NetFlow Anomaly Detection Though Parallel Cluster Density Analysis in Continuous Time-Series

Kieran Flanagan, Enda Fallon, Paul Connolly, Abir Awad

NetFlow Anomaly Detection Though Parallel Cluster Density Analysis in Continuous Time-Series

Kieran Flanagan[1,2], Enda Fallon[1], Paul Connolly[2], Abir Awad[3]

[1]Software Research Institute, Athlone Institute of Technology, Athlone, Ireland
[2]The NPD Group, Inc, IDA Business Park, Athlone, Co. Westmeath, Ireland
[3]Faculty of Computing, Engineering and Science, University of South Wales, UK

k.flanagan@research.ait.ie, efallon@ait.ie, paul.connolly@npd.com,
abir.awad@southwales.ac.uk

Abstract. The increase in malicious network based attacks has resulted in a growing interest in network anomaly detection. The ability to detect unauthorized or malicious activity on a network is of importance to any organization. With the increase in novel attacks, anomaly detection techniques can be more successful in detecting unknown malicious activity in comparison to traditional signature based methods. However, in a real-world environment, there are many variables that cannot be simulated. This paper proposes an architecture where parallel clustering algorithms work concurrently in order to detect abnormalities that may be lost while traversing over time-series windows. The presented results describe the NetFlow activity of the NPD Group, Inc. over a 24-hour period. The presented results contain real-world anomalies that were detected.

Keywords: Anomaly Detection, NetFlow, Clustering, Density Analysis

1 Introduction

In recent years, research into new methods of anomaly detection within a network has increased in prominence. The need for a fast, reliable method to identify possible malicious activity has grown in response to emerging threats. Protecting confidential and proprietary data is of paramount importance to any organization to ensure that both legal and contractual obligations are kept. In addition, the data stored may not necessarily be the property of the company storing and handling the data. Malicious activity such as Botnets and Port Scans are increasing in frequency. These attacks, while simple, have the potential to allow for unauthorized access onto the network.

Within any organization, it is common place to use network monitoring and analysis tools to help with the detection of any anomalous behaviour on the network. Tools, such as McAfee ePO and Tipping point for example, are signature based models, which require a known example of a threat to be catalogued and a signature generated. The signature based model, while highly exact, fails if a novel attack occurs (e.g. zero-day vulnerabilities), since no previous signature exists. This limitation gave rise to anomaly based detection mechanisms. These methods require no signature database, but instead models the "normal" traffic on a network and alerts to any activity that happens outside of these normality bounds.

While much research has been conducted on various methods for anomaly based systems using a variety of approaches [1–5], key limitations apply when attempting to adapt these approaches to a real-time system. These include, most notably, computational

cost. Within commonly used distance based outlier detection mechanisms, the need for distance based calculations for each new sample can be overwhelming for high volume data. This gave rise to optimized algorithms designed to mitigate this limitation. Algorithms such as Fast Local Outlier Factor (FastLOF) [6] and Micro-Cluster based Outlier Detection (MCOD) [7], reduce the overall cost of the range queries with varying degrees of success.

It has been shown that the application of time series can be beneficial in the detection of network anomalies. Applying time series over time-windows of increasing size has been shown to be capable of normalizing normal behaviours over time. However, at smaller time-intervals, it is possible for abnormal behaviour to traverse time windows, allowing for the possibility of becoming a false positive. This is particularly prevalent among anomalies that generate low numbers of NetFlow. It is possible for it to become hidden within other network traffic as time progresses, making detection increasingly unlikely.

Moreover, while it is possible that a large increase may occur, small deviations in established traffic behaviour may also be an indication of unauthorized activity. For example, an increase in failed login attempts may produce little difference with respect to NetFlow, it may be indicative of someone trying to guess a password. Detecting such an instance would be of paramount importance, particularly if followed by a successful login attempt [8]. This paper proposes a solution to this problem. By implementing parallel clustering algorithms, it is possible to gain a higher level of granularity while maintaining the normalization techniques gained from an incrementally increasing time-window. Concurrent algorithms can detect minor deviations from established behaviour that can occur, regardless if they occur while traversing between time-windows.

