APPLICATION LAYER DDoS ATTACK DETECTION IN THE PRESENCE OF FLASH CROWD

Biruk Asmare Muse, Surafel Lemma Abebe
Addis Ababa Institute of Technology, Addis Ababa University
{biruk.asmare, surafel.lemma}@aait.edu.et

ABSTRACT

Application layer DDoS attacks are growing at an alarming rate in terms of attack intensity and number of attacks. Attackers target websites of government agencies as well as private business for different motives. In some situations, application layer DDoS attacks occur together with characteristically analogous flash crowds. This paper focuses on distinguishing application layer DDoS attacks from flash crowds. Both flash crowd and application layer DDoS attack cause denial of service. Flash crowds come from sudden surge in traffic of legitimate requests. Whereas, application layer DDoS attacks are intentionally generated by attackers to cause denial of service. Distinguishing between application layer DDoS attack and flash crowd is important because the response taken for the case of flash crowd is different from response taken for application layer DDoS attack. Flash crowds are legitimate requests which should be serviced. Application layer DDoS attacks, on the other hand, are malicious requests that should not be serviced. In this research, supervised machine learning based application layer DDoS detection approach is proposed to distinguish between application layer DDoS attack and flash crowd. Features that help distinguish application layer DDoS attacks from legitimate flash crowds were identified. Six supervised classifiers were evaluated using World Cup 98 flash crowd dataset and experimentally generated application layer DDoS attack dataset. The results show that decision tree outperformed other classifiers considering combination of classification time, F1-score and FPR. Decision tree has F1-score of 99.45% and false positive rate of 0.47%.

Keywords: DDoS attack, flash crowd, application layer

INTRODUCTION

Distributed Denial of Service (DDoS) attacks are attacks against availability of Internet services. DDoS attacks are divided into application layer and network layer attacks. Network layer attacks exploit flaws of network and transport layer protocols while application layer DDoS (APP-DDoS) attacks use application layer protocols such as HTTP, FTP and SMTP [1, 2]. The attack is conducted after creating a successful TCP connection. This characteristics makes the attack resistant to most network layer detection and mitigation systems, and hence, difficult to detect [1].

As the threats of APP-DDoS attacks grow in type and complexity, a number of approaches were proposed to help distinguish between APP-DDoS attack and normal activity [1, 2, 3, 4, 5, 6].

In this research, we deal with APP-DDoS attacks that occur together with flash crowds. Flash crowd is a sudden or anticipated large surge in number of requests to a website by legitimate clients due to the addition of some news or when a new product is released [4]. Application layer DDoS attacks have similar characteristics as legitimate flash crowds. Hence, distinguishing between flash crowds and APP-DDoS attacks is a very important network security problem. Realizing the importance, several researches proposed different approaches [3, 7, 8, 9, 10].

The existing approaches that are used to distinguish APP-DDoS attacks from flash crowd have three limitations. The first limitation is that the approaches rely on one or two features for detection [8, 10]. This impacts the robustness of such

detection systems and helps attackers to easily mimic legitimate requests in their attack. The second limitation is that some approaches rely on network layer information such as IP address entropy and packet flow rate [3, 9]. However, it is not difficult to deploy proportional attack machines to that of legitimate machines in flash crowd. The third limitation is that existing information theory based approaches require accurate model of legitimate traffic as a baseline [9]. This is usually difficult to obtain considering the variable nature of internet traffic.

To address the aforementioned problem, we propose a system that distinguishes APP-DDoS attacks from legitimate flash crowds using combination of five features: *request rate, page popularity, download rate, request inter-arrival time* and *ratio of successful requests*. We conjecture that these features will help to distinguish DDoS attack from flash crowd. The features could easily be obtained from web server logs and computed by considering a given time interval, called session time, for each unique client. A client is a machine which is identified by an IP address and makes a request to a server. The core part of the proposed detection system is a supervised learning classifier that classifies a client to either normal client or attack client. The classifier is trained using examples of both flash crowds and APP-DDoS attack. The examples are collection of records which contain feature values.

To evaluate our proposed approach, a data set containing examples of flash crowd and APP-DDoS attack is prepared. The World Cup 98 data [1] is used to model flash crowds. World Cup 98 data is a collection of requests made to *www.france98.com* during the duration of World Cup 98 football game. World Cup 98 data set is used as flash crowd data set in related researches [3, 8, 11, 12]. We prepared application layer DDoS attack data set by performing attack on locally hosted version of the same website using BoNeSi [13] DDoS attack tool.

Using the prepared dataset, we tested our proposed approach in terms of performance of candidate classifiers for detection, effect of session time on detection performance and contribution of identified features for detection. The result shows that although AdaBoost, random forest and decision tree classifiers have very close classification performance, decision tree outperformed all other tested classifiers considering classification time. Decision tree has F1-score of 99.45% and false positive rate of 0.47%. Furthermore, variation of session time has very little impact on the performance of decision tree classifier. Among all features, download rate and request rate have highest contribution for detection.

The specific contributions of this paper are as follows:

- A supervised classifier-based detection system that distinguishes between APP-DDoS attacks and flash crowds is proposed. The proposed approach uses features directly available from server access logs that can be computed with small resources. In addition, our detection model has minimal computational and memory overhead during operation which is important requirement for real time DDoS detection and defense systems. The proposed approach does not rely on establishing accurate legitimate traffic baseline. It is adaptive to different APP-DDoS attack and flash crowd behaviors.
- The commonly used World Cup 98 flash crowd dataset is complemented by performing APP-DDoS attack on locally cloned World Cup 98 website. The combined flash crowd and APP-DDoS dataset is available on request for replication and comparison purposes.
- The detection and computation performance of the proposed approach is empirically evaluated. The contribution of each feature to

distinguish between APP-DDoS attacks and flash crowds is also discussed.

The rest of the paper is organized as follows. Section 2 describes related research work. The proposed approach is presented in Section 3. Sections 4 and 5 discuss experiments used to evaluate the proposed approach and evaluation result, respectively. In Section 5, the proposed approach is also compared with state of the art. Section 6 discusses conclusion and future work.

Related work

Taxonomy of flash crowds and some features that help differentiate APP-DDoS attacks from flash crowds were discussed in the work of Bhandari et al. [11]. Features such as distribution of requests among source IP, geographical distribution of source IP, URL access behavior and change in rate of request were suggested to distinguish between flash crowd and APP-DDoS attack [11]. The authors used World Cup 98 dataset to model flash crowds and created APP-DDoS attack using simulation to investigate the significance of the suggested features. The result showed that URL access behavior has more contribution to distinguish APP-DDoS attacks from flash crowds. Page popularity in our work is used to capture URL access behavior. The authors recommended combination of network layer and application layer features for APP-DDoS detection from flash crowd.

