Fingerprinting Crowd Events in Content Delivery Networks: A Semi-supervised Methodology

Amine Boukhtouta[1(✉)], Makan Pourzandi[1], Richard Brunner[2], and Stéphane Dault[3]

[1] Ericsson Security Research,
8275 Trans Canada Route, Saint-Laurent, Montréal, QC, Canada
{amine.boukhouta,makan.pourzandi}@ericsson.com
[2] Ericsson Universal Delivery Network,
8275 Trans Canada Route, Saint-Laurent, Montréal, QC, Canada
richard.brunner@ericsson.com
[3] Ericsson Business Area Digital Services, R&D Security Operations,
8275 Trans Canada Route, Saint-Laurent, Montréal, QC, Canada
stephane.dault@ericsson.com

Abstract. Crowd events or flash crowds are meant to be a voluminous access to media or web assets due to a popular event. Even though the crowd event accesses are benign, the problem of distinguishing them from Distributed Denial of Service (DDoS) attacks is difficult by nature as both events look alike. In contrast to the rich literature about how to profile and detect DDoS attack, the problem of distinguishing the benign crowd events from DDoS attacks has not received much interest. In this work, we propose a new approach for profiling crowd events and segregating them from normal accesses. We use a first selection based on semi-supervised approach to segregate between normal events and crowd events using the number of requests. We use a density based clustering, namely, DBSCAN, to label patterns obtained from a time series. We then use a second more refined selection using the resulted clusters to classify the crowd events. To this end, we build a XGBoost classifier to detect crowd events with a high detection rate on the training dataset (99%). We present our initial results of crowd events fingerprinting using 8 days log data collected from a major Content Delivery Network (CDN) as a driving test. We further prove the validity of our approach by applying our models on unseen data, where abrupt changes in the number of accesses are detected. We show how our models can detect the crowd event with high accuracy. We believe that this approach can further be used in similar CDN to detect crowd events.

1 Introduction

CDNs are the global networks delivering content from different content providers to cope with the increasing demand for the QoE required by the commercial

F. Kerschbaum and S. Paraboschi (Eds.): DBSec 2018, LNCS 10980, pp. 312–329, 2018.
https://doi.org/10.1007/978-3-319-95729-6_20

content providers. To address the ever increasing demand for content in the Internet, CDNs turned out to be the de-facto solution to cache content, including video streaming, news, and social media [5]. CDNs are meant to accelerate the delivery of content on the Internet to cope with the business-grade performance. As such, their importance increased within the cyberspace ecosystem over time. A recent report stipulates that 70% of all Internet traffic will cross CDNs by 2021 [4].

Among the key players in networks deployment, CDNs have been facing many challenges from the complexity and versatility of emerging online services. Thus, CDNs are exposed to benign events such as crowd events and cyber-threats like DoS, DDoS and harmful crawling of cached assets. The CDN operators are therefore increasingly interested in the prediction of faulty events in CDNs resulting from misconfigurations, unpredictable networking conditions, or the result of cyber-attacks. In recent years, sophisticated malicious artifacts are used by attackers to take advantage of any vulnerability to conduct sabotaging CDN itself or target critical infrastructures to cause service unavailability. By metamorphosing CDNs to support security as a built-in asset to counter different cyber-threats have become then of a paramount importance to operators. As part of this effort, there is a keen interest shown operators to investigate events logging data for identifying misbehavior of CDNs. Crowd events (flash crowds) are simultaneous and huge access to web or media based content from legitimate users. There have been several efforts to predict the DDoS attacks based on analyzing the event logs. However, few works targeted to distinguish between the benign crowd events from DDoS attacks. However, this distinction is of high importance to avoid false positive DDoS attacks and better planning of resources to address legitimate users during crowd events. To this end, we aim at addressing the problem of framing crowd events in CDNs and differentiate them from unsolicited/malicious activities by exploring CDN's data obtained from a large operator. In this research effort, we aim to provide an answer to the following questions: (1) What are the key indicators identified in CDNs ecosystem? (2) Given observable crowd events, how to profile them and isolate them from normal events? (3) By considering profiles, how to use engineered features to distinguish between crowd events and anomalies?

To answer these questions, we shape the contributions of this paper as follows: (1) We draw upon 169 GB of logging data collected from a large CDN operator to characterize access events in a hybrid CDN, where web and media assets are cached. The number of events is more than 452 million events (more than 386 million access events, whereas the rest are routing events). We present different perspectives to engineer features, namely, delivery, cache, IP and HTTP based features. (2) We propose a semi-supervised methodology to identify crowd events with high detection rate on the training dataset (99%). The methodology is driven by the number of requests to profile patterns in a timely manner (time series). By using a density clustering algorithm (DBSCAN), we manage to create profile normal and crowd accesses. The clustering plays the role of a first filter layer towards crowd events fingerprinting. The resulted labeling is then used for classification (XGBoost) which subsequently identifies the crowd events. We manage to identify two patterns of crowd accesses patterns that can be considered as a ground truth to potentially identify anomalies. (3) To test further our methodology, we used anomalous unseen data to test classifiers. We showed that our methodology allowed to discern crowd events and anomalies. Thus, we believe our methodology can be used to create multi-level time series classification system to identify anomalies in CDN's deployment.

The remainder of this paper is as follows: In Sect. 2, we explain the methodology used to discern crowd and normal accesses, as well as the features set used to characterize. Section 3 puts forward a description of the dataset and experiments layout as well as results obtained from them. In Sect. 4, we expose the different related works as well as how they compare with our work. In Sect. 5, we conclude with a few observations and future directions of our research.

