This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

Software-Defined-Networking-Enabled Traffic Anomaly Detection and Mitigation

Daojing He, *Member, IEEE*, Sammy Chan, *Member, IEEE*, Xiejun Ni, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—Traffic anomaly detection has been a principal direction in the network security field, which aims to identify attacks based on significant deviations from the established normal usage profiles. Recently, a new networking paradigm, software defined networking (SDN), has emerged to facilitate effective network control and management. In this paper, we present the advantages of leveraging SDN to detect traffic anomaly, and review recent progresses in this direction. Despite their effectiveness for traditional traffic, SDN-based traffic anomaly detection methods have to face the challenge of continuously increasing network traffic. To this end, we propose two refined algorithms to be used in an anomaly detection framework which can handle voluminous data, and report some experimental results to demonstrate their performance.**

I. Introduction

With the rapid advancements in information technologies, network-based services carrying sensitive information have grown tremendously. At the same time, end systems and networks are subject to more cyber attacks. For example, companies, institutions and governments are increasingly targeted by distributed denial-of-service (DDoS) attacks.

One possible approach to detect network attacks is to deploy an intrusion detection system to automatically monitor network activities. Broadly speaking, intrusion detection can be based on either misuse detection or anomaly detection approaches. Intrusions are detected based on the characteristic patterns of already known attacks in the misuse detection approach. As for the anomaly detection approach, it looks for patterns that significantly deviate from the established normal usage profiles. Its advantage is that it is applicable for both known and unknown attacks. Thus, significant research and development effort have been made on this approach, and a vast body of work is available in the literature. With the emergence of Internet of Things, many different devices, such as sensors or smart phones, communicate with each other through networks. This results in vast amounts of traffic being distributed over wide operation areas. It becomes challenging to detect anomalies among the wide-ranging voluminous data carried in the network.

D. He and X. Ni are with the School of Computer Science and Software Engineering, East China Normal University, Shanghai, P.R.China (e-mail: djhe@sei.ecnu.edu.cn).

S. Chan is with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong.(e-mail: eeschan@cityu.edu.hk).

M. Guizani is with the Department of Electrical and Computer Engineering, University of Idaho, USA. (e-mail: mguizani@ieee.org)

Daojing He is the corresponding author.

Recently, the innovative software defined networking (SDN) paradigm has emerged [1]. Its main idea is to separate the data plane and the control plane. This makes the switches in the network only to perform simple forwarding function, and the entire network to be controlled by a logically centralized controller through a standardized open interface. With this architecture, networks are made programmable at the network level through an open programming interface. This facilitates the development of more effective network configuration and management methods [2]. Leveraging on the capabilities of SDN, effective network anomaly detection approaches have been developed [3]–[7].

Motivated by these observations, this paper makes three main contributions:

1) We describe the challenges for designing an anomaly-based network intrusion detection system (ANIDS).
2) We then review recent progresses in SDN-enabled anomaly detection techniques. Moreover, we develop two refined algorithms to be used in an anomaly detection framework which can handle large-scale, high dimensional and unlabeled network data. The first one is the density peak based clustering algorithm with sampling adaptation. Besides the capability of extracting cluster centers and outlier points automatically, this algorithm has better time and memory efficiency than the original clustering method in the center selection stage. The second one is an unsupervised cluster-based feature selection mechanism. We use two different ways to compute the relations for discrete and continuous attributes respectively. Different from other feature selection mechanisms, we cluster the relevant features into groups according to their maximum redundancy from each other. Eventually redundant features are removed to make the feature number as least as possible.
3) We also implement the proposed algorithms, and evaluate their performance by comprehensive experiments. We demonstrate that our algorithms outperform other commonly used algorithms.

The rest of the paper is organized as follows. The next section provides an overview of traffic anomaly detection. In Section III, the challenges for designing an ANIDS are described. In Section IV, we summarize the capabilities of SDN for improving network security, and review recent progresses in SDN-enabled anomaly detection techniques. In Section V, we refine an unsupervised cluster-based feature selection algorithm and a density peak based clustering algorithm. In Section VI, the algorithms are evaluated by extensive

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

experiments. Finally, conclusions and some future research directions are provided.