Information obtained from network packets such as source address, destination address is used together with packet flow rate and time interval to distinguish APP-DDoS attacks from flash crowd [3, 7, 8, 9]. Sahoo et.al [7] exploited generalized entropy and information distance to distinguish between APP-DDoS and flash crowd in software defined networks. Behal et al. [8] proposed ISP level detection approach using the aforementioned information distance metrics. Daneshgadeh et al. [3] used a combination of machine learning based, Shannon entropy and Mahalanobis distance to distinguish between normal traffic, flash crowd and APP-DDoS attack. The authors used similar flash crowd dataset and the same APP-DDoS attack tool with our work and obtained a precision of 93% and recall of 100%. Khalf et al. [10] proposed a software agent-based model using attack intensity and IP address information to address the problem of distinguishing APP-DDoS from flash crowd attack. Although promising results are obtained in the information based metrics, All approaches rely on the assumption that APP-DDoS attackers use less number of unique IP addresses compared to legitimate users. However, considering the large number of available IOT devices that can potentially be deployed in this attack, the attackers can deploy proportional number of unique IP addresses with legitimate users. This makes the aforementioned approaches ineffective in such scenarios.

In addition to network layer features, there are some application layer features to distinguish flash crowds from APP-DDoS attacks. Yu et al. [14] suggested page popularity to identify APP-DDoS attack from flash crowd, while, Xie et al. [15] and Ye et al. [16] suggested page access transition to identify APP-DDoS attack from legitimate flash crowd. In the work of Yu et al. [14], page access entropy was suggested by assuming that the entropy of flash crowd page access is different from APP-DDoS attack. This approach may not work when the attacker requests popular pages by studying the website. Ye et al. proposed the transition behavior between web pages for detection of APP-DDoS attack [16]. Xie et al. modeled spatial and temporal user access patterns of flash crowds using hidden Semi-Markov model to achieve the same goal [15]. Another approach that uses a combination of network layer and application layer features was suggested by Ramamoorthi et al. [17]. It uses features such as HTTP

request rate and page viewing time from application layer and session rate, number of TCP, UDP and ICMP packets from network layer. Enhanced support vector machine with string kernel was used to model legitimate flash crowd. They obtained a classification accuracy of 99.32%. Request rate, page popularity and page access pattern were commonly used features for detection of APP-DDoS attacks against normal or flash crowd [11, 14, 17].

Existing approaches that are used to distinguish APP-DDoS attacks from flash crowd have three limitations. The first limitation is that the approaches rely on one or two features for detection which impacts the robustness of such detection systems. This in turn helps attackers to easily mimic legitimate requests in their attack. For example, detection systems that rely on page popularity may fail when the attacker studies the website to identify most popular pages and then programs its zombies to request most popular pages. Again, if the detection system considers page access transition for detection, the attacker may easily program its zombies to follow a similar access pattern to that of legitimate users. This shows that using a combination of the above features will make the detection system more robust. The second limitation is that some approaches rely on network layer information such as IP address entropy and packet flow rate. However, it is not difficult to deploy proportional attack machines to that of legitimate machines in flash crowd. The third limitation is that, most existing approaches require accurate model of legitimate traffic as a baseline which is difficult to obtain considering the variable nature of internet traffic.

To address this gap in state of the art, we propose a supervised machine learning based APP-DDoS detection approach that distinguishes APP-DDoS attacks from flash crowd using a combination of features. The features used for the detection can be obtained from web server access logs. Hence, minimal extra effort is required to collect the features. The proposed detection approach is simple and computationally efficient enough to be deployed in real systems. We evaluate our proposed detection system using World Cup 98 dataset and simulated APP-DDos attack dataset. Similar flash crowd and APP-DDoS attack dataset is used in recent researches [5]. We further investigate the relevance of the features for the detection of APP-DDoS attack against legitimate flash crowds.

APP-DDoS DETECTION

The proposed APP-DDoS attack detection system has two stages. The stages are *feature computation from server access log and detection stage based on the computed features*. The input for feature computation stage is web server log data. Web server logs contain information about the requests made by clients. Server log information includes the client address, time stamp, URL of the requested object, reply size and client browser information. It is difficult to have accurate attack detection by considering only the information available on server logs. Some literatures suggested additional features that are derived from basic server log information [1, 2, 4, 5, 6, 18]. We have selected *Request rate (RR), page popularity (PP), request inter-arrival time (RIA), download rate (DR), and ratio of successful requests (RSR)*. The selection was done by looking into potential contribution of the feature for detecting DDoS attack and the computational requirement of the feature in terms of memory and processing time. The justifications for selecting the features are presented in Table 1.

All features are computed for each unique client by considering a predefined time interval called session time. Client is defined as the source of the request identified by IP address. Each client has its own unique IP address. The details of these features are discussed in Sub-section 3.1.

Table 1: Feature selection reason

Feature	Justification of choice
RR	Request flooding APP-DDoS attack is characterized by high number of requests per client whereas the number of requests per client is small for flash crowd. RR is selected to help detect Request flooding attacks from flash crowds.
PP	Legitimate users in a flash crowd tend to access popular pages more frequently because they look for similar news. However, APP-DDoS attacks request different pages randomly because if they choose few popular pages, they are forced to make many requests per page compared to normal users. This will make them easy target for request rate-based filters. The PP value of APP-DDoS attack is lower than PP value of flash crowd.
RIA	Normal users take some time to view a requested page before requesting the next object. APP-DDoS attack is generated by machines that do not need viewing time. So the request inter-arrival time is smaller for APP-DDoS attacks as compared to with legitimate users in a flash crowd.
DR	When a page is requested to the server, a disk access operation is performed. The disc access time depends on the size of the requested object. Large size web objects require higher disk access time. Large size web objects can be selected to conduct asymmetric APP-DDoS attacks [1]. But normal users do not intentionally request only large size web objects. This creates a difference in download rate between APP-DDoS and flash crowd.
RSR	APP-DDoS attackers may request web objects that do not exist in the server. This makes the sever to reply with 404 error message. Legitimate users in flash crowd, however, have very low probability of requesting an object that does not exist in the website. This creates a difference in RSR between flash crowds and APP-DDoS attack.

The input of the detection stage is the value of features computed in the feature computation stage. The expected output of the detection stage is either the client is legitimate or attack. In the detection stage, we put a supervised learning classifier to make a decision.

