|}\}$ denote the group of routers that have been identified as participating in the request and response exchange for theSL_B, Identify the number of occurrences $ls_{rs_j}(ro_i)$ of the router within the request sequencers_j for each router $(ro_i \exists ro_i \in RO_B \wedge i = 1, 2, \ldots, |RO_B|)$. Determine the global-support $gs(ro_i)$ for each router $(ro_i \exists ro_i \in RO_B \wedge i = 1, 2, \ldots, |RO_B|)$, which indicates the number of times a router appears in a request sequence $\{rs_i \exists rs_i \in RS(SL_B) \wedge i = 1, 2, \ldots |RS(SL_B)|\}$.

Determine the Local-Support Provider Absolute Deviation based on (Eq (19)) for each router in the global-support:

$$\overset{|RO_B|}{\underset{i=1}{\forall}} \{ro_i \exists ro_i \in RO_B\}$$

$$ad(ro_i) = \frac{\sqrt{\sum_{j=1}^{|RS(SL_B)|} \left(gs(ro_i) - ls_{rs_j}(ro_i)\right)^2}}{|RS(SL_B)|} \qquad \text{(Eq 19)}$$

Then, for each router ro_i, compute the router-context $roc(ro_i)$, as shown in (Eq (20)):

$$\overset{|RO_B|}{\underset{i=1}{\forall}} \{ro_i \exists ro_i \in RO_B\}$$

$$roc(ro_i) = \frac{\sum_{j=1}^{|RS(SL_B)|} ls_{rs_j}(ro_i)}{|RS(SL_B)|} + ad(ro_i) \qquad \text{(Eq 20)}$$

3.2.5.5. Router chain context. The order of routers classified as genuine or vulnerable within a specific timeframe is separated from the cached records used for training. Following that, the model predicts whether the given request is genuine or vulnerable.

Develop a router pair group $ROP_B = \{rop_1, rop_2, rop_3, \ldots, rop_{|ROP_B|}\}$ for the cached-records setSL_B so that every two requests in series rop_i are treated as a pair and find local-support $ls_{rs_j}(rop_i)$ for each pair $(rop_i \exists rop_i \in ROP_B \wedge i = 1, 2, \ldots, |ROP_B|)$, how many times that pair rop_i is requested in a row. The number of times rs_j a particular router pair appears in the entire chain of routers is known as the global-support $gs(rop_i)$ of that router pair$(rop_i \exists rop_i \in ROP_B \wedge i = 1, 2, \ldots, |ROP_B|)$. The global-support column in the router sequence SL_B can be used to determine this.

The equation shown in (Eq. (21)) can be used to determine the absolute difference between the local and global levels of support for each router pair.

$$\overset{|ROP_B|}{\underset{i=1}{\forall}} \{rop_i \exists rop_i \in ROP_B\}$$

$$ad(rop_i) = \frac{\sqrt{\sum_{j=1}^{|RS(SL_B)|} \left(gs(rop_i) - ls_{rs_j}(rop_i)\right)^2}}{|RS(SL_B)|} \qquad \text{(Eq 21)}$$

Then, for each router pair rop_i, the router chain context $rocc(rop_i)$ can be calculated as (Eq (22)):

$$\overset{|ROP_B|}{\underset{i=1}{\forall}} \{rop_i \exists rop_i \in ROP_B\}$$

$$rocc(rop_i) = \frac{\sum_{j=1}^{|RS(SL_B)|} ls_{rs_j}(rop_i)}{|RS(SL_B)|} + ad(rop_i) \qquad \text{(Eq 22)}$$

3.2.5.6. Ratio of intervals among requests. Session duration is defined as the time interval between one request and the next in a sequence within a single session from a group of cached records considered for training. Each time frame's ratio of genuine to suspicious intervals is calculated using the access period that is assigned to all request pairs within a single session.

Compute the local average $avg_l(rs_i)$ of the intervals between requests within each request sequence $\{rs_i \exists rs_i \in RS(SL_B)\}$, as defined in (Eq (23)):

$$\overset{|RS(SL_B)|}{\underset{k=1}{\forall}} \{rs_k \exists rs_k \in RS(SL_B)\}$$

$$avg_l(rs_k) = \frac{\sum_{j=1}^{|I(rs_k)|} ti_j}{|I(rs_k)|} \qquad \text{(Eq 23)}$$

In (Eq 3.23), a $I(rs_k)$ group of time intervals identified among the pair requests within a sequence rs_k is identified.