2 Methodology

2.1 Overview

The reason behind showing an interest to crowd events, lies in the fact that they tend to be frequent over time, therefore, prone to be fingerprinted in comparison with cyber-attacks like DDoS, where data needs to be recorded during an on-progress attack (e.g., [6]), or inferred from network telescopes (e.g., [10,11]), or even simulated through attacking tools (e.g., [12]). Our strategy is to characterize thoroughly crowd events through number of accesses, then differentiate them from anomalies based on attributes collected from different perspectives (delivery, IP, cache, HTTP). Figure 1 depicts our approach. We pre-process the input logs and events to extract patterns representing aggregated counters collected during a time window. We then use them to train a first model to discern between the normal and crowd accesses (fingerprinting component in Fig. 1). Subsequently, the

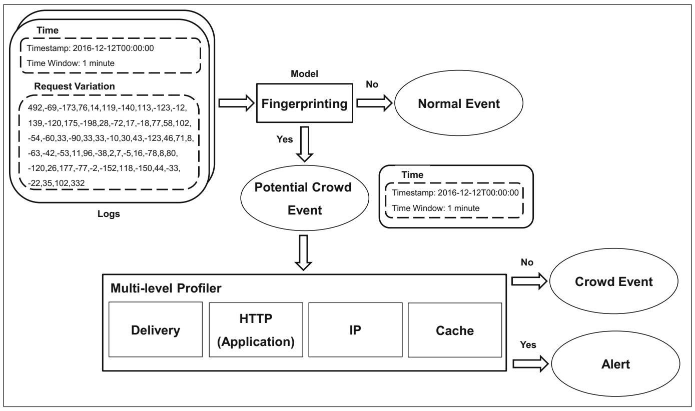


Fig. 1. General approach.

potential crowd events are subject to a more in-depth multi-level profiling to check whether the event is a real crowd event or an alert (Multilevel profiling component in Fig. 1). In our approach, we use different perspectives to aggregate several features (attributes) into new features allowing to detect crowd events. Then, we use collected aggregates as a downstream outcome to fingerprint crowd events. We refer readers to Sect. 2.3, where we describe the different features considered in our work.

2.2 Fingerprinting Crowd Events

Figure 2 illustrates the fingerprinting methodology, which is a semi-supervised approach, where we use empirically a density clustering algorithm to label crowd and normal accesses. Based on that, we create a detection model to identify labeled patterns. As such, we apply a data-driven approach based on logs collected from CDN's operator. The logs are used to compute different attributes indexed per time. Given a time granularity and aggregation window, we create patterns, which are fed to a density clustering algorithm to segregate between normal and crowd events. However, obtained solution does not allow to balance between normal and crowd accesses cardinalities. Therefore, we employ data augmentation technique to increase the number of crowd accesses patterns. This is done to balance the number of normal access patterns with crowd access patterns. Afterwards, we label the balanced data to create a ground truth for classification. The latter's result is a model that represents a decision system that discerns crowd events from normal events. It is important to mention that all these steps are done offline. In the sequel, we detail different components.

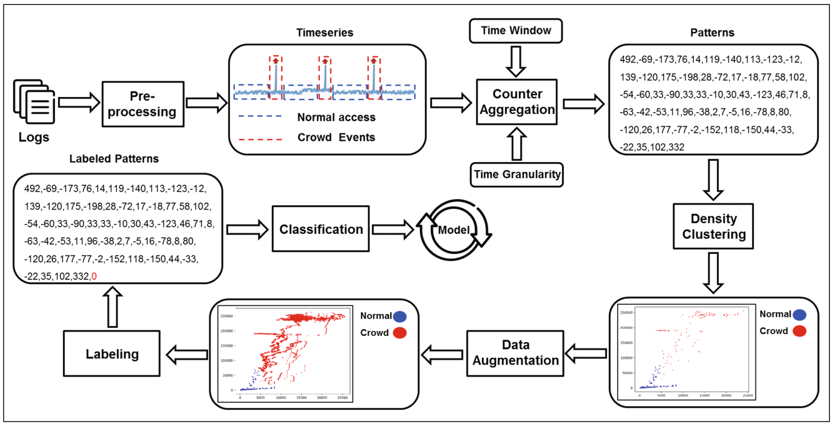
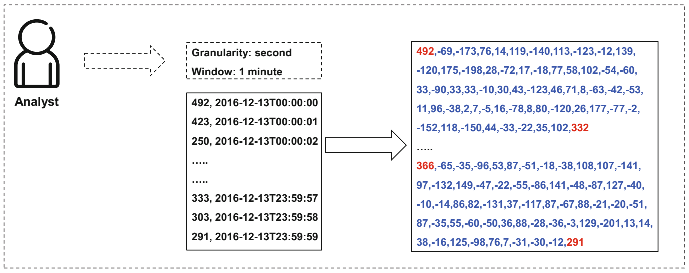


Fig. 2. Detailed view of the fingerprinting approach.



```
492, 2016-12-13T00:00:00
423, 2016-12-13T00:00:01
250, 2016-12-13T00:00:02
.....
.....
333, 2016-12-13T23:59:57
303, 2016-12-13T23:59:58
291, 2016-12-13T23:59:59
```

```
492,-69,-173,76,14,119,-140,113,-123,-12,139,
-120,175,-198,28,-72,17,-18,77,58,102,-54,-60,
33,-90,33,33,-10,30,43,-123,46,71,8,-63,-42,-53,
11,96,-38,2,7,-5,16,-78,8,80,-120,26,177,-77,-2,
-152,118,-150,44,-33,-22,35,102,332
.....
366,-65,-35,-96,53,87,-51,-18,-38,108,107,-141,
97,-132,149,-47,-22,-55,-86,141,-48,-87,127,-40,
-10,-14,86,82,-131,37,-117,87,-67,88,-21,-20,-51,
87,-35,55,-60,-50,36,88,-28,-36,-3,129,-201,13,14,
38,-16,125,-98,76,7,-31,-30,-12,291
```

Fig. 3. Aggregation example.