The mitigation stage could use information obtained from the detection stage to block any pending current and future APP-DDoS requests. The IP address of the attack client could also be added to a blacklist. Mitigation stage is not the focus of this research. Figure 1 shows the stages of the proposed approach.

Features

Server access log

Web servers register basic information about each request such as request address, time stamp, URL, request type, response code, replay size and user agent information. Each entry in a web access log represents one request. One example entry of apache web server access log is shown in Figure 2. The URL is relative to the web server's home directory. The time stamp has one second precision.

Feature computation

The features used in the detection of APP-DDoS attack from flash crowds are *Request rate (RR), page popularity (PP), download rate (DR), request inter-arrival time (RIA)* and *ratio of successful requests to total requests (RSR)*. The description and computation of the features are provided in the paragraphs below. All features are computed using a predefined session time (*T*).

Session time (*T*) is a time interval in which all requests that arrive in that interval are considered together when computing features.

Request rate (RR) is defined as the number of requests that arrive in a session time divided by session time. RR is computed for each unique client identified by its IP address. RR can be derived from server logs by counting the number of

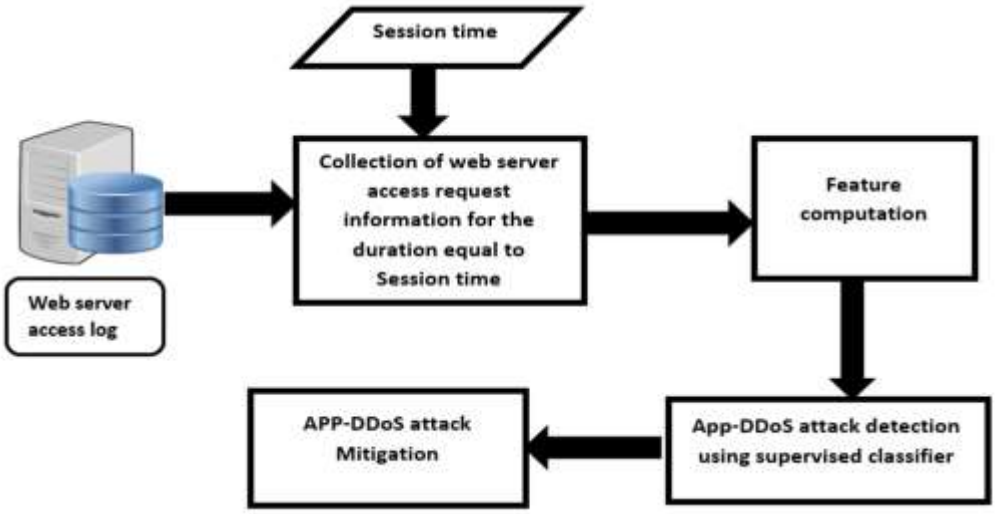

Figure 1: Proposed APP-DDoS attack detection and mitigation

requests served in T for each unique client. Equation 1 is used to compute request rate.

$$RR = \frac{N_r}{T} \quad (1)$$

where N_r is number of requests and T is the session time.

Page popularity (PP) is defined as the number of requests of a page or web object divided by total number of requests of all pages or web objects in the same website. A page or web object is any file that is identified by URL such as web page, image, audio, video, script, style sheet file and any other components of a website.

Before using page popularity for detection, we need to compute average popularity value of all web objects in a website. To compute average popularity, we will consider server log collected during normal operation of the website. From the collected server log, the page popularity value of each web object is computed using Equation 2.

```
::1 - - [17/Nov/2016:21:02:52 +0600]
"GET /PhpProject1/index.php HTTP/1.1"
200 2109 "-" "Mozilla/5.0 (Windows NT
10.0; WOW64; rv:39.0) Gecko/20100101
Firefox/39.0"
```

Figure 2: Example web server log entry

$$PP_j = \frac{NR_j}{NR_t} \quad (2)$$

where PP_j is page popularity of object j, NR_j the number of requests of object j and NR_t total number of requests of all objects in the considered period during normal operation.

In the detection stage, we will take the average page popularity value of each web object requested by the client in a session time. The average popularity value of each requested object in the session time (T) is summed up using Equation 3

$$PP = \sum_j NR_j * PP_j \quad (3)$$

where NR_j the number of requests of object j and PP_j page popularity of object j. All requested pages in the window time are considered in the summation. PP is computed for each client.

Download rate (DR) is defined as total number of bytes of reply of all requested objects in a session time divided by session time (T). When the requested object is not found on the server, the reply size is taken as zero. Equation 4 is used to compute download rate of each client.

$$DR = \frac{\sum_i Reply_i}{T} \quad (4)$$

Where $Replay_i$ the replay size in bytes for request i and T is session time.

Request inter-arrival time (RIA) is defined as the time duration between current request and previous request. The inter-arrival time between all requests in a session time are summed up. Equation 5 is used to compute request inter-arrival time of each client in a session time (T).

$$RIA = \sum_k (t_k - t_{k-1}) \quad (5)$$

Where t_k is time stamp of request k and t_{k-1} time stamp of the immediate predecessor request $k-1$.

Ratio of successful request to total requests (RSR) is defined as the ratio of requests with a reply code of 200 divided by total number of requests in session time. Requests with a reply code of 200 are considered as successful requests. RSR is computed for each unique client. Equation 6 is used to compute RSR. The value of RSR is between 0 and 1. A value of 0 means there is no successful request while a value of 1 means all requests are successful.

$$RSR = \frac{NRP}{NRT} \quad (6)$$

Where NRP is the total number of requests with 200 reply code that occur in session time (T) and NRT total number of requests in the session time (T).

Feature scaling

The values of each feature used in the detection system have different range. For example, the download rate is usually in the range of thousands while others are in the range of decimal fractions. Some classifiers such as decision tree and Adaboost does not require all the features to be in similar scale while Support Vector Machine requires all inputs to be on the same range [19].

We applied feature scaling in order to make the values of all features in a similar range by transforming feature distribution to a normal distribution with a mean of zero and unit standard deviation, we used Equation 7.

$$x_n = \frac{x - \underline{x}}{\sigma} \quad (7)$$

Where x_n is the transformed feature value, x is original feature value, $\underline{x}$ is the mean of all feature values and σ is the standard deviation of all feature values.

Detection

The detection system distinguishes APP-DDoS from flash crowd using a supervised learning classifier. The input to the classifier is an array of five feature values corresponding to RR, PP, DR, RIA and RSR respectively. The output of the classifier is either one or zero. One means the input feature vector corresponds to APP-DDoS while zero means the input feature vector corresponds to flash crowd.