As shown in (Eq (24)), compute the global average $avg_g(SL_B)$ of the intervals between requests in the sequenceSL_B:

$$avg_g(SL_B) = \frac{\sum_{j=1}^{|I(SL_B)|} ti_j}{|I(SL_B)|} \qquad \text{(Eq 24)}$$

Set of intervals $I(SL_B)$ derived from SL_B pair requests, arranged in ascending order.

On the basis of (Eq (25)), calculate the absolute deviation of time intervals $ad(ti)$.

$$ad(ti) = \frac{\sqrt{\sum_{j=1}^{|RS(SL_B)|} \left(avg_g(SL_B) - avg_l(rs_j)\right)^2}}{|RS(SL_B)|} \qquad \text{(Eq 25)}$$

Then, using (Equation (26)), compute the time interval ratio $tir(SL_B)$:

$$tir(SL_B) = \frac{\sum_{i=1}^{|RS(SL_B)|} avg_l(rs_i)}{|RS(SL_B)|} + ad(ti) \qquad \text{(Eq 26)}$$

Table 1
Values for metrics devised from the experimental data.

Metric	Value Observed	
	App-DDOS	Normal
Absolute Time Frame	2700 ms	2700 ms
Request chain length	176	124
Number of Request Chain Context Patterns	25	14
Ratio of packet types	0.8574899	0.568959
Ratio of Packet Count	0.8373156	0.509015
Ratio of Router Context	0.396215	0.734201
The Patterns of the Router Chain Context, Counted	6	4
Ratio of intervals between requests	0.178284	0.226098

4. Simulations and implementation environment

The experiments were finished with regards to evaluating the location exactness and versatility, robustness and cycle intricacy of the proposed ARTP.

The LLDOS 2.0.2-scenario two is used to generate the App-DDOS attack requests of aggregate time interval of 1800 s. 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 metrics38 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. According to (Equation (27)), a total of 666 distinct periods of time have been identified over the course of the experiment.

Table 2
Metrics of statistical performance and observed values.

TN	82
FN	21
TP	120
FP	18
Sensitivity	0.927543
Specificity	0.863452
Accuracy	0.892341
Precision	0.895436

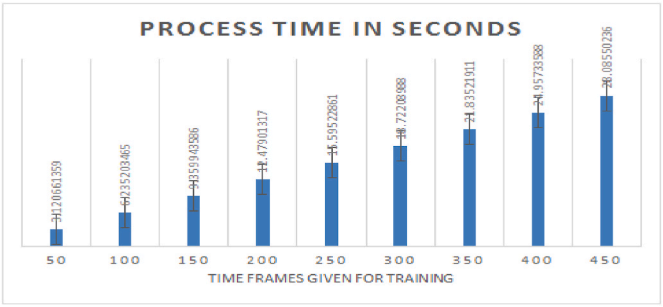


Fig. 3. Procedure completion duration for different number of frames allocated to training.

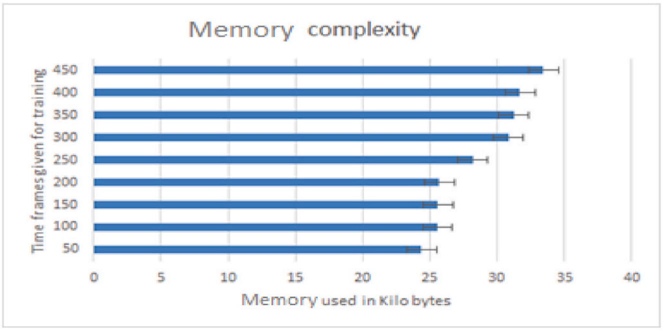


Fig. 4. Memory utilized for training as identified across different frames.

$$sr(H_N) = \frac{\sum_{j=1}^{|PS(H_N)|} \{s_j \exists s_j \in PS(H_N)\}}{|PS(H_N)|} \qquad \text{(Eq 27)}$$

A total time *ati* period of (1800*1000) milliseconds will be used for the simulation, and an absolute time period of *atf* will be determined (2700 ms).

444 frames of the 666 frames are used to train the model, while the remaining 222 frames are used to assess the model. 135 frames correspond to attack streams, and 87 frames correspond to genuine request streams, out of the 222 frames used for assessment.

5. Findings and comments with comparitive analysis

The ARTP model detection statistics are explored in Table 2. For demonstrating model efficiency, the following assessment parameters are taken into account.