In Section 2, an overview of related work is given. In Section3, a brief overview of the technology used is presented. In Section 4, we propose a framework for the identification of anomalies within NetFlow data. The architecture is presented as well as an overview given on algorithms created and used. In Section 5, testing methodology is presented. In Section 6, results obtained from live data are presented and analysed. In Section 7, conclusions and future work are presented.

2 Related Works

Recent research into anomaly detection has largely focused on applying anomaly detection mechanisms on network data to successfully identify anomalous behaviour. Many problems still exist however. Performance is a key factor when trying to utilize anomaly detection techniques and there are many examples where this is apparent. Methods such as Principal component analysis [3], K nearest neighbour [9] and ensemble techniques [10], [11] have been used to various degrees of success in this task.

However, there comparatively expensive operations have led to a rise in clustering techniques to mitigate the calculations need when associating anomaly detection with big data [12], [13]. Using aggregated data, such as NetFlow can be used to reduce the calculations further [14]. Limitations are present with these techniques however. Kumari et al [15] looks at a clustering technique for the use of anomaly detection over a network, setting a distance based threshold as the 100th farthest data point from the obtained cluster centroids. This threshold is a common theme across multiple anomaly detection

solutions [16–18]. However, it can be argued that using a common threshold over all clusters within real data is non-representative of the various forms of traffic created. E.g. Traffic from different applications do not act in a similar manner.

This brings forward an interesting problem. While distance based outliers have been shown to be of significance in a plethora of works [19–21], we propose another indicator of possible anomalous behaviour. By monitoring cluster density over a time series, changes in underlying behaviour can be detected. Rather than only focusing on samples that are anomalous via distance based calculations, changes in the density of activity over time are also monitored.

Asmuss et al [18] demonstrates the use and effectiveness of utilizing a time series based approach over live data. The aggregation of traffic is highly beneficial in this case, as it reduces the computational sources needed. Furthermore, it also provides a tangible benefit when comparing results across clusters. This idea of time-aware analysis has been used elsewhere also [22], and has shown to be a valuable tool in mapping continuous behaviour.

This ability to generate a normalized view of traffic over time, practically speaking, has some limitations however. The potential for an anomaly to traverse though sequential time windows can lead to the anomaly threshold not being broken, thus leading to a false positive reading. Presented in Section 4 is an architecture that utilizes concurrent time windows in order to mitigate the risk of this happening. Anomalies can be gathered from individual instances of the clustering algorithm, while the correlation of clustering behaviours through instances can also indicate anomalous activity.

3 Technology Overview

3.1 Cisco NetFlow

Cisco NetFlow is a system that amalgamates network traffic information into a format that successfully describes communications occurring on a network. Through a NetFlow enabled device (Fig. 1), packet traces are identified and stored as a single flow representation of a specific set of communications between two devices. These are used for multiple tasks such as network performance monitoring and used as a means of security evaluation when an incident has been detected. Visualization of the NetFlow has also proven to be of tangible benefit [23]. Using this aggregated data for anomaly detection has numerous benefits, such as data size being reduced for processing purposes and storage. For the analysis in this paper, NetFlow was used as it was found that using NetFlow for network monitoring purposes was highly common in the area[24], as well, in this case, the current infrastructure of NPD allowed for the collection of NetFlow with relative ease.