The supervised classifier used for detection is trained off-line using examples of both flash crowd and APP-DDoS attack. The training data is composed of input features X_i and corresponding label Y_i. The input feature X_i is a vector of dimension five with components RR, PP, DR, RIA and RSR respectively. The output Y_i is a binary value that indicates weather the example represents DDoS attack or normal. Attack sessions will have a value of 1 while normal sessions will have a value of 0.

After the classifier is trained it can be deployed for detection to separate legitimate flash crowd from APP-DDoS attack. The output of the classifier is used as an input to the mitigation system. The mitigation system terminates current and pending requests of an attack client. It then adds the IP address of the attack client to blacklist. Any future connection attempts are also terminated. On the other hand, a request from a legitimate client is processed as usual.

EXPERIMENTS

In this section, we discuss evaluation of our proposed approach. We evaluate our proposed approach in terms of the following research questions.

[RQ1: APP-DDoS detection.] *Can we detect flash crowds from APP-DDoS attacks using our proposed approach?*

This research question helps us to evaluate our APP-DDoS detection system. More specifically, it deals with evaluation of our candidate classifiers and selecting the best classifier for our detection system.

[RQ2: Effect of session time.] *What is the effect of session time on APP-DDoS detection?*

In order to answer RQ2, we use the outperforming classifier from RQ1 to study the effect of session time on the classification performance of our detection system.

[RQ3: Feature contribution.] *What is the contribution of each feature for detection?*

In order to answer this research question, we perform qualitative and quantitative analysis to identify features that have higher contribution for detection.

Dataset preparation

In this sub-section, we discuss details of dataset preparation. We describe how we prepared datasets from World Cup 98 access log and experimentally generated APP-DDoS attack access log. Dataset generation involves data preparation of flash crowds and APP-DDoS.

World Cup 98 access log

World Cup 98 access log data is used in this experiment to model legitimate flash crowds. World Cup 98 dataset [20] consists of all the requests made to the 1998 World Cup Web site (www.france98.com) between April 30, 1998 and July 26, 1998. The World Cup website provided information about France 1998 World Cup during that period.

The website was hosted on multiple servers at different locations. The website received large number of requests from all clients who were interested in the World Cup game. 1,352,804,107 requests were received by the website during the specified period. Although this dataset is old, the characteristics of flash crowd that it models, is not different than what we would have as a flash crowd in these days. World Cup 98 data set is used as a flash crowd dataset in this and related recent researches [3, 8, 11, 12, 15, 21].

The server logs of the World Cup 98 website are provided in a binary format. The tools required to process the dataset are also provided [20]. World Cup 98 dataset is divided in to multiple files with more than one file per day. The number of files depends on the number of requests on that particular day. We have chosen day 66 (June 30, 1998) of the dataset to model flash crowds because it registered maximum number of requests. From the day 66 data, we have chosen server logs of the last three hours of the day. In these three hours, there was a game between Argentina and England, which included 30 minutes extra time, causing high number of requests to the website.

Each entry in the server log files represents a single request. The recorded information for each request is timestamp, clientID, objectID, size, method, status, type and server.

An example of the log entry is shown in Figure 3. The request information contains clientID, time stamp, request type and URL of the requested object, HTTP version, response code and reply size respectively from left to right. The IP address of the client is substituted by auto-generated ID number to keep anonymity.

APP-DDoS attack access log

To the best of our knowledge, there is no dataset available for APP-DDoS attacks. As a result, we generated APP-DDoS attack on a locally hosted version of the World Cup 98 website. The World Cup 98 website (www.france98.com) was hosted locally on closed environment. We performed APP-DDoS attack using a DDoS attack tool, BoNeSi [13]. BoNeSi can generate ICMP, UDP and HTTP

flooding attacks from pre-defined botnet size. This tool also accepts URL lists in a file and requests pages randomly. It also generates summary of the attack statistics.

The experimental setup used to generate APP-DDoS attack is as follows. BoNeSi tool is installed on attack machine and the cached version of World Cup website (www.france98.com) is hosted on the target machine. Apache web server application was used to host the website. The attack machine is directly connected to the server machine using cat-6 cable on its network card. BoNeSi attack tool is installed on Ubuntu 16.04 Linux operating system.

In order to conduct attack using BoNeSi tool, the response of the server must be routed back to the attack machine. To achieve this, the IP address of the default gateway of the server must be the same as the IP address of the attack machine. Request flooding and asymmetric attacks are included in the DDoS attack. In request flooding attack, attacker sends application layer requests such as HTTP GET request at higher rate than normal. Request flooding attacks are characterized by high number of requests per machine [1]. In asymmetric attacks, attacker uses requests that require high workload on the server and by making such multiple requests, the attacker easily crushes the server [1]. The request rate in asymmetric attack is usually very low to avoid detection. Repeated one shot attack is a special case of asymmetric attack and hence it is included as part of asymmetric attack. In repeated one-shot attack, the attacker sends requests that require high server workload in multiple secessions to avoid detection.

```
104858 - - [30/Jun/1998:21:41:24 +0000]
"GET /english/images/nav_home_off.gif
HTTP/1.0" 200 828
```

Figure 3: Example entry of World Cup 98 access log.

Request flooding attack was generated by sending large number of requests per source IP. This is achieved by limiting the maximum number of bots involved in the attack. BoNeSi provided 50,000 unique number of IP addresses to be used. In order to cover attack scenarios of very small and very large number of bots, 50 bots were taken for small number of bots and 50,000 were taken for large number of bots.

URL of requested object is randomly chosen from all web objects in the World Cup 98 website.

To simulate asymmetric APP-DDoS attacks, 50,000 bots were deployed. The total request rate is lowered so that the number of requests per bot is small. It is difficult to calculate precise request workload. We assume that the server load is proportional to the reply size.

This assumption works for static web pages whose contents are retrieved from hard drive. All pages on the World Cup 98 website are static pages. Fifty web objects with highest reply size are chosen as a target URL. BoNeSi randomly selects one URL at a time for the request.

The attack generation lasted a day. About 1GB of access log data was obtained after conducting the attack for a day.

Combined dataset

We merged World Cup 98 dataset access log, representing flash crowd, and APP-DDoS attack access log, representing APP-DDoS attack, to build our evaluation dataset. Since all candidate classifiers require a numerical input data, feature computation is required to convert access log dataset to the numerical dataset. As discussed in Section 3, the detection system uses five features for classification. The features are: *request rate, page popularity, download rate, request inter-arrival time and ratio of successful requests*. All features can be computed using equations discussed in Sub-Section 3.1 from server access log.