Aggregation. By aggregation, we mean encompassing observed raw values during a granularity time unit (e.g., 1 s) into one value observed during a time window period (e.g., 1 min). As such, the result of the aggregation is a data point that reflects a statistical view of raw values, which can be a count, sum, average, etc. In the context of our work, we consider initially the number of access requests to CDN, indexed per second. It is important to mention that the aggregation period ca be adjusted, but, needs to be selected carefully to obtain a rich set of patterns to fingerprint crowd events. In this work, as a first attempt, we consider two aggregation time windows, namely, 1 min or 5 min. Other aggregation periods can be considered, although the aggregation period is longer (e.g., 10 min), less is the number of collected patterns. Figure 3 illustrates an aggregation pattern recorded for 1 min aggregation time window. A pattern is represented by a starting and an ending value, as well as, differences (shifts) between values observed every second.

Density Clustering. Density clustering is meant to segregate between high and low density data, thus, we assume that crowd events happen less in comparison with normal events. As such, we consider using this clustering technique to characterize normal events as a highly dense data (highly dense core cluster), whereas crowd events are seen like low dense data (low dense clusters or outliers). Based on prior usage of DBSCAN [3] in different works [13–15], we exploit it to cluster data collected from accesses aggregates. DBSCAN algorithm is based on two parameters, namely, the radius distance and the minimum core number. Given sampled data points, the algorithm iteratively looks for other data points located within radius distance to create a cluster. If two data core points are close within the radius distance, they are merged into the same cluster. Based on the minimum core number, the algorithm sets a minimal cardinality to create a cluster. Points that are not enclosed within clusters, are considered as outliers or singletons. In our case, we target mainly to group normal accesses in a core cluster and crowd accesses in low density clusters or singletons. Thus, we set the minimum core number to 1, to find data points representing crowd events as singletons. As a distance function, we use Euclidean distance between points. We use the silhouette score to evaluate the quality of clustering solution.

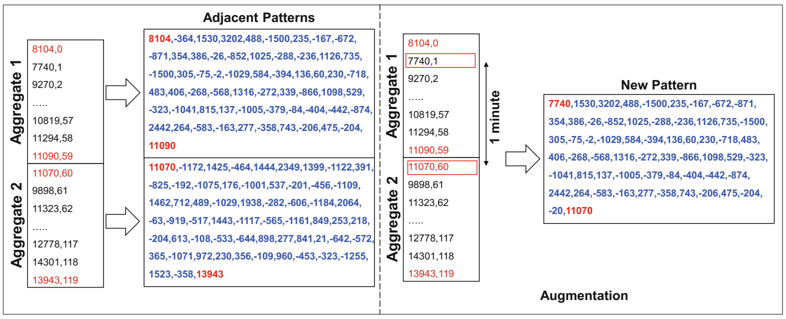


Fig. 4. Augmentation example.

Data Augmentation. Being inspired by works from [16–18], we employ data augmentation on time series. The main reason behind doing so, lies in the fact of unbalance between normal accesses and crowd accesses. With this intent, low density clusters and singletons representing crowd access patterns are used to create new patterns. Timely adjacent patterns are used to extract new patterns to create a balanced dataset between normal accesses and crowd accesses. We use a sliding window (e.g., 1 s) to extract a new pattern, Fig. 4 depicts an example of two adjacent patterns aggregated during 1 min and used to obtain a new pattern by utilizing a sliding window of 1 s.

Labeling and Classification. As a downstream outcome from clustering solution and data augmentation, we label the core cluster patterns representing normal accesses with 0, whereas singleton and augmented patterns are labeled with 1. Thus, a labeled dataset is created, and used as an input for a binary classifier. The latter is built by applying the XGBoost algorithm [1], which is based on optimization through tree models [19,20] and boosting [21]. It supports many learning and boosting parameters that can be used to build classification models. XGBoost has three loss functions to control prediction, namely, Mean Square Error for regression, Log-Loss for binary classification and mLog-Loss for multi-classification. XGBoost uses regularization functions to control the complexity of the model to avoid over-fitting. Both loss function and regularization terms to define the objective function. The latter is optimized by using the gradient descent algorithm to compute gradients. XGBoost builds the boosting tree by computing predictions of leaves and greedily finding splitting points optimizing the objective function. In [22], the authors enumerated the advantages of XGBoost: (1) Tree models have rich representational abilities. (2) The boosting is adaptive, thus, models are flexible to determine neighborhoods in different parts of the input space. (3) Bias-Variance trade-off control, XGBoost starts with low variance and high bias model and reduces the bias accordingly by decreasing the size of neighborhoods in the input space.

Table 1. Features description.

Perspective	Feature	Description
Delivery	Hit ratio	The number of caching hits divided by the total number of requests during a time period
	Volume sum	The sum of bytes saved by caching during a time period, it is negative if content assets are fetched from the origin
	Volume average	The average value of volume records during a time period
	Volume deviation	The standard deviation of volume records during a time period
	Volume minimum	The minimum value of volume records during a time period
	Volume maximum	The maximum value of volume records during a time period
	Duration average	The duration average for content assets delivery during a time period
	Duration deviation	The standard deviation of content assets' delivery duration during a time period
	Duration maximum	The maximum value of delivery duration records during a time period
IP	Number of requests	The total number of requests observed during a time period
	Number of distinct IPs	The total number of requesting unique IPs during a time period
	Maximum requesting IP	The maximum number of requests issued by the most accessing IP during a time period
	IPs entropy	The number of unique IPs divided by number of requests issued by IPs during a time period
	Average request per IP	The average value of requests issued per IP during a period of time
	Deviation request per IP	The standard deviation value of requests issued per IP during a period of time
Cache	Number of distinct caches	The total number of unique caches serving requests during a time period
	Maximum requested cache	The maximum number of requests observed on the most serving cache during a time period
	Caches entropy	The number of unique caches divided by number of requests served by caches during a time period
	Average request per cache	The average value of requests served by a cache during a period of time
	Deviation request per cache	The standard deviation value of requests served by a cache during a period of time
HTTP	Ratio status	The HTTP status ($20X$, $40X$, $50X$) observed requests divided by the total number of requests
	Ratio HTTP method	The HTTP method (GET, POST, HEAD, DELETE, PUT) observed requests divided by the total number of requests