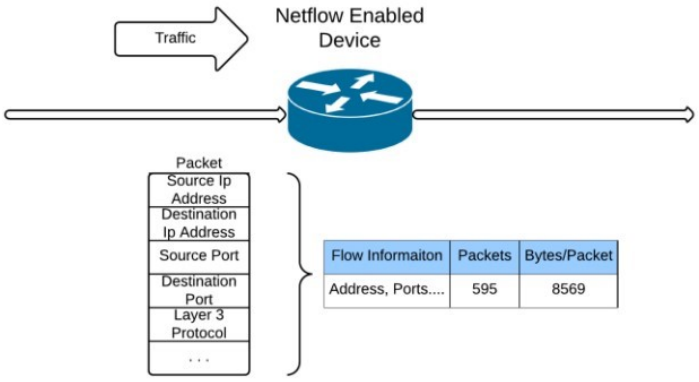

Fig. 1. NetFlow Example

3.2 MCOD

MCOD is a distance based outlier detection mechanism that utilizes clustering to reduce the amount of distance based calculations needed to identify possible anomalies. This reduced performance cost of the algorithm makes it an ideal candidate for real-time anomaly detection over data streams when compared to other distance based algorithms such as Local Outlier Factor (LOF). As described in [7], MCOD uses a sliding window approach to identify outliers over a data stream. By using an expiring data set, the algorithm can be optimized to only use a data set that is large enough for satisfactory anomalies to be detected, while maintaining the low amount of calculations needed. Distance is calculated between cluster centroids and the NetFlow sample being queried. If the point is within this range of a cluster centroid and the cluster has the specified density, k, then the point is determined to be a non-anomaly, or inlier.

4 Proposed Framework

The architecture is a two-step approach that involves monitoring traffic at different levels of abstraction. Firstly, an adaptation of the MCOD algorithm is applied on the NetFlow data in sequential time windows (Fig. 2). This stage can outline distance based outliers contained within the NetFlow information. Following this, the clusters generated within each time-window are correlated to identify those representing similar traffic. The density values (how many input samples are contained within the cluster) are then gathered at the end of each window 5 minute period.

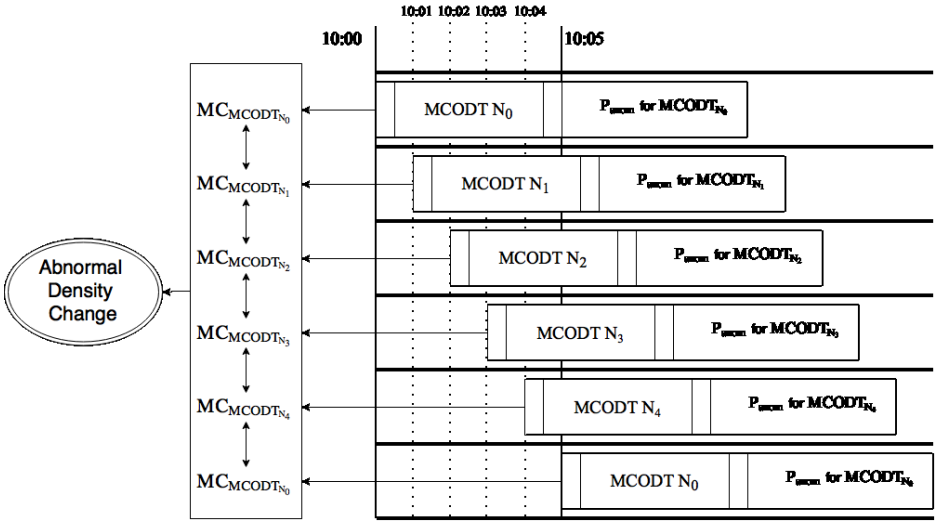

Fig. 2. Architecture Overview

4.1 Anomaly Detection

When using algorithms such as k-Nearest Neighbour and MCOD, a common variable that must be tuned to the dataset is k. While it's exact use may change, the principal remains consistent. k is used to describe the limit of normality within anomaly detection frameworks. It is a single variable used to tune the classification or detection rate within an algorithm. Within MCOD, it is the value at which a micro cluster is generated and an outlier is classified as an inlier. Within the context of NetFlow anomaly detection, this is a severe limitation, as it assumes that all network traffic activity has an equal distribution across the network, which is an extremely unlikely assumption (Discussed further in Section 5).

In order to successfully mitigate this limitation, all cluster densities are monitored irrespective of k's value. This allows for the successful detection of abnormal increases and decreases in cluster densities with respect to the time window being analysed.

4.2 Density Normality Measurement Generation

Within the proposed architecture, MCODT is initialized with a time window of size t. MCODT clusters data within this window, identifying anomalies using distance based calculations. The clusters generated within this time window contain samples that correspond to a type of network behaviour (Table 5). In parallel, windows are initialized in time steps of St using the same configuration as the initial algorithm.