II. Anomaly-based Network Intrusion Detection System

The generic architecture of an ANIDS is depicted in Fig. 1. It has the following components.

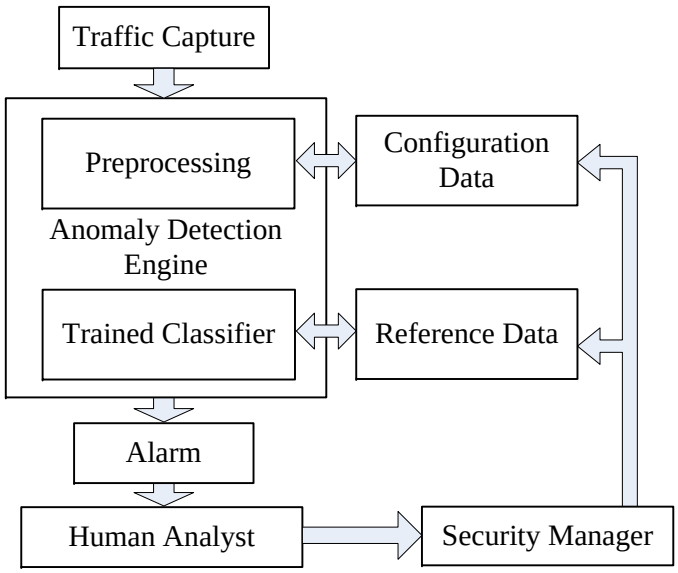


Fig. 1. The generic architecture of an ANIDS.

1) Traffic capture: Capturing traffic is the first step in network intrusion detection. As intrusions can be due to either outside or inside attackers, both incoming traffic and local traffic need to be monitored. Traffic can be monitored and captured at different granularity levels, which not only affects the amount of information available for analysis, but also the accuracy of detection algorithms.

2) Anomaly detection engine: This is the core component of an ANIDS. Its aim is to detect the existence of intrusions. Generally, network traffic is preprocessed before being passed to the detection engine. This can filter known attacks based on the misuse detection approach. Subsequently, unknown attacks are detected using the anomaly-based approach. In general, anomaly detection algorithms could be based on learning approaches, which can be further classified into supervised, semi-supervised and unsupervised learning. In supervised learning approaches, they first build a reference profile from the training data with labels obtained in a normally operated network. Since the reference profile only corresponds to the past network environment or services, any change in those aspects of the monitored network could lead to false alarm. Moreover, as training data are usually obtained from operational networks, they might be contaminated by hidden intrusions already. Such training data would affect the detection capability. To avoid these limitations of supervised approaches, unsupervised learning approaches have become a focus recently. The major advantage of unsupervised approaches is that they do not need labeled or attack-free training data.

3) Reference data: They contain signature information of known intrusions or profiles corresponding to normal behavior. Once new knowledge of the observed behavior becomes available, the profiles are updated by the processing elements.

4) Configuration data: They represent intermediate results, for example, partially created intrusion signatures. Combining existing knowledge and these intermediate results, consistent and up-to-date results can be generated.

5) Alarm: Having received results from the anomaly detection engine, this component generates alarm signals.

6) Human analyst: This component is responsible for analyzing and interpreting the alarm signals, and then taking appropriate actions.

7) Security manager: When new intrusions are found, the security manager updates the stored intrusion signatures.

Readers are referred to [8] for a more detailed description of an ANIDS.

III. Anomaly Detection Challenges and Related Work

In this information age, data come from a wide variety of sources such as images, videos, web searches, health records, and interactions in social networking applications. The massive amount of digital information are generated by today's wide variety of information sources.

As a result, designing ANIDSs needs to take into account the "4V's" characteristics:

1. **Volume**. The scale and complexity of network traffic has been increasing faster than that predicted by Moore's law. Thus, traditional time-consuming traffic anomaly detection methods can no longer satisfy the requirement of timeliness.
2. **Variety**. Network traffic are generated from various sources. Novel attacks can elude signature-based detection systems and cause great damage.
3. **Value**. With the low value density, some features of the traffic data do not play a role in identifying anomaly, or even hinder the process.
4. **Velocity**. Detection speed is very important. Ultimately, anomalies should be detected in real time.