2.3 Features Engineering

The features engineering aims to the creation of attributes from the domain knowledge, namely, log traces collected from CDN deployment. Based on internal experts' inputs, we define four perspectives based on field attributes: delivery perspective, IP perspective, cache perspective, and HTTP perspective (see Table 1). We consider these metrics as an increase in the delivery duration metrics indicates a bandwidth saturation indicating a possible crowd event. From a delivery perspective, we are interested in: (1) the hit ratio through counting how frequent the content assets are found in caches, (2) the cached volume through computing saved bytes (content objects' size) as well as different statistical metrics, and (3) the delivery duration of different content assets as well as the average, the standard deviation and the maximum values. As the delivery perspective asses the effectiveness of CDNs, we propose to use those features for fingerprinting.

The IP perspective is meant to monitor clients requesting content from CDN. The metrics associated with IPs help to describe the dynamics of accessing content, thus, they are potential indicators for DDoS attacks or massive contents' crawling. From IP perspective, we are interested in: (1) the total number of requests produced by clients (IP addresses), (2) the total number of distinct IP addresses observed during aggregation time, and (3) the maximum number of requests generated by the most occurring IP address, (4) IPs entropy score, which represents the total number of distinct IP addresses divided by the total number of requests, (5) the average of requests' number generated per IP, and (6) the standard deviation of requests' number generated per IP.

The cache perspective represents how cached content is served to clients instead of accessing the content from origin servers. As such, being aware how content is distributed can help detection of caching anomalies. For instance, the distribution of requests through caches pinpoints to how fairly or unfairly requests are distributed to caches. Low cached volume metric indicates potential high number of requests to unpopular content indicating possible DDoS events. From cache perspective, we are interested in: (1) the total number of distinct caches serving requests during aggregation time, (2) the maximum number of requests observed on the most serving cache, and (3) caches entropy score, which represents the total number of distinct caches divided by the total number of requests, (4) the average of requests' number served by a cache, (5) the standard deviation of requests' number served by a cache.

The HTTP perspective is meant to be aware of the application protocol used to request content from CDN. A drastic change in the number of POST or GET can indicate the presence of a flooding attack; thus a misuse of HTTP protocol. From HTTP perspective, we are interested in: (1) the ratio of HTTP status codes (e.g., 200 or 404), and (2) the ratio of HTTP methods (e.g., *GET*, *POST*). At the end, we discard some constant features: (1) the minimum delivery value since it tends to 0, (2) the number of requests from the less requesting IP addresses since it tends to 1, and (3) the number of requests observed on less accessed caches, which tends to 1.

3 Experiments

3.1 Experiments Setup

We run the experiments on a virtual machine deployed on Intel Xeon CPU E5-2060 2 GHz, consisting of 12 virtual CPUs and 32 GB of memory. The experiments are done on a dataset collected from the 12th to 19th of December 2016. It represents Web access logs collected from a large operator hosting a sport league website. The size of the data is 169 GB of logs, which corresponds to 386, 396, 885 access events. We enumerate 1, 268, 160 IPs spanning over 200, 634 "/16" subnets, geo-located in 219 countries and 15, 646 cities. The crowd events were observed on 14th and 15th of December 2016, whereas anomalies were observed the 12th of December 2016. Figure 5 represents the distribution of requests' number between 13th to 19th December 2016. We notice two peaks of the request number in the 14th and 15th of December, these peaks represent crowd accesses during games. We consider then data collected from 13th to 19th December 2016 to cluster normal and crowd accesses. The clustering is done on the patterns as described in Sect. 2.2. Once the clustering is done, we augment the patterns collected from the original data, then, label different patterns to create the classifier. To build the latter, we use a 10 rounds' classification process with a 5-fold cross validation. To test our approach, we apply the classifier on the anomalous day (12th of December 2016, not used for the training) to check, if the model detects abrupt changes. As such, we can use patterns extracted from other features, to see which ones can segregate between crowd accesses and anomalies.

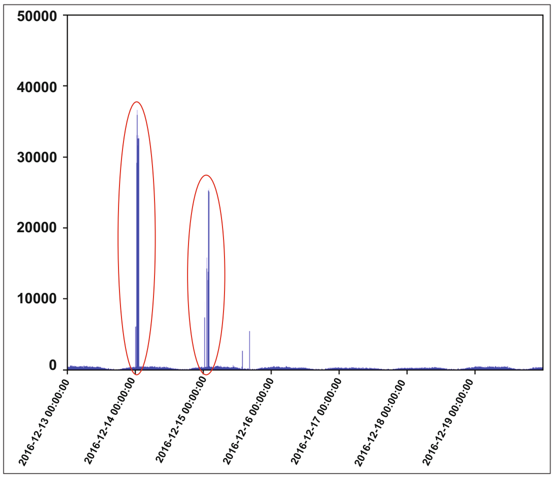


Fig. 5. Number of requests (13th to 19th December 2016).