Symbol	Meaning
q_i	the i-th query point.
S_q	stream of query points.
W_t	window of Size t-time.
p_i	i-th point p; $i = 1, ..., n$.
n_{W_t}	number of non-expired points in window W_t .
$p_i.arr$	arrival time of point p.
$p_i.exp$	expiring time of point p.
p_{anom}	point p declared as anomalous.
now	current time.
r	query range, $q.r$ is the distance parameter for query q.
k	number of neighbours parameter $q.k$ is the neighbours parameter for query q.
nn_{p_i}	number of neighbours for point p.
MC_i	i-th micro cluster, $i = 1, ..., n$.
D	micro cluster density, $MC_i.D$ is the density of the i-th microcluster.
$x.MC_i$	all time windows microcluster i is present in.
$y.MC_i$	all Density measurements for microcluster i

Fig. 3. Commonly Used Notation

Algorithm 1 Micro-Clustering Outlier Detection in Time-Series (MCODT)

Input: Query Point Stream S_q
Output: MC_i for all i, P_{anom}

```
while (S_q) do
    Obtain point q from S_q
    q.arr = now
    if q.r < |MC_i| then
        q is added MC
        MC.D++
    else if nn_q >= k then
        new MC
        MC + q
    else
        p = p_anom
```

Fig. 4. MCODT Definition

The positions of the clusters generated are then correlated with each subsequent MCODT instance to successfully capture the clusters activity through the other MCODT instances. This correlative step is needed as, due to the cluster generation step of MCODT, a cluster might not be in the exact same position throughout the sequential time windows, even though they represent the same class of network traffic.

Algorithm 2 Parallel Cluster Corrolation

Input: MC_i for all i within $MCODT_{N_0}$
Output: $D.MC_i$ for all i within $MCODT_N$ for all N
  for MC_i for all i within $MCODT_{N_0}$ **do**
    for MC_i for all i within $MCODT_N$ where $N \neq 0$ **do**
      if $MC_iN_0 = MC_iN$ **then**
        Clusters Represent the Same Class
        $D.MC_i = D.MC_N$ for all N

Fig. 5. Cluster Correlation

Once the clusters have been correlated and shown to be representative of the same class of network traffic, the cluster is persisted and given an ID. At the end of each instance of MCODT, the clusters density is measured and compared to its own historical activity and its activity in the other instances.

Algorithm 3 Residual Calculation For Cluster Density

Input: MC_i for all i within $MCODT_N$ for all N
Output: Residual of $D.MC_i$ for $MCODT_N$ for all n
  for $MCODT_N$ for all N **do**
    for MC_i for all i in $MCODT_N$ **do**

$$\bar{X}_n = n^{-1}[X_n + (n-1)\bar{X}_{n-1}]$$

$$s_n^2 = \frac{n-2}{n-1}s_{n-1}^2 + \frac{1}{n}(X_n - \bar{X}_{n-1})^2$$

Residual = $D.MC_i - 3(s_n) + \bar{X}_n$
if Residual> 0 **then**
  Flag as Anomaly

Fig. 6. Residual Calculation

This allows for the identification of anomalies, using the 3-standard deviation rule, within the persisted cluster. Furthermore, when a cluster is not generated in all the instances of MCOD, it is indicative of non-homogenous network activity. This specific type of traffic (as shown in Section 6) is highly irregular, and corresponds to network traffic that is extremely uncommon within the testing environment.

5 Methodology

5.1 Collected Data

To successfully test the proposed method, NetFlow was collected in a 24-hour period from within NPD. This was live data, and no previous insight about this 24hour period was held. It was unknown if it held anomalies or not, simulating realworld conditions. The NetFlow contained all communications, both internally and externally, during this period. A total of 151,995,634 NetFlow samples. From these samples, 8 attributes were extracted (Table 1). And from these, 6 attributes were selected to be used in the anomaly detection calculations.