According to [9], anomaly detection can be based on statistical, machine learning, or data mining techniques. Statistics-based methods tend to have a high false positive rate because they cannot handle the non-stationary variation of network traffic [10]. This shortcoming is alleviated by data mining techniques [11], which aims to discover understandable patterns or models from given data sets. Some earlier work show that data mining techniques can efficiently identify profiles of normal network activities and build classifiers to detect abnormal activities.

Supervised approaches [11] highly rely on training data from normal activities. Clearly, training data are collected from past activities. Thus, the established normal profile is only relevant to historical activities, and new normal activities from different network environments or services may be treated as attacks. When anomaly detection systems (ADSs) are trained by data with hidden intrusions, they usually do not have the capability to detect intrusions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

To overcome the limitations of supervised approaches, unsupervised approaches have recently become a focus [12], [13]. Attack-free training data are not needed for unsupervised anomaly detection methods. In distance-based methods, data are grouped as clusters characterized by their centers. However, since a data point is naturally assigned to the nearest center, these methods are not suitable for non-spherical clusters. In density-based spatial clustering methods, the points in those regions with densities less than a pre-defined threshold are discarded as noise. Moreover, disconnected regions of high density are assigned to different clusters. However, choosing a suitable threshold is not trivial.

Due to increasing dimensionality and size of datasets, many existing algorithms used in ADSs become less effective and efficient. This issue can be addressed by applying feature selection algorithms as a pre-processing step in ADSs to remove irrelevant and redundant features. Usually, the mutual information between feature and labels is used to select features. However, when continuous variables are contained in datasets, it becomes a challenge to measure the relations between features as the result significantly depends on the discretization methods.

In conclusion, the conventional learning methods used in anomaly detection are not suitable to handle the coupling relations due to the increased complexity, heterogeneity and multiple dimensionality of the data. They are not able to find the inter- and intra- relationship efficiently. They neither perform well nor scale well under highly unstructured and unpredictable conditions in terms of data volume and variety.

IV. Leveraging SDN for anomaly Detection

A. The Capabilities of SDN for Improving Network Security

The following capabilities of SDN can be exploited to enhance network security functions.

- **Centralized control of the network:** Network managers are provided a global view of the network. Thus, with the support of OpenFlow (OF) [4], intrusions can be easier to spot.
- **Software-based traffic analysis:** With OpenFlow enabled forwarding devices, data can be extracted from network traffic at a desired granularity level. The data analysis can be carried out in a separate unit, which has high processing power, by a wide range of available tools such as expert systems, statistical analysis and machine learning.
- **Flow abstraction:** All traffic are abstracted as flows. In the flow table of forwarding devices, an entry is maintained for each flow. The same control element can manage different flow types by applying different rules to entries.

With the global view and control capabilities enabled by SDN, network security can be improved by implementing well-known intrusion detection systems on top of the controller as applications. Integrating security functions into SDN in this way is less complex. Also, with the global view of network events, events could be better classified and false alarm rate could be reduced. In addition, SDN has the advantage of allowing easy share of information between different security applications [14].

B. SDN-enabled Traffic Anomaly Detection Methods

The first step in anomaly detection is to collect statistics of traffic flows. This is inherently supported in SDN as the controller can request flow statistics from the forwarding devices. However, it is important to choose a suitable granularity level of flow statistics collection. If the granularity level is too high, too much resources, such as CPU power, memory and network bandwidth, would be consumed. On the other hand, if the granularity level is too low, insufficient information are collected, based on which the anomaly detector may fail to identify certain anomalies.

In [3], the author propose an adaptive and interactive approach to collect flow statistics in SDN. Essentially, the detector can dynamically specify the granularity levels for different flows according to some preliminary observations. For example, the detector can request a finer granularity for flows exhibiting suspicious behaviour, or a coarser granularity for flows behaving normally. Thus, a higher detection accuracy is achieved while less resources are consumed. Such an adaptive approach exploits the important characteristics of SDN that the central controller can dynamically control the functions of the forwarding devices.