3.2 Clustering and Augmentation Results

We apply DBSCAN algorithm on patterns extracted from two aggregation periods (1 min and 5 min). The intent is to find empirically a core cluster grouping normal accesses, and discerning crowd accesses. To do so, we tweak the distance parameter of DBSCAN and check the quality of clustering solutions through silhouette scores and how many patterns are enclosed in the core cluster. The intent is to find a distance parameter that produces a good quality of clustering, meanwhile segregating crowd accesses from normal ones. For each distance, we compute clustering execution time, silhouette score and core clustering coverage. Figure 6 depicts clustering running time, silhouette score and core cluster coverage with respect to distance parameter, which varies from 5 to 65 for 1 min patterns, and from 5 to 110 for 5 min patterns. The running time to build clustering solutions spans from 10.7 to 28.25 s for 1 min patterns, and from 2.69 to 5.01 s for 5 min patterns. The clustering running time average is 14.2 s for 1 min patterns and 3.71 s for 5 min patterns. Regarding silhouette scores, we observe that it tends to 1, which means that clustering quality is good and therefore no need to increase the distance beyond the current distances used for our experiments. The core cluster is discernible, since the majority of patterns are grouped together (core cluster covers the majority of patterns). We observe that the silhouette scores increase when the distance gets higher, but we need to monitor a trade-off between high silhouette scores and missing patterns representing crowd accesses. To illustrate this trade-off, we consider two clustering solutions for both 1 min (distances equal to 25 and 65, silhouette scores equal to 0.855 and 0.931, coverage values equal to 9965 and 10055) and 5 min (distances equal to 80 and 110, silhouette scores equal to 0.877 and 0.909, coverage values equal to 1995 and 2002) patterns (see Fig. 6).

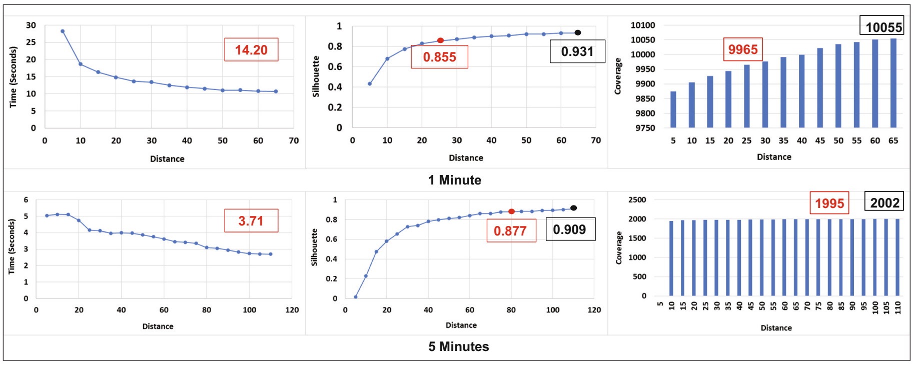


Fig. 6. Clustering: running time & silouhette scores & core cluster coverage.

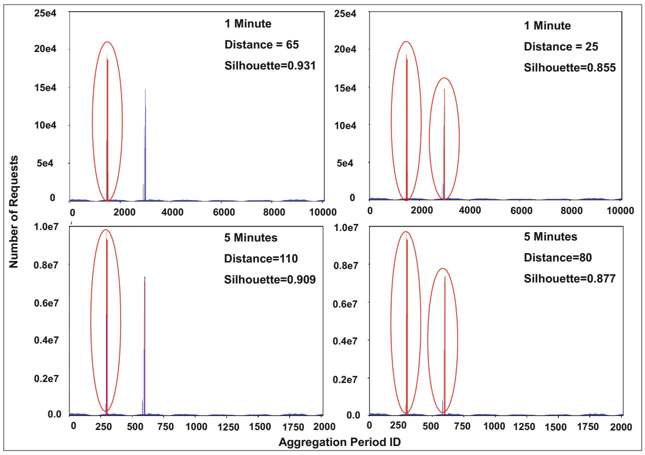


Fig. 7. Silhouette & distance vs. Crowd events identification.

We refer to Fig. 7 to depict the trade-off between solutions for different patterns. Based on observations inferred from aforementioned illustration, we select two solutions, meaning a distance equals to 25 for 1 min patterns and a distance equals to 80 for 5 min patterns. Despite the fact that these distances do not output the best silhouette score, their values manage to segregate better between core cluster patterns (normal accesses) and singleton patterns representing crowd accesses (see circled peaks in Fig. 7). In the second case observed in Fig. 7, where distance values are respectively 65 and 110 for 1 min and 5 min patterns, the silhouette scores are slightly better, but the clustering solutions do not segregate effectively between normal and crowd accesses. We can observe

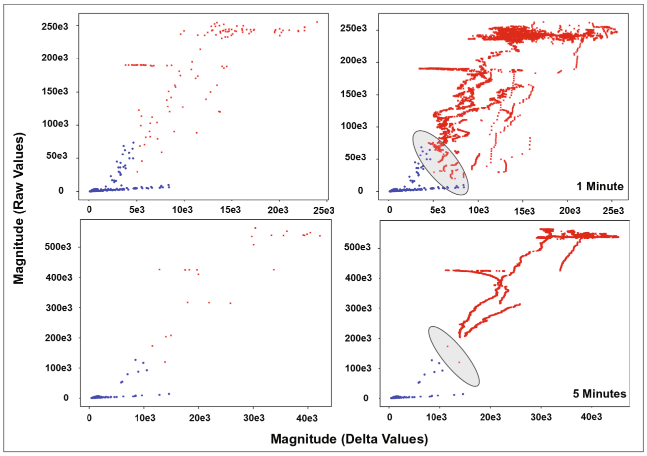


Fig. 8. Patterns augmentation (1 min & 5 min).

that some peaks are not distinguishable from normal accesses (the peaks that are not circled in Fig. 7).

We apply augmentation to balance between the patterns belonging to the core cluster (normal accesses) and other patterns (crowd accesses). The motivation behind doing so, is to infer more crowd access patterns from existing patterns and scale their cardinality with normal access patterns in order to label them for classification. Figure 8 illustrates the plotting of magnitude values of raw values and difference values within patterns. Data points at the bottom of both plots represent the core cluster, whereas data points in the middle up to the top of plots represent crowd accesses. Augmentation of crowd accesses can be observed in the right hand side, where new patterns are created to balance classification dataset.