Each web-page of the World Cup 98 website has associated page popularity value computed using flash crowd access logs. The page popularity is obtained by summing the number of requests on Day 66 access logs for each page and dividing it by total number of requests. Equation 2 is used to compute the page popularity. Page popularity value of each web object is between 0 and 1.

During feature computation, we take the requested web object popularity value. If a client requests more than one web object or more than one request for similar web object in a session time, we use Equation 3 to compute the total page popularity.

A feature computation code is implemented using C++ language. The input to this code is server access logs of both attack and flash crowd. The output of the program is a CSV file. One line in the file represents the information of one client. It contains values of the five features and the label designating if the client is attacker (label value 1) or legitimate flash crowd (label value 0).

For example, the entry [0.35, 0.231039, 1950.65, 15, 1, 0] in the output file is read as *request rate, page popularity, download rate, request inter-arrival time, ratio of successful requests to total number of requests and label* respectively.

We have generated dataset for session time of 20, 40, 60, 120, 180, 240, 300, 360, 420, 480 seconds. Each dataset contains 20,000 entries of which half are flash crowd and the rest are APP-DDoS entries.

We have done experiments on Scikit-learn machine learning tool [22]. Scikit-learn learn is a python machine learning library that implements machine learning algorithms and provides API for training and testing. When training and testing classifiers, we used 10-fold cross validation technique.

RESULTS AND DISCUSSION

APP-DDoS detection

The result of RQ1 showed that it is possible to detect flash crowds from APP-DDoS attack using our proposed approach. The core part of our detection system is a supervised classifier. Although AdaBoost, random forest and decision tree classifiers have very close classification performance, decision tree outperformed all other tested classifiers considering classification time. Decision tree has F1-score of 99.45% and false positive rate of 0.47%.

In order to select a classifier for our detection system, we have tested GNB (Gaussian Naive Bayes), DT (Decision tree), SVML (SVM with linear kernel), SVMP (SVM with polynomial kernel), SVMR (SVM with radial basis kernel), Boost (AdaBoost) and random forest classifiers. We used the dataset generated using a session time of 20 seconds to compare the performance of the classifiers.

Figures 4 and 5 show the F1 and FPR scores of candidate classifiers on 20 second dataset respectively. GNB classifier showed the lowest F1 score of 90.97%. However, the best FPR was obtained by GNB. When lower F1 score is accompanied by lower FPR, it implies that most of the time the classifier guesses the input as flash crowd.

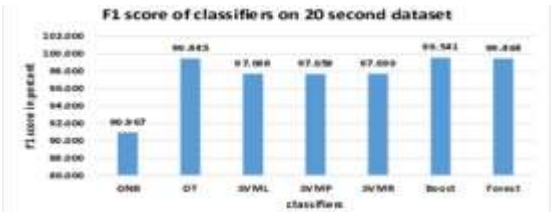

Figure 4: F1 score of candidate classifiers.

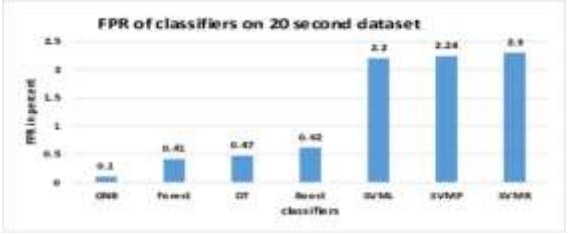

Figure 5: FPR of candidate classifiers.

As it can be seen in Figure 4, DT, Boost and Forest have F1 score higher than 99%. Boost has the highest F1 score of 99.541%. The F1 score of DT and Forest are also very close. We performed a statistical test whether the difference among scores of Forest, Boost and DT are statistically significant. We took the 10 F1 score values, obtained during 10-fold cross validation, of DT, Boost and Forest and performed analysis of variance (ANOVA) test. This test is a parametric test that requires normality check of each variable. We used the Kolomogorov-Simirnove (K-S) test of normality. The result shows that DT, Boost and Forest have a test statistic value of 0.171, 0.244, and 0.151, respectively. The corresponding P-values are 0.882, 0.51 and 0.95, respectively. The low test statistic value and high P-value (> 0.05) indicates that each distribution is not significantly different from normal distribution. ANOVA test shows that the difference among the three classifiers is not statistically significant with 95% confidence. The same is true when we do ANOVA on FPR score of the three classifiers. The K-S test of normality of FPR shows that DT, Boost and Forest have a test statistic value of 0.17, 0.23, and 0.18, respectively. The corresponding P-values are 0.82, 0.6, and 0.81, respectively. Since all P-values are above 0.05, we can apply ANOVA test on FPR. The result of the ANOVA test implies that *we can choose any classifier for our APP-DDoS detection among DT, Boost and Forest*.

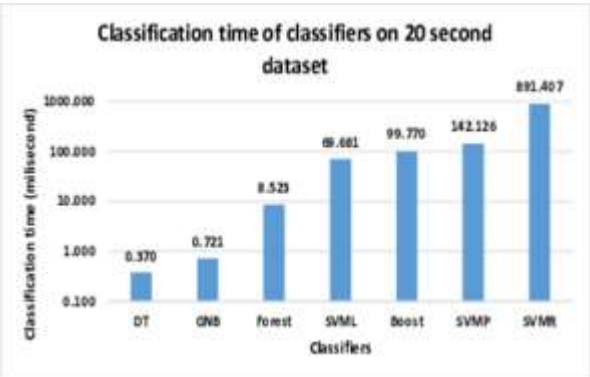

Figure 6: Classification time of candidate classifiers. (Note: The training time was not included in this measurement.)

To see the effect of kernels on SVM performance, we have used ANOVA to test the difference among F1 score of linear, polynomial and radial basis kernels. ANOVA test showed that the difference in F1-score among the three kernels is not statistically significant with 95% confidence. The K-S test of normality on F1-score shows that SVML, SVMP and SVMR have a test statistic value of 0.143, 0.133 and 0.18 respectively. The corresponding P-values are 0.97, 0.984 and 0.847 respectively.

Figure 6 shows the classification time of candidate classifiers. Classification time measures the time a classifier took to classify 10,000 examples in milliseconds. DT was the fastest classifier with 0.37 milliseconds. The classification time of DT is the smallest because DT mainly traverses a tree during classification. DT training has techniques to make the decision tree depth as small as possible. Traversing small depth trees requires small time. GNB is the second fastest with 0.721 milliseconds. The classification algorithm of GNB is relatively simpler compared to other classifiers. SVM and Boost took high classification time. SVM's require scaling of feature values which makes classification time longer compared to DT and GNB. Even though, Boost does not require scaling, Boost has to make fifty iterations to classify one example. This makes the classification time higher.