- Source/Destination IP
- Destination Port
- Source Port
- Destination Bytes
- Source Bytes
- Protocol ID

These attributes were normalized using theoretical maximums as well as observed maximums over a 3-month period (Table 2).

Table 1. Selected NetFlow Attributes

Destination Bytes	Volume of traffic sent from the Destination IP in bytes
Destination I.P.	The destination I.P address of the NetFlow
Destination Port	The Destination port used for the NetFlow
Protocol I.D.	The ID of the Protocol used. (6 = TCP, 17 = UDP)
Source Bytes	Volume of traffic sent from the Source IP in bytes
Source I.P.	The Source IP of the NetFlow
Source Port	The Source port of the NetFlow
Start Time	The time at which the represented communiqué was initiated.

Table 2. Maximum Values used for Normalization

Variable	Maximum Value
Source/Destination Port	65535
Source/Destination Bytes	4294967270 (Bytes)
Protocol ID	255

The remaining collected attributes were excluded from the anomaly calculations due to various reasons. The IP addresses were excluded due to the IP address leases allocated by DHCP servers were inconsistent in both maintaining the allocated IP's, and time-out periods for leased IP's. This would lead to inconsistent results within networks, as IP addresses could be re-allocated in as little as 30 minutes, drastically changing their perceived normal traffic pattern. Instead, IP addresses were categorized as either internal or external, in order to develop separate clusters in feature space to represent internal-to-internal and external-to-external traffic types. These attributes, along with the Start Time attribute, were collected for the investigation of identified anomalies and clusters within the proposed architecture.

5.2 Program Configurations

Table 3. Program Configurations

Name	Symbol	Value
Minimum Density Required	K	50
Maximum range for Sample	R	.0025
Number of Algorithm Instances	N	5
Size of Window	T	360 seconds

For testing, a 24-hour example was chosen with no specific preference. No previous assumptions existed about this data before testing. The architecture was configured with an initial time window size of 5 minutes, and parallel instances were configured to run at one minute intervals after this, leading to a total of 5 MCOD instances processing the data in parallel. Configured variables are outlined in Table 3.

6 Results

In this section, we discuss the results of the proposed method of anomaly detection. Anomalous samples that were identified at both stages of the proposed architecture are outlined and analysed. Examples of normal activity of various types will also be presented.

6.1 Point Anomalies

Due to the two-stage architecture of the proposed method, anomalies may be detected in two different manners. Firstly, anomalies that are anomalous by the distanced based calculations are outlined at the end of every time window. An anomaly outlined in this window contains a point that never meets the required density irrespective of time. These are regarded as Point Anomalies. Due to the relatively small time window of the MCODT instances, the number of point anomalies detected within the first hour of processing was vast. The number of additional anomalies fell rapidly over the course of the analysis.

Because of this, focus was placed on point anomalies that were detected after the initial 12 hours of analysis. Table 4 outlines two such samples that were correctly identified as an anomaly.

Table 4. Sample Point Anomalies

Sample ID	Source Port	Destination Port	Source Bytes	Destination Bytes	Protocol ID
A	57838	53	116	262	17(UDP)
B	49886	1900	8428	0	17

Sample A represents a simple DNS request, which at first seemed like a false positive. However, upon investigation, it was shown that this DNS request was from an internal asset to an external DNS server. This incident was of interest to security technicians within the NPD Group. Sample B was an unauthorized UPnP (Universal Plug and Play) device connected to the network. It has been well documented how network security can be effected by having a UPnP device hosted on a network [25]. Upon Detection, the device was disconnected from the network.