Table 2 shows the number of patterns before and after augmentation. In this experiment, we randomly select some adjacent crowd access patterns and infer augmented data. For 5 min aggregation period, data augmentation is used for both normal and crowds accesses to increase the number of samples to more than $5,000$; whereas for 1 min aggregation, we consider increasing the number of crowd accesses patterns since we already have more than $9,000$ patterns for normal accesses. However, the data augmentation process is to subject for refinement, since we can infer more normal access and crowd patterns, thus, increasing number of patterns in the classification dataset. Moreover, we need to carefully label patterns in the border between normal and crowd accesses. This is depicted in the grey zone illustrated within the right hand side of Fig. 8, where normal and crowd accesses can be mixed, therefore a potential over or under fitting of the classification model can take place.

Table 2. Number of patterns before and after data augmentation.

Patterns	Normal access (before)	Crowd access (before)	Normal access (after)	Crowd access (after)
1 min	9,965	115	9,965	7,015
5 min	1,992	24	5,893	5,409

3.3 Classification Results

We apply the XGBoost algorithm by considering its default execution layout. We use first and second order gradients (*grad* and *hess*) by applying logistic transformation (sigmoid) [2] on *LogLoss* function. To evaluate trained models, we consider *stacking*, an ensemble learning technique, where the predicted value is computed from cross validation. The number of learning rounds is 10, where the number of folds within the dataset is 5. The evaluation metrics are: (1) the Area Under Curve (AUC) of Receiver Operating Characteristic (ROC) function, which represents the trade-off between sensitivity (fall-out) and specificity (recall), and (2) the accuracy average for both training and testing.

Table 3. 10 rounds of 5 fold cross validation results (Tr. Training Phase, Te. Testing Phase).

Period	AUC Mean (Tr.)	AUC Std (Tr.)	AUC Mean (Te.)	AUC Std (Te.)	Acc. Mean (Tr.)	Acc. Mean (Te.)
1 min	%99.9128	%0.0144	%99.7489	%0.0766	%99.8274	%99.6370
	%99.9552	%0.0168	%99.8066	%0.0964	%99.8542	%99.6489
	%99.9814	%0.0075	%99.8740	%0.1108	%99.1070	%99.6608
	%99.9831	%0.0048	%99.9125	%0.1063	%99.9137	%99.6371
	%99.9838	%0.0050	%99.9200	%0.1187	%99.9420	%99.6906
	%99.9861	%0.0037	%99.9442	%0.0773	%99.9509	%99.7144
	%99.9861	%0.0037	%99.9665	%0.0528	%99.9524	%99.7263
	%99.9872	%0.0033	%99.9661	%0.0536	%99.9628	%99.7203
	%99.9945	%0.0040	%99.9667	%0.0529	%99.9673	%99.7204
	%99.9980	%0.0026	%99.9663	%0.0533	%99.9702	%99.7322
5 min	%99.9915	%0.0217	%99.9739	%0.0214	%99.9911	%99.9735
	%99.9915	%0.0217	%99.9739	%0.0214	%99.9911	%99.9735
	%99.9915	%0.0217	%99.9739	%0.0214	%99.9911	%99.9735
	%99.9915	%0.0217	%99.9739	%0.0214	%99.9911	%99.9735
	%99.9915	%0.0217	%99.9739	%0.0214	%99.9911	%99.9735
	%99.9915	%0.0217	%99.9739	%0.0214	%99.9911	%99.9735
	%100	%0.0217	%99.9738	%0.0214	%99.9911	%99.9735
	%100	%0.0217	%99.9738	%0.0214	%99.9911	%99.9735
	%100	%0.0217	%99.9736	%0.0216	%99.9911	%99.9735
	%100	%0.0217	%99.9736	%0.0216	%99.9911	%99.9735

Table 3 depicts classification results for both 1 min and 5 min patterns. Each row contains a round of 5 fold cross validation, where we consider AUC mean and standard deviation, as well as accuracy mean. These metrics are computed for both training and testing phases. From results observed in the table, we notice that for each round, AUC statistics are maintained, since the mean tends to 1, whereas the standard deviation tends to 0. We also observe that the accuracy is high and tends to 1 for both training and testing. Regarding 5 min patterns classification, the results are constant; consequently, any round can be considered. Regarding 1 min patterns classification, the results change slightly from a round to another. Usually, the model with the highest accuracy rate, or with the lowest difference between AUC mean and standard deviation (best sensitivity and specificity trade-off), can be selected. As such, we can consider models obtained from the 10^{th} round, which has the best AUC and high accuracy rate metrics. To test their detection of abrupt changes, the models are then tested on unseen data (12^{th} of December). Figure 9 illustrates patterns detected by models as abrupt changes in the number of requests for two training days (14^{th} and 15^{th} of December 2016) and unseen data (12^{th} of December 2016).

Regarding crowd accesses, we notice 2 types of patterns illustrated in the top and bottom plots within Fig. 9 (dashed ellipses). The first crowd accesses' patterns illustrate a continuous periodic access to a sport event, whereas the second ones illustrate some crowd accesses at the beginning of the game, then, another set of crowd accesses during up to the end of the game. This can be explained as people followed up the first game continuously at the opposite of

Table 4. Classification runtime (milliseconds) per pattern.