The kernel choice had big effect on the classification time of SVM. The computational complexity of SVM during classification is dependent on the complexity of the kernel. Radial basis kernel took 891.407 milliseconds while linear kernel took 69.68 milliseconds.

We have seen that DT, Forest and Boost showed comparable F1 score and FPR. But the classification time of DT is much smaller than Forest and Boost. APP-DDoS detection system must be computationally efficient in order not to contribute to the

already exhausted server resources. DT has bigger advantage compared to Boost and Forest when we consider classification time. That makes DT the recommended classifier to distinguish between flash crowd and APP-DDoS attack.

Effects of session time

To investigate effect of session time on the decision tree classifier detection performance, we tested decision tree classifier on data sets generated using 20, 40, 60, 120, 180, 240, 300, 360, 420 and 480 seconds session time. The result showed that the effect of session time on the performance of decision tree classifier is very small. We can choose the smallest session time of 20 seconds for our detection system without losing much in detection accuracy.

Figure 7 shows the effect of session time on F1 score of decision tree classifier. The highest F1 score was observed for 120 second session time. The difference between the highest and lowest F1 score is 0.275 %. This shows that the effect of session time on the F1 score is very small.

Figure 8 shows the effect of session time on FPR score of decision tree classifier. The FPR has even smaller variation among all session times. The difference of FPR among all session times is not statistically significant when we applied ANOVA. The K-S test of normality on FPR shows that all session times have test statistic value less than 0.33 and P-value of test statistics greater than 0.17. This shows that we can apply ANOVA test for the session times.

Session time has direct implication on the response time of the detection system. If the session time is smaller, then the detection system can respond quickly. When we see the difference between F1 score of 120 second, highest F1 score, and 20 second session time, it is only 0.2% and the FPR difference is 0.14%. As we can see, there is very little advantage gained by using 120 second session time compared to 20 second in terms of accuracy.

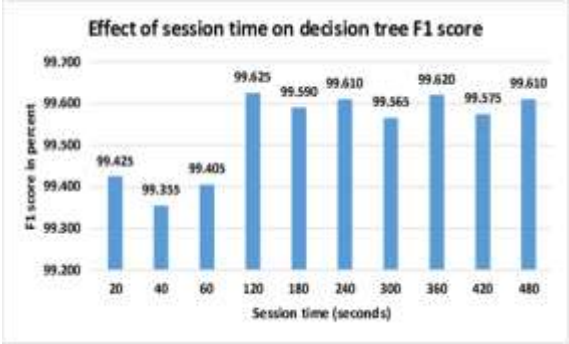

Figure 7: Effect of session time on F1 score of decision tree classifier.

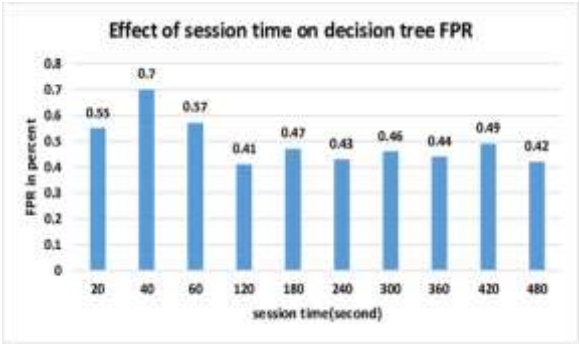

Figure 8: Effect of session time on the FPR score of decision tree classifier

For the smallest session time, 20 second, decision tree has F1 score of 99.425% and FPR of 0.55%. Based on the result obtained, we suggest using 20 second as a session time for feature computation in our detection system.

Feature contribution

The result of RQ3 showed that Request rate and download rate have higher contribution for detection among the five features based on qualitative analysis as well as experiment.

To investigate the contribution of each feature for detection, we have made a box plot for each feature using the 20 second data set (see Figure 9). Box plots provide insight on the contribution of each feature for detection through qualitative analysis. For easier visualization, we have normalized each feature value to a mean of zero and unit variance.

The median difference between APP-DDoS and flash is approximately 1 unit for request rate (see Figure 9 (a)). This shows that request rate contribution is potentially