6.2 Cluster Density Analysis

Table 5. Detected Anomalies

ID	Source Port	Destination Port	Source Bytes	Dest. Bytes	Protocol ID	N0	N1	N2	N3	N4
1	34701	6001	272	296	6	2	4	2	6	3
2	2598	6773	3486	3486	6	3	5	7	3	5
3	6188	41781	212	74	6	48	30	32	24	27
4	**54787**	**5355**	**376**	**0**	**17**	**5**	**NA**	**NA**	**NA**	**NA**
5	58544	2181	2840	1640	6	10	10	10	10	10
6	7858	443	1026	782	6	308	256	337	201	339
7	1720	61444	320	0	6	1418	1418	1418	1418	1418
8	45549	8879	22472	7580	6	755	755	755	755	755
9	**5925**	**5040**	**442**	**0**	**17**	**28**	**21**	**116**	**182**	**167**
10	1521	46172	3532	5624	6	10	15	9	13	14

Cluster densities were measured at regular intervals. Five instances of MCODT were run in parallel, each with the same configuration settings (Table 3) The initial instance, $MCODT_{N0}$, was initialized at 00:00am on the day in question. One minute after this, $MCODT_{N1,}$ was initialized, followed by $MCODT_{N2}$ and so on. This low level of analysis allows for the detection of possible malicious activity in as little as one minute after an incident Table 5 outlines sample clusters, selected based on being classified as an anomaly, and their activity over the course of one hour within the scope of all the

independent instances. This totals 1440 total densities measured for 1532 clusters generated and persisted over the course of testing.

Anomalies were detected in all instances of MCODT. However, anomalies were shown in both the independent analysis and correlative analysis. Of interest is an anomaly detected (Table 5. Sample 9). This anomaly appears in all instances of the algorithm, and is shown to have significant divergence from observed normal behaviours. The rapid increase, once investigated, was attributed to a single asset. It was shown to be connected to an external IP. It proceeded to attempt to open a connection to the external asset, but never received any connection. This was of interest to the security team within NPD, and was swiftly resolved. Sample 4 was also anomalous within the test. The Cluster only appeared in one window, showing the extremely temporal nature of the event. The other instances did not detect sufficient activity to generate a cluster. It represents an extremely short burst of activity to an external device.

7 Conclusions and Future Work

This paper proposed an architecture designed to detect anomalies within NetFlow data. To achieve this at a micro level, a clustering algorithm was run in parallel to determine anomalies in cluster activity in time-series. It was shown to be able to detect anomalies in live data without any previous knowledge on the data. These anomalies were investigated and shown to be of security interest. The result was interesting given that the testing was conducted on real world, live data. However, to compile a complete effectiveness rating for the algorithm would require a considerable amount of resources if used on real world representative data.

Future work would include refining the extensibility of the algorithm. Due to the abstraction of the density monitoring, it is possible to add attributes to MCODT's feature space in order to monitor changes in not only network traffic, but other metrics that could attribute to the risk of a malicious attack, such as the vulnerability of an asset determined by an external program.

References

1. Chen Z, Yeo CK, Francis BSL, Lau CT (2016) Combining MIC feature selection and feature-based MSPCA for network traffic anomaly detection. In: 2016 Third Int. Conf. Digit. Inf. Process. Data Min. Wirel. Commun. DIPDMWC. pp 176–181
2. Lin W-C, Ke S-W, Tsai C-F (2015) CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. Knowl-Based Syst 78:13–21. doi: 10.1016/j.knosys.2015.01.009
3. Fernandes Jr. G, Carvalho LF, Rodrigues JJPC, Proença Jr. ML (2016) Network anomaly detection using IP flows with Principal Component Analysis and Ant Colony Optimization. J Netw Comput Appl 64:1–11. doi: 10.1016/j.jnca.2015.11.024
4. Ciplinskas R, Paulauskas N (2016) Outlier Detection Method Use for the Network Flow Anomaly Detection. Moksl - Liet Ateitis 8:327–333. doi: 10.3846/mla.2016.928
5. Wankhede R, Chole V (2016) Intrusion Detection System using Classification Technique. Int J Comput Appl 139:25–28. doi: 10.5120/ijca2016909397
6. Goldstein M (2012) FastLOF: An Expectation-Maximization based Local Outlier detection algorithm. In: 2012 21st Int. Conf. Pattern Recognit. ICPR. pp 2282–2285