An early work of anomaly detection in SDN is a framework for flow-based DDoS detection proposed in [4]. The framework consists of three modules; *flow collector, feature extractor* and *classifier*. Whenever a packet arrives at a forwarding device, the statistics of its corresponding flow entry maintained by the forwarding device is updated. Then, the flow controller requests flow statistics from forwarding devices periodically via the Openflow protocol. Based on these statistics, features relating to DDoS attacks are extracted by the feature extractor. The extracted features are then passed to the classifier to determine whether they correspond to a DDoS attack. When a problematic flow is identified, appropriate actions will be triggered. The classifier can be implemented by any learning or statistical method.

In traditional networks, Remote Triggered Black-Hole (RTBH) [5] is a commonly used approach to mitigate DDoS attacks. It is executed in the edge routers to prevent malicious traffic from entering the network. When a victim of DDoS attack is identified by any available method, all traffic, including legitimate and attack traffic, destined to the victim will be forwarded to a discard interface, or the so-called *black hole*. This approach avoids using the filtering function based on access control list because it is not as efficient as forwarding-rules processing. However, the drawback of this approach is that even normal traffic are prevented from reaching the victim. In [6], an SDN-based framework is proposed to improve the performance of RTBH in legacy networks. The framework has three modules. The first module is composed of two independent components, namely, statistics collection and anomaly detection. The statistics collection component collects traffic statistics from edge routers using an appropriate method, e.g, sFlow. It then feeds the statistics to the anomaly detection

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

component, which is to detect potential attacks and then identify the victims. This is a generic component, in which the network manager can choose any anomaly detection algorithm that deemed to be suitable. Once a victim is identified, the first module instructs the RTBH module to propagate a static route to the entire network to redirect the victim's traffic to a OF-enabled switch. From there, the anomaly mitigation module inspects the traffic on a per-flow basis in order to identify the malicious flows. Then packets belonging to malicious flows are discarded, while normal packets are forwarded to the destination. The advantages of this framework are two-fold. First, defense actions are confined to malicious flows only. Thus, the victim can still maintain normal communications even when under attack. Second, the RTBH mechanism is triggered dynamically.

Very often, adversaries launch DDoS attacks by compromising computers with malware and then employing them as zombies to send large amount of fake traffic to the attack target. Usually, computers in SOHO (Small Office/Home Office) and home networks are most vulnerable because their users are not well aware of security issues. On the other hand, ISPs can hardly exert any security control in these networks as they are outside their premises. To tackle this problem, in [7], the authors propose to delegate the anomaly detection function to home networks by making use of SDN, without involving the home users to carry out complicated security management tasks. This requires the home router to be openflow compatible. They demonstrate the efficacy of their proposal by implementing four prominent anomaly detection algorithms in a NOX platform and testing their performances. Their experiments confirm that these algorithms, when operated in home networks, can identify anomalies more accurately than when they are operated in ISP networks. Moreover, they can operate at line rates without degrading the network performance experienced by the normal traffic in the home network.

V. Refined Algorithms

Although SDN can help to improve the scalability and flexibility of traffic anomaly detection, these methods still incur significant network traffic and analysis delay. To address the challenges described in Section III, we need to refine the anomaly detection methods.

A common two-stage unsupervised learning approach comprises feature selection and clustering. Feature selection aims to remove redundant features in the dataset. Then, clustering classifies the dataset with reduced dimension into normal and mis-behaved patterns. Here, we propose refined feature selection and clustering algorithms that can handle large-scale, high dimensional and unlabeled network data.

A. Feature Selection

Suppose F denotes the whole set of features, F_i denotes an element of F, C denotes the target concept and S_i denotes the F-F_i. There are mainly three kinds of features:

Definition 1: Strong Correlation. F_i is strong relevant to C if and only if

$$p(C|S_i, F_i) \neq p(C|S_i). \tag{1}$$

Strong relevant features can have impact on distribution of classification. If there is a lack of relevant features, the result may not be accurate.

Definition 2: Weak Correlation. F_i is weak relevant to C if and only if

$$p(C|S_i, F_i) = p(C|S_i),\ \exists S'_i \subset S_i, p(C|S'_i, F_i) \neq p(C|S'_i). \tag{2}$$

A weak relevant feature only has impact on the distribution of classification under certain conditions.

Definition 3: Independence. F_i is an independent feature if and only if

$$\forall S'_i \subset S_i, p(C|S'_i, F_i) \neq p(C|S'_i). \tag{3}$$

Independent features have no influence on the distribution of classification, therefore, they can be removed first.