- Accuracy rates – attack detection ability
- Sensitivity-the ratio of attack request identification
- Specificity- The proportion of genuine requests identified
- Accuracy-to-identification ratio for genuine and malicious streams

A higher sensitivity to specificity ratio indicates that the chances of incorrectly classifying attack requests as genuine requests are much lower than the chances of incorrectly classifying genuine requests as attack requests. The sensitivity value for the proposed approach is 0.126, with a specificity value of 0.184. As a result, the model presented in this chapter is more sensitive in detecting attack requests.

Training was given in various time frames allocated for testing in order to determine the extent of complexity involved in the proposed model. As shown in the figures below, the study's findings show that both time-complexity and memory-complexity values are linear for training. Fig. 5 depicts the importance of the metrics in detecting App-DDoS attacks.

In all the instances of Fig. 3, the completion duration is smaller than that of the anticipated completion duration. Accordingly, the procedure complexity can be considered as linear. Fig. 4 depicts the linearity in memory usage for the process of training phase carried under different time frames.

The parameters identified for assessment of the ARTP model have

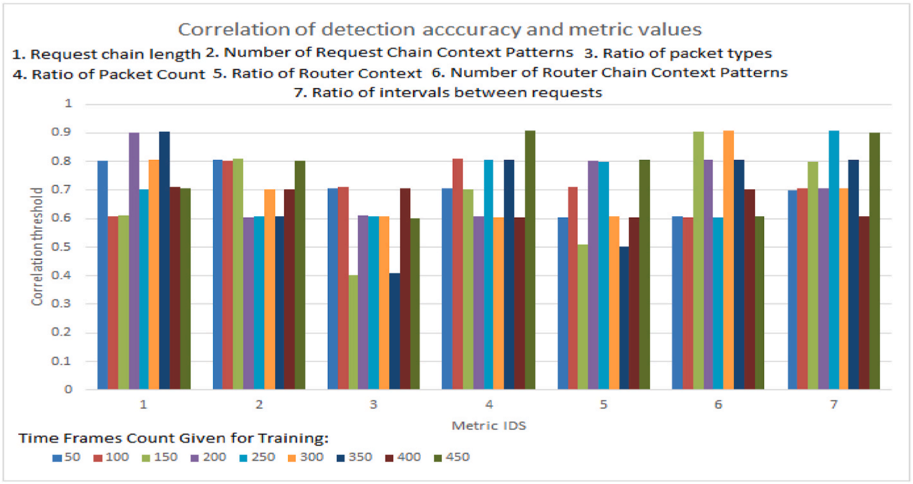


Fig. 5. Correlation computed between different computational metrics and identification accuracy.

been identified to be prominent towards attack detection. In particular, the assessment parameter ratio of packet types is identified to be having less influence over detection accuracy among all the considered parameters. The final experimental analysis demonstrates that the parameters considered for ARTP model assessment are prominent and the model has better performance in terms of detection accuracy with respect to Application DDoS attacks.

6. Conclusion and future work

Anomaly-based-Real-Time-Prevention (ARTP) is a novel Machine Learning (ML) based approach for detecting Application Distributed Denial of Service (DDoS) attacks in internet networks quickly and reliably. The contribution of the work can be broadly categorized in three stages. The first part determines the attributes required for determining if the inflow request stream is genuine user generated or malicious type. Contrary to conventional models, the attributes are evaluated on the request streams within a specific time frame and not within a session. In the next part, the proposed model learns to classify the inflow request streams into genuine and malicious on the basis of threshold levels preset for the assessment metrics. In the final part, the model is tested for its efficiency in classification using the standard LLDoS corpus. During the test phase, the model demonstrated better performance and computational simplicity. The empirical analysis concludes that the metrics considered for classification are prominent for the model to learn the classification process from the training records corpus. In addition, the threshold levels determined during the training phase are utilized to classify the request stream within specific time frame. The model also demonstrated scalability, robustness, low computational complexity and quick detection. Accordingly, the proposed approach ARTP achieves high detection accuracy rates with least possible complexity.