7. Kontaki M, Gounaris A, Papadopoulos AN, et al (2011) Continuous monitoring of distance-based outliers over data streams. In: 2011 IEEE 27th Int. Conf. Data Eng. pp 135–146
8. Purwanto Y, Kuspriyanto, Hendrawan, Rahardjo B (2015) Time based anomaly detection using residual polynomial fitting on aggregate traffic statistic. In: 2015 1st Int. Conf. Wirel. Telemat. ICWT. pp 1–5
9. Putra IWOK, Purwanto Y, Suratman FY (2015) Modified K-means algorithm using timestamp initialization in sliding window to detect anomaly traffic. In: 2015 Int. Conf. Control Electron. Renew. Energy Commun. ICCEREC. pp 19–23
10. Goeschel K (2016) Reducing false positives in intrusion detection systems using datamining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis. In: SoutheastCon 2016. pp 1–6
11. Uddin M, Rehman AA, Uddin N, et al (2012) Signature-based Multi-Layer Distributed Intrusion Detection System using Mobile Agents.
12. Vijayakumar V, Neelanarayanan V, Balan EV, et al (2015) Big Data, Cloud and Computing ChallengesFuzzy Based Intrusion Detection Systems in MANET. Procedia Comput Sci 50:109–114. doi: 10.1016/j.procs.2015.04.071
13. Singh K, Guntuku SC, Thakur A, Hota C (2014) Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests. Inf Sci 278:488–497. doi: 10.1016/j.ins.2014.03.066
14. Carela-Español V, Barlet-Ros P, Cabellos-Aparicio A, Solé-Pareta J (2011) Analysis of the impact of sampling on NetFlow traffic classification. Comput Netw 55:1083–1099. doi: 10.1016/j.comnet.2010.11.002
15. Kumari R, Sheetanshu, Singh MK, et al (2016) Anomaly detection in network traffic using K-mean clustering. In: 2016 3rd Int. Conf. Recent Adv. Inf. Technol. RAIT. pp 387–393
16. Alsayat A, El-Sayed H (2016) Social media analysis using optimized K-Means clustering. In: 2016 IEEE 14th Int. Conf. Softw. Eng. Res. Manag. Appl. SERA. pp 61– 66
17. T.Velmurugan D (2012) Efficiency of k-Means and K-Medoids Algorithms for Clustering Arbitrary Data Points. ResearchGate 3:1758–1764.
18. Asmuss J, Lauks G (2015) Network traffic classification for anomaly detection fuzzy clustering based approach. In: 2015 12th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD. pp 313–318
19. Abid A, Kachouri A, Mahfoudhi A (2016) Anomaly detection through outlier and neighborhood data in Wireless Sensor Networks. In: 2016 2nd Int. Conf. Adv. Technol. Signal Image Process. ATSIP. pp 26–30
20. Fu P, Hu X (2016) Biased-sampling of density-based local outlier detection algorithm. In: 2016 12th Int. Conf. Nat. Comput. Fuzzy Syst. Knowl. Discov. ICNC-FSKD. pp 1246–1253
21. Tsiatsikas Z, Fakis A, Papamartzivanos D, et al (2015) Battling against DDoS in SIP: Is Machine Learning-based detection an effective weapon? In: 2015 12th Int. Jt. Conf. EBus. Telecommun. ICETE. pp 301–308
22. Gajic B, Nováczki S, Mwanje S (2015) An improved anomaly detection in mobile networks by using incremental time-aware clustering. In: 2015 IFIPIEEE Int. Symp. Integr. Netw. Manag. IM. pp 1286–1291
23. Wong PC, Haglin D, Gillen D, et al (2015) A visual analytics paradigm enabling trillion-edge graph exploration. In: 2015 IEEE 5th Symp. Large Data Anal. Vis. LDAV. pp 57–64
24. Li B, Springer J, Bebis G, Hadi Gunes M (2013) A survey of network flow applications. J Netw Comput Appl 36:567–581. doi: 10.1016/j.jnca.2012.12.020
25. Zheng H, Li C, Chen Z (2011) Petri Nets Based Modeling and Analysis of UPnP Security Ceremonies. In: 2011 Third Pac.-Asia Conf. Circuits Commun. Syst. PACCS.