To measure the correlation between two features, *symmetric uncertainty* is used in previous work [15], which is defined as

$$SU(F_i, F_j) = \frac{2 * Gain(F_i, F_j)}{H(F_i) + H(F_j)}, \tag{4}$$

where $H(F_i)$ is the entropy of a discrete random variable F_i. Let $p(f)$ be the prior probabilities for all values of F_i, $H(F_i)$ is defined by:

$$H(F_i) = -\sum_{f \in F_i} p(f) \log_2 p(f). \tag{5}$$

Let $H(F_i|F_j)$ be the conditional entropy of F_i with a priori knowledge of F_j. $Gain(F_i, F_j)$ is defined as:

$$Gain(F_i, F_j) = H(F_i) - H(F_i|F_j) = H(F_j) - H(F_j|F_i). \tag{6}$$

$Gain(F_i, F_j)$ is a measure of how much uncertainty of an unknown variable is reduced by a known variable.

In supervised learning methods, features with small $SU(F_i, C)$ are regarded as independent features and can be removed first. However, the distribution of C is not available in unsupervised learning methods. In that case, $SU(F_i, C)$ is replaced by another measurement called *relevancy*, ref, which is defined as:

$$ref(F_i, C) = \frac{1}{n} \sum_{j=1}^{n} SU(F_i, F_j), \tag{7}$$

$$ref(F_i, F_j) = SU(F_i, F_j). \tag{8}$$

The above formulae are directly applicable to discrete attributes such as protocol type, but not to continuous attributes since they have infinite number of possible values. In a simple way, infinite values can be discretized and mapped to finite values. However, simple discretization methods like equal-width and clustering can only roughly compute the relation.

Methods based on mutual information estimators are not suitable for identifying potentially interesting relationships in a dataset because they only have a strong preference for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

certain relations. In this paper, maximal information coefficient (MIC) [16] is used to calculate the relation information between two continuous features because it is able to identify potentially interesting relationships independent of their form.

Assume that a relationship exists between two variables, the intuitive idea of obtaining MIC is to partition the data by drawing a grid on the scatterplot of the two variables, and then to encapsulate the relationship from the partitioned data. Grids of variable sizes, up to a maximum value, are applied. Then, for a given two-variable dataset, the calculation of MIC is carried out as follows. First, for every pair of integers (x, y), the largest possible mutual information achievable by any possible grid is computed. Second, the mutual information values are normalized to be between 0 and 1 to facilitate a fair comparison between grids of different sizes. Third, a characteristic matrix $M = (m_{x,y})$ is formed, where $m_{x,y}$ is the highest normalized mutual information among all grids. Finally, MIC is the maximum value in M.

Having $ref(F_i, C)$ and $ref(F_i, F_j)$, the features are filtered as follows. First, features with $ref(F_i, C) < threshold1$ are removed as they do not make any significant contribution for identification. Second, features with $ref(F_i, C) > threshold2$ are regarded as redundant and removed. Both $threshold1$ and $threshold2$ are pre-selected.

Time complexity analysis. The main portion of runtime in this algorithm is the computation procedure of MIC values for each pair of features. So part 1 of the algorithm has a time complexity of $O(n^2)$ in terms of the number of continuous features n. In part 2, we construct feature clusters according to their redundancy. The time complexity of part 2 is $O(k^2)$ where k is the number of relevant features derived from the continuous features and k is not larger than n. In part 3, the feature with highest $ref(F_i, C)$ value will be chosen in every cluster constructed in part 2. Every feature in clusters is visited once. So part 3 has linear complexity $O(k)$. Overall, the time complexity of the algorithm is $O(n^2) + O(k^2) + O(k)$. Given $k <= n$, the overall complexity is $O(n^2)$.

B. Density Peak based Clustering

As discussed before, density-based spatial clustering methods are less restrictive than distance-based methods. However, it could be non-trivial to choose an appropriate threshold for the density-based spatial clustering methods.