Further research scope can include designing more optimal features or attributes for model learning and formulating new strategies towards machine learning from anomalies. In addition, the proposed model can be extended to identify the other significant attacks like Flash crowd.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Simon Byers, Aviel D. Rubin, David Kormann, Defending against an Internet-based attack on the physical world, ACM Trans. Internet Technol. 4 (2004) 239–254, https://doi.org/10.1145/1013202.1013203, 3 (August 2004).
[2] J.M. Estevez-Tapiador, P. Garcia-Teodoro, J.E. Diaz-Verdejo, Detection of Web-based attacks through Markovian protocol parsing, in: 10th IEEE Symposium on Computers and Communications (ISCC'05), 2005, pp. 457–462, https://doi.org/10.1109/ISCC.2005.51.
[3] V. Jyothsna, V V Prasad Rama, Article: a review of anomaly based intrusion detection systems, Int. J. Comput. Appl. 28 (7) (August 2011) 26–35.
[4] C. Ishida, Y. Arakawa, I. Sasase, K. Takemori, Forecast techniques for predicting increase or decrease of attacks using Bayesian inference," PACRIM, in: 2005 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2005, 2005, pp. 450–453, https://doi.org/10.1109/PACRIM.2005.1517323.
[5] R.A. Cacheda, D.C. García, A. Cuevas, F.J. Castano, J.H. Sánchez, G. Koltsidas, V. Mancuso, J.I. Novella, S. Oh, A. Pantò, QoS requirements for multimedia services, in: Resource Management in Satellite Networks 2007, Springer, Boston, MA, 2007, pp. 67–94.
[6] D. Palmer, DDoS Attacks that Come Combined with Extortion Demands Are on the Rise, ZDNet, City, 2022.
[7] A. Gutnikov, DDoS Attacks in Q3 2021. Yandex and Qrator Labs, 2021. City.
[8] A. Gutnikov, DDoS Attacks in Q3 2021. DDoS Attacks in Q3 2021, 2021. City.
[9] C. Douligeris, A. Mitrokotsa, DDoS attacks and defense mechanisms: classification and state-of-the-art, Comput. Network. 44 (5) (2004 Apr 5) 643–666.
[10] P. Nicholson, Five Most Famous DDoS Attacks and Then Some, 2021. City.
[11] M.S. Elsayed, N.-A. Le-Khac, S. Dev, A.D. Jurcut, Ddosnet: A Deep-Learning Model for Detecting Network Attacks, IEEE, City, 2020.
[12] T. Peng, C. Leckie, K. Ramamohanarao, Survey of network-based defense mechanisms countering the DoS and DDoS problems, ACM Comput. Surv. 39 (1) (2007 Apr 12) 3.
[13] Sekaran Kaushik, G. Raja Vikram, B.V. Chowdary, Design of effective security architecture for mobile cloud computing to prevent DDoS attacks, Int. J. Wireless Microw. Tech.(IJWMT) 9 (No.1) (2019) 43–51, https://doi.org/10.5815/ijwmt.2019.01.05.
[14] M. Handley, DoS-resistant Internet Subgroup Report, Internet Architecture WG, 2005.
[15] W.R. Cheswick, S.M. Bellovin, A.D. Rubin, Firewalls and Internet Security: Repelling the Wily Hacker, Addison-Wesley Longman Publishing Co., Inc., 2003 Feb 1.
[16] McAfee, Personal firewall. http://www. mcafee.com, 2003.
[17] Y. Xie, A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors, IEEE/ACM Trans. Netw. (2009) 54–65.
[18] S. Hameed, U. Ali, HADEC: Hadoop-based live DDoS detection framework, EURASIP J. Inf. Secur. 1 (2018) 1–19.
[19] M. Sokolov, N. Herndon, Predicting Malware Attacks Using Machine Learning and AutoAI, 2021. City.
[20] M. Osman, J. He, F.M.M. Mokbal, N. Zhu, S. Qureshi, ML-LGBM: a machine learning model based on Light gradient boosting machine for the detection of version number attacks in RPL-based networks, IEEE Access 9 (2021) 83654–83665.
[21] N. Bakhareva, A. Shukhman, A. Matveev, P. Polezhaev, Y. Ushakov, L. Legashev, Attack Detection in Enterprise Networks by Machine Learning Methods, IEEE, City, 2019.
[22] R. Sanjeetha, A. Raj, K. Saivenu, M.I. Ahmed, B. Sathvik, A. Kanavalli, Detection and mitigation of botnet based DDoS attacks using catboost machine learning algorithm in SDN environment, Int. J. Adv. Technol. Eng. Explor. 8 (2021) 445, 76.
[23] R. Oppliger, Internet security: firewalls and beyond, Commun. ACM 40 (5) (1997 May 1) 92–102.
[24] C. Leys, C. Ley, O. Klein, P. Bernard, L. Licata, Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median, J. Exp. Soc. Psychol. 49 (4) (2013 Jul 1) 764–766.
[25] https://www.ll.mit.edu/ideval/data/2000data.html.
[26] M.I. Mit, data, Retrieved from LINCOLN LABORATORY: https://www.ll.mit.edu/ideval/data/2000data.html, 2000.
[27] D.M. Powers, Evaluation: from Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation, 2011.