Patterns	Minimum	Maximum	Average	Deviation
1 min	0.715971	3.186941	0.749865	0.103742
5 min	0.961065	2.573013	1.008195	0.123094

the second one. In the latter, people were more interested to know what is the issue of the game than following it continuously. The abrupt changes present in the middle plot are different than the aforementioned crowd accesses' patterns. As such, we will consider studying other attributes during the period, where we observed crowd accesses and suspicious patterns. Regarding the classification runtime, we compute the time taken to predict each pattern. Table 4 illustrates different statistics observed on classification runtime expressed in Milliseconds. We notice that the average time to classify 1 min pattern is in the range of

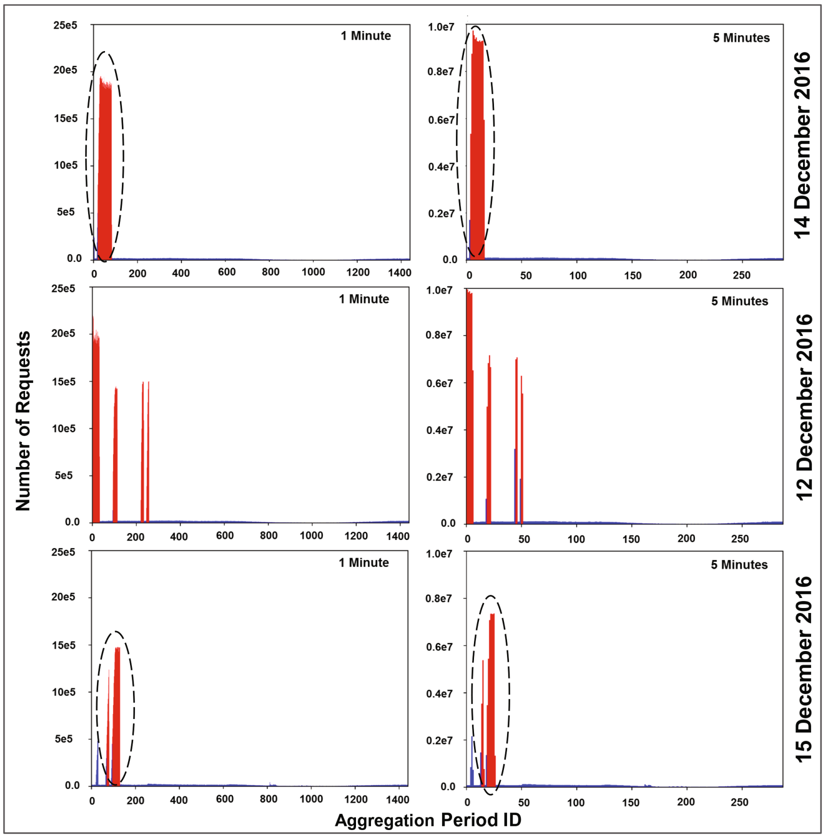


Fig. 9. Prediction on 12th 14th 15th December 2016 Patterns (1 min & 5 min).

0.75 ms, whereas it is in the range of 1 ms for 5 min pattern. The standard deviation is in the range of 0.1 ms for both 1 min and 5 min pattern.

4 Related Work

Several works considered studying abnormal access patterns to the web sites in order to detect DDoS attacks. In [23], authors analyzed IBM Olympic Games Web site, and developed models to predict seasonal patterns based on peak request rates and traffic variation. The study did not consider the implication of CDNs, or DoS attacks. In [24], authors studied a peak workload analysis of the football World Cup 1998 Web site [9]. They focused on the reference of few extremely popular webpages, where clients inter-session time were short. The authors considered the workload observed on the world cup website as an initial characterization of how future workloads look like. They profiled HTTP server response codes, type of content, unique file distribution and, files' reference behavior (temporal locality and concentration of references). In [28], the authors proposed a behavior based detection that can discriminate DDoS attack traffic from traffic generated by real users. Their detection method relies on the repeatable features of the packet arrivals. They used Pearson's correlation coefficient to define a segregation threshold between predictable and non-predictable data. They used [8,9] datasets to test thresholds defined from simulated inter-arrival time data. This work did not consider CDNs' logs and consider traffic flows. The inter-arrival time can be subject to other networking constraints like congestion, type of content (e.g. Video) and cached and non-cached web objects. In addition, as any correlation analysis, an error metric needs to be considered to support threshold decision. In [29], the authors used also Pearson coefficient on users' activity through number of requests. In [30], the authors introduced a method to detect application-layer DDoS attack based on the entropy of HTTP GET requests per source IP address. They used adaptive auto-regressive model to transform time series into a multi-dimensional vector, then, applied SVM classification to identify the attacks. The authors utilized NS-2 simulator to create attacks ground truth and considered World Cup 1998 Web dataset [9] for crowd events. The approach is promising, however, the use of simulated data can be biased or noisy. None of the mentioned studies considered patterns to build a crowd events detection model.

In [25], the authors considered crowd events analysis, where they studied two HTTP log traces collected from a popular TV show (24 h) and Chilean election site (approximately 33 h). They showed that number of clients was in the proportion of request rate. They studied also the number of clients' clusters during crowd events, the clusters overlap, and request rate, as well as reference to files access. They also considered datasets representing password cracking and five web servers disabling traces to characterize DoS attacks. They looked at the same perspectives, clients and files, and drew upon results some differentiation between crowd events and DoS. They proposed an enhancement to CDNs, namely, adaptive CDN, by using collected trace-driven simulation to study their enhancement. Despite the fact the study considered clustering clients, it has not investigated temporal aggregation of counters to create a crowd event detection model. In [26], the authors examined usage patterns, files' characteristics (popularity and referencing), transfer behaviors of YouTube, and compare them to traditional Web and media streaming workload. The data were collected from a university network, where staff and students accessed two Youtube's points of presence. This work focused more on usage patterns and file referencing without elaborating predictive tasks. In [27], the authors differentiated DDoS and crowd access flows by considering the fact that generated flows from DDoS tools can be fingerprinted (high level of similarity), whereas crowd accesses are randomly distributed (low level of similarity). The authors used Jeffrey distance, Sibson distance, and Hellinger distance to measure the similarity among flows. They concluded that Sibson distance is the most suitable, after applying experiments on two distinctive datasets, Aukland VIII [7] representing crowd events and Lincoln Laboratory DDoS scenario [8]. Despite the fact that the approach is interesting, they used old datasets (collected on 2003 and 1999 respectively). As explained above, none of the previous works, consider using a temporal set of patterns as we use in our approach to detect crowd events.