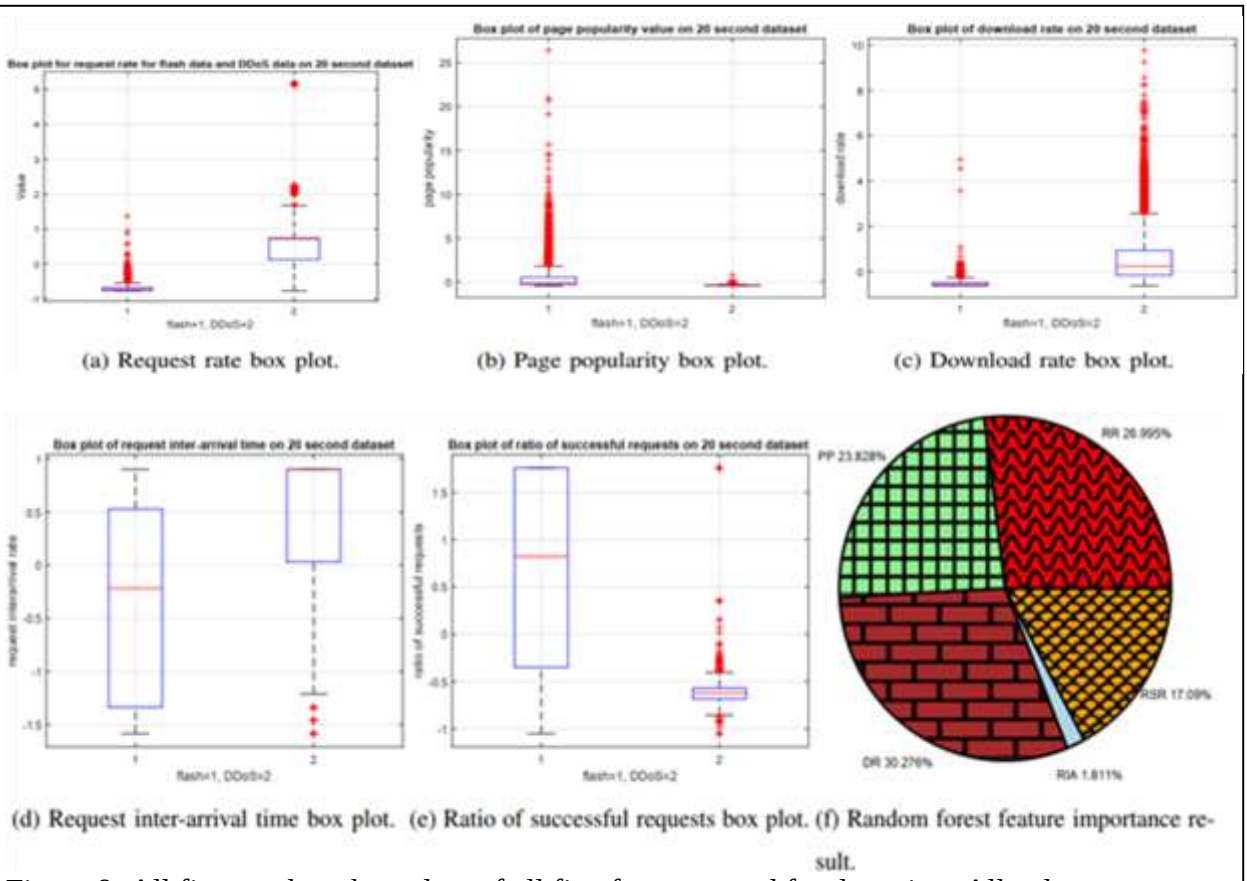

Figure 9: All figures show box plots of all five features used for detection. All values are normalized to have a mean of 0 and unit variance. The lower line of box plot represents the 25% or first quartile. The top of the box represents the third quartile. The line inside the box indicates the median. Values labeled in '+' sign are outliers.

higher. The reason for higher feature contribution is that most of the attack data is request flooding attack which is characterized by higher request rate.

The box plot for page popularity in Figure 9 (b) shows that the median of flash and APP-DDoS is very close and difficult to separate. This means that using page popularity only to separate APP-DDoS attack from flash crowds is difficult.

For the case of download rate in Figure 9 (c), the median difference between APP-DDoS and flash is approximately 0.8 units. Due to the asymmetric APP-DDoS attack, there are many outliers observed on the APP-DDoS box plot. The median difference is close to that of request rate.

Figure 9 (d) shows the box plot of request inter-arrival time (RIA). The difference between the median of APP-DDoS and flash was approximately 1.05 unit. However, there is high overlap between flash and APP-DDoS boxes. As a result, the potential contribution of RIA for detection is relatively low.

Figure 9 (e) shows the highest median difference between flash and APP-DDoS for ratio of successful requests (RSR), which are approximately 1.35 units. The big median difference occurred because legitimate users in flash crowds request pages by following links which increases the probability of the request being successful. But APP-DDoS attacks select pages randomly which reduces the probability of the request being successful. This shows that RSR has also higher contribution for detection.

To practically test the contribution of each feature, we used random forest classifier. After training the classifier with 20 second

data set, we obtained the feature importance value and plotted a pi-chart as shown in the Figure 9 (f).

The result obtained shows that download rate has the highest contribution of 30.276% and request rate is second with contribution of 26.995%. This result is in coherence with the qualitative analysis of feature importance.

Comparison with literature

To the best of our knowledge, the closest approach in terms of flash crowd and APP-DDoS dataset choice is the work of Daneshgadeh et al. [3]. Their approach leverages machine learning with information distance. They obtained a maximum of 100% recall and 93% precision which corresponds to F1-score of 96.4%. Our approach has 99.45% F1-score which is higher than Daneshgadeh et al.'s approach. They did not evaluate the computational complexity of their approach. Behal et al. [8] proposed an approach based on information theory. They used World Cup 98 dataset together with synthetically generated attack data. They achieved a true positive rate of 95%. It is not possible to compare their approach because we used different APP-DDoS attack dataset.

Threats to validity

a) Internal threats to validity

Threats to internal validity are mainly caused by variation in instrumentation, and effect due to uncontrolled variables. The instrumentation used to measure variables is computer. We run all experiments on the same computer to avoid instrumental variation. The effect of uncontrolled variables is mainly observed when we measured the classification time of classifiers. The classification time may be affected by external concurrent processes that run on the same computer at the time of the experiment. To address this threat, we have repeated the measurement ten times and took the average. In addition, we closed non-vital applications during experiment.

b) External threats to validity

The proposed approach is evaluated on specific flash crowd data set and using only one attack tool. In addition, the flash crowd data set is old which may not represent current flash crowds. Those are major threat to external validity. But our proposed approach is independent of the data. We can test our approach on any data set. We will reevaluate our approach when recent flash crowd data set is available. In addition, we can test our approach on any website without changing our detection system. The other problem is that we only found one DDoS attack tool suitable for our research. But the tool is very flexible with many configurable parameters. We tried to approximate the functionality of other DDoS attack tools by manipulating the configuration. This makes the attack tool more representative.

c) Construct threats to validity

The main threats to construct validity occur during choice of features for detection and during choice of classifiers. For example, we evaluated six classifiers from all supervised classifiers. But the best classifier may not be among the candidate classifiers. To minimize threats of construct validity because of classifier selection, we selected representative examples from most commonly used supervised machine learning algorithms. Most other supervised classifiers are derivatives of the candidate classifiers. In addition, we did not consider deep learning based classifier as it requires big data for training and we already have good results using other supervised classifiers.

The other potential threat to construct validity is the choice of features. We did not consider all possible features for detection. The reason for this is that if we choose a feature that cannot be computed from our data set, it is difficult to evaluate our proposed approach. But we have

obtained very good result using only six features by systematically choosing features that have higher contribution for detection. When we compute we did not consider requests with 300 response code (redirect). The reason for this is that we cannot determine if the redirected requests are successful or not from our server access log data. This may introduce some bias on the results.

d) Conclusion threat to validity

The main threats to conclusion validity are too small sample size, measurement error and violation of assumption in test statistics. We have 20,000 examples for both flash and APP-DDoS in our data set. When we observe both APP-DDoS attack and flash crowd data set, the feature values are similar or very close to each other. We believe that our data set sample size is not small for our problem. Since the measurement and experiment was done on computers, the measurement error only comes from computation errors from machines. Hence, the measurement error is negligible. We have used analysis of variance (ANOVA) as test statistics. The assumption of ANOVA is that the data must be normally distributed. In order not to violate this assumption, we tested our data for normality using Kolmogorov-Smirnov test. The result showed that the data used in the ANOVA test is normally distributed.

CONCLUSIONS

In this paper, the problem of identifying application layer DDoS attacks from legitimate flash crowds is addressed. The researchers proposed a supervised machine learning based detection system that uses request rate, page popularity, download rate, request inter-arrival time and ratio of successful requests as features to distinguish between APP-DDoS attack and flash crowds.