In [31], the authors applied a discrimination algorithm based on a similarity metric, namely, entropy variations to identify suspicious flows. They formulated the problem in the Internet with botnets, and presented theoretical proofs for the feasibility of their method. In this work, the authors relied on simulations to prove their approach. For our work, we used a recent dataset collected from a major operator, and applied a semi-supervised approach to profile crowd accesses.

5 Conclusion and Future Work

The distinction between crowd events and DDoS attacks is difficult, making it of an increasing interest to CDN operators. In this paper, we applied a semi-supervised approach on a sport league dataset collected from a major operator to identify normal and crowd access patterns. The patterns are represented by number request shifts during 1 and 5 min. We first used DBSCAN to group normal accesses into a core cluster, a crowd accesses into low dense clusters and singletons. By applying data augmentation, we balanced classification vectors representing 1 min and 5 min patterns. Then, we utilized XGBoost to fingerprint crowd and normal accesses. The results of the classification (99% accuracy) showed the great potential of our approach. We tested our approach by applying it to unseen data. The approach detected abrupt changing patterns, even though these change patterns do not have the same shape like the ones identified in the training dataset. We believe our approach can be successfully used to detect crowd events in other CDN environments. Despite of our initial good results, the diversity of CDN environments would necessitate more investigation. We frame our future works to consider other features described in Sect. 2.3 to distinguish anomalies (e.g., DDoS) from crowd events. We will rely on patterns found on unseen data to carry on this research. In addition, we plan to tweak XGboost models to study the trade-off between their complexity and performance. Moreover, we want to thoroughly test the classification model on additional data, as well as deploy it in online mode.

References

1. Tianqi, C., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2016)
2. Chen, T., He, T., Benesty, M.: xgboost: Extreme gradient boosting, pp. 1–4. R package version 0.4-2 (2015)
3. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 96, no. 34, pp. 226–231 (1966)
4. Cisco, Cisco Visual Networking Index: Forecast and Methodology 2016–2021. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html
5. Stocker, V., Smaragdakis, G., Lehr, W., Bauer, S.: The growing complexity of content delivery networks: challenges and implications for the internet ecosystem. Telecommun. Policy **41**, 1003–1016 (2017)
6. The CAIDA UCSD "DDoS Attack 2007" Dataset. http://www.caida.org/data/passive/ddos-20070804_dataset.xml
7. WITS: Waikato Internet Traffic Storage. https://wand.net.nz/wits/auck/8/
8. Lincoln Laboratory MIT. DARPA Intrusion Detection Evaluation. https://www.ll.mit.edu/ideval/data/2000/LLS_DDOS_1.0.html
9. The Internet Traffic Archive, WorldCup98. http://ita.ee.lbl.gov/html/contrib/WorldCup.html
10. Fachkha, C., Bou-Harb, E., Debbabi, M.: Fingerprinting internet DNS amplification DDoS activities. In: The 6th International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5. IEEE (2014)
11. Rossow, C.: Amplification hell: revisiting network protocols for DDoS abuse. In: Proceedings of the 21st Network and Distributed System Security Symposium (NDSS) (2014)
12. Moustis, D., Kotzanikolaou, P.: Evaluating security controls against HTTP-based DDoS attacks. In: 2013 Fourth International Conference on Information, Intelligence, Systems and Applications (IISA). IEEE (2013)
13. Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data. ACM (2006)
14. Manh, T.T., Kim, J.: The anomaly detection by using DBSCAN clustering with multiple parameters. In: 2011 International Conference on Information Science and Applications (ICISA). IEEE (2011)
15. Shahaboddin, S., Amini, A., Anuar, N.B., Kiah, M.L.M., Teh, Y.W., Furnell, S.: D-FICCA: a density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks. Measurement **55**, 212–226 (2014)
16. Le Guennec, A., Malinowski, S., Tavenard, R.: Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (2016)

17. Howard, A.G.: Some improvements on deep convolutional neural network based image classification. arXiv preprint arXiv:1312.5402 (2013)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
19. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. SSS. Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7
20. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
21. Breiman, L.: Arcing classifier (with discussion and a rejoinder by the author). Ann. Stat. **26**(3), 801–849 (1998)
22. Nielsen, D.: Tree Boosting With XGBoost-Why Does XGBoost Win Every Machine Learning Competition? MS thesis. NTNU (2016)
23. Iyengar, A.K., Squillante, M.S., Zhang, L.: Analysis and characterization of large-scale web server access patterns and performance. World Wide Web **2**(1–2), 85–100 (1999)
24. Arlitt, M., Jin, T.: A workload characterization study of the 1998 world cup web site. IEEE Netw. **14**(3), 30–37 (2000)
25. Jaeyeon, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. In: Proceedings of the 11th International Conference on World Wide Web. ACM (2002)
26. Phillipa, G., Arlitt, M., Li, Z., Mahanti, A.: Youtube traffic characterization: a view from the edge. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, pp. 15–28. ACM (2007)
27. Yu, S., Thapngam, T., Liu, J., Wei, S., Zhou, W.: Discriminating DDoS flows from flash crowds using information distance. In: Proceedings of the 3rd IEEE International Conference on Network and System Security (NSS 2009), 18–21 October 2009 (2009)
28. Thapngam, T., et al.: Discriminating DDoS attack traffic from flash crowd through packet arrival patterns. In: 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE (2011)
29. Chuan, X., Du, C., Kong, X.: An application layer DDoS real-time detection method in flash crowd. In: IACSIT Hong Kong Conferences, pp. 68–73 (2012)
30. Ni, T., Gu, X., Wang, H., Li, Y.: Real-time detection of application-layer DDoS attack using time series analysis. J. Control Sci. Eng. **2013**, 4 (2013)
31. Prasad, K.M., Munivara, K., Reddy, A.R.M., Rao, K.V.: Discriminating DDoS attack traffic from flash crowds on internet threat monitors (ITM) using entropy variations. Afr. J. Comput. ICT **6**(2), 53 (2013)