Six supervised classifiers are evaluated on our dataset. F1 Score and false positive rate are used as classification performance evaluation criteria. Classification time is also used as computational complexity evaluation criteria to compare the classifiers. The results show that it is possible to identify APP-DDoS attack from flash crowd with our proposed approach.

Decision tree (DT) outperformed other candidate classifiers considering a combination of F1 score, FPR and classification time as evaluation criteria. DT classifier has 99.445% F1 score, 0.47% FPR and the smallest classification time of 0.37 milliseconds. This shows that DT is a good candidate for the detection system.

Variation of session time has very small impact on the performance of decision tree classifier. The difference between F1 scores when 20 second and 120 second session times are used is very small. In addition, the difference between FPR scores of 20 and 120 seconds session time for decision tree classifier is not statistically significant. This implies that any session time can be chosen with very small impact on performance of the detection system.

From the proposed features, download rate has the highest contribution for detection followed by request rate and page popularity.

Future work

The main limitation of the research is the unavailability of latest data set of flash crowds. For this research, the World Cup 98 data set is used. The World Cup 98 data set is the standard application layer flash crowd data set up to now even though it was recorded before 19 years. Our proposed approach should be tested on latest data set for more concrete and applicable result. The other limitation was the unavailability of APP-DDoS data set which forced us to use DDoS attack generation tool. Based on the aforementioned limitation of this research, the following points are recommended to be addressed as a future work.

The features used for training as well as prediction are computed per session, but the researchers believe that tracking users usage history will contribute to a more robust detection. In the future, this could be used to address the impact of users history on the performance of the detection approach.

The proposed approach should be tested on a new data set that contains examples of real flash crowds and APP-DDoS attacks. We did not find standard criteria to generate application layer DDoS attack in simulation. Some standard should be set on how to generate APP-DDoS attack that closely resembles real attacks. This can be done by analyzing patterns of real world APP-DDoS attacks.

In our work, due to the limitation of our dataset, we did not consider low rate APP-DDoS attacks. One way to account for low rate APP-DDoS attack is to borrow information from TCP layer about the time it takes to complete a single request. Low rate APP-DDoS attack usually take more time to complete a single request. Furthermore, those attacks usually send partial requests so data about request content may be another clue. One extension, of our work could be to combine information from application layer and TCP layer to effectively handle low rate APP-DDoS attacks.

REFERENCES

[1] Ranjan, S., Swaminathan, R., Uysal, M. & Knightly, E. W., *DDoS-Resilient Scheduling to Counter Application Layer Attacks Under Imperfect Detection.*, in INFOCOM, 2006.

[2] Xu, C., Zhao, G., Xie, G. & Yu, S., *Detection on application layer DDoS using random walk model*, in 2014 IEEE International Conference on Communications (ICC), 2014.

[3] Daneshgadeh, S., Ahmed, T., Kemmerich, T. & Baykal, N., *Detection of DDoS attacks and flash events using Shannon Entropy, KOAD and Mahalanobis Distance*, in 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), France, 2019.

[4] Wang, J., Yang, X. & Long, K., *A new relative entropy based app-DDoS detection method*, in Computers and Communications (ISCC), 2010 IEEE Symposium on, 2010.

[5] Patil, M. M. & Kulkarni, U. L., *Mitigating App-DDoS Attacks on Web Servers*, International Journal of Computer Science and Information Security **9**, 40 (2011).

[6] Yadav, S. & Selvakumar, S., *Detection of application layer DDoS attack by modeling user behavior using logistic regression*, in Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on, 2015.

[7] Sahoo, K. S., Tiwary, M. & Sahoo, B., *Detection of high rate DDoS attack from flash events using information metrics in software defined networks*, in 10th International Conference on Communication Systems and Networks (COMSNETS), Bengaluru, India, 2018.

[8] Behal, S., Kumar, K. & Sachdeva, M., *A generalized detection system to detect distributed denial of service attacks and flash events for information theory metrics*, Turkish Journal of Electrical Engineering and Computer Sciences, 1759-1770 (2018).

[9] Behal, S., Kumar, K. & Sachdeva, M., *D-Face: An anommaly based distributed approach for early detection of DDoS attacks and flash events*, Journal of Network and Computer Application, 49-63 (2018).

[10] Khalaf, B. A., Mostafa, S. A.,

Mutapha, A. & Abdullah, N., *An adaptive model for detection and prevention of DDoS and flash crowd flooding attacks*, in International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR), Malaysia, 2018.

[11] Bhandari, A., Sangal, A. L. & Kumar, K., *Characterizing flash events and distributed denial-of-service attacks: an empirical investigation*. Security and Communication Networks **9**, 2222-2239 (2016).

[12] Yu, S., Zhou, W., Jia, W., Guo, S., Xiang, Y., & Tang, F., *Discriminating DDoS attacks from flash crowds using flow correlation coefficient*, IEEE Transactions on Parallel and Distributed Systems, vol. 23, pp. 1073-1080, 2012.

[13] Markus-Go, *BoNeSi - the DDoS Botnet Simulator*, 2016. https://github.com/Markus-Go/bonesi. Accessed August 2019.

[14] Yu, J., Li, Z., Chen, H. & Chen, X., *A detection and offense mechanism to defend against application layer DDoS attacks*, in Networking and Services, 2007. ICNS. Third International Conference on, 2007.

[15] Xie, Y. & Yu, S.-Z., *Monitoring the application-layer DDoS attacks for popular websites*. IEEE/ACM Transactions on networking **17**, 15-25 (2009).

[16] Ye, C. & Zheng, K., *Detection of application layer distributed denial of service*, in Computer Science and Network Technology (ICCSNT), 2011 International Conference on, 2011.

[17] Ramamoorthi, A., Subbulakshmi, T. & Shalinie, S. M., *Real time detection and classification of DDoS attacks using enhanced SVM with string kernels*, in Recent Trends in Information Technology (ICRTIT), 2011 International Conference on, 2011.

[18] Xie, Y. & Yu, S.-Z., *A novel model for detecting application layer DDoS attacks*, in Computer and Computational Sciences, 2006. IMSCCS'06. First International Multi-Symposiums on, 2006.

[19] Graf, A. B. & Borer, S., *Normalization in support vector machines*, in Pattern Recognition: 23rd DAGM Symposium, Munich, Germany, September 12-14, 2001. Proceedings, 2001.

[20] Arlitt, M. & Jin, T., *1998 World Cup Web Site Access Logs*, (1998).

[21] Yu, S., Thapngam, T., Liu, J., Wei, S. & Zhou, W., *Discriminating DDoS flows from flash crowds using information distance*, in NSS 2009: Proceedings of the third International Conference on Network and System Security, 2009.

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.