In some clustering algorithms [12], [13], certain parameters such as number of clusters need to be pre-defined, but their selected values affect the detection accuracy. Recently, Rodriguez *et al.* develop an alternate clustering approach based on fast search and find of cluster centers [17]. It is based on the idea that cluster centers would have a higher density than their neighbouring points and a relatively large distance from points with higher densities. For a set of data samples, this approach calculates two quantities of each data sample; ρ_i and δ_i. ρ_i is a measure of local density of data point i. It can be computed in two different ways.

In cut-off kernel,

$$\rho_i = \sum_{j \in I_S \setminus \{i\}} \chi(d_{ij} - d_c) \tag{9}$$

$$\chi(x) = \begin{cases} 1, x < 0; \\ 0, x \geq 0, \end{cases} \tag{10}$$

In Gaussian kernel,

$$\rho_i = \sum_{j \in I_S \setminus \{i\}} e^{-\left(\frac{d_{ij}}{d_c}\right)^2} \tag{11}$$

δ_i is a measure of the minimum distance between point i and any other point with higher density. Those points with anomalously large δ_i are regarded as cluster centers.

In the process of computing the density and distance of data points, a matrix of size $N \times N$ is maintained, where N is the number of data points. When the dataset is very large, a lot of memory is consumed. On the other hand, we notice that if randomly sampled data are used, the location of cluster centers only changes slightly because, after unbiased sampling, those data points with high density still have higher density than other points. Therefore, we propose only a portion of network data is used to approximately locate cluster centers. Similar to other algorithms, once the cluster centers have been identified, points are assigned to the nearest cluster. The details of our refined clustering algorithm are given in Algorithm 1.

Time complexity analysis. The original density peak clustering algorithm uses the whole dataset in both center selection and data labeling. Assuming there are N samples in dataset D. The time complexity for calculating distance matrix $dist$ for each pair in D is $O(N^2)$ and the memory complexity to store the $N \times N$ matrix $dist$ is also $O(N^2)$. Assuming it takes 4 bytes to store a floating-point data. The upper boundary of N in a 64-bit computer with 4GB memory is 2^{15}, raising bottleneck to large-scale data. In the clustering stage, every data is assigned to a selected center. Assuming the center number is k, the time complexity of this part is $O(kN)$. The whole time complexity is $O(N^2)$, which is rather high when N is too large. However, in our proposed method, a downsampling operation is carried out in the first stage to obtain an approximate center information. Part 1 in the algorithm takes linear complexity $O(N)$ since there is only one loop. In Part 2 we use sampled data to select centers with high local density and distance. While the scale of sampling is crucial to make the tradeoff between computing resource and accuracy of selection, here we pick $C\sqrt{N}$ samples where C is a constant not greater than 5 and the time complexity of Part 2 is $O(N)$. Part 2 is the same as the original Density Peak and therefore the runtime of our proposed method is $O(N) + O(N) + O(Nk))$. Since k is the number of centers chosen from samples so it is less than sample number $O(\sqrt{N})$. Eventually, we obtain the time upper boundary of sampled density peak clustering, $O(N^{3/2})$, which is less than unsampled method.

In summary, our proposed feature selection algorithm is unsupervised and cluster-based, and can deal with both discrete and continuous features. Using MIC, potentially interesting relationships between two continuous attributes can be identified. In density peak clustering algorithm, we improve its time and memory efficiency by only operating on the sampled data points.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

Algorithm 1 Data clustering by sampled Density-Peak algorithm

Require: D=$\{F_0, F_1 ... F_n\}$ - the dimension reduced dataset without label
m - sample reduce factor $Percent$ - position of d_c
$\theta 1$ - threshold for density $\theta 2$ - threshold for distance
Ensure: label - labels of data
for $i = 0$ *to* N **do**
if $random.(0, m) == 0$ **then**
$Sample.insert(D[i])$
end if
end for
=====Part1:Choose samples for centers =====
$List\ LL$
for *each pair* $(Sample[i], Sample[j])$ *in Samples* **do**
$dist[i][j] = eculidean\_distance(Sample[i], Sample[j])$
$LL.append(dist[i][j])$
end for
$d_c = percent * sorted(LL)$
for *each* i *in Sample* **do**
$Rho[i] = count_{j \in Sample\ \cap\ dist[i][j] < dc}(j)$
$Delta[i] = min_{j \in Sample\ \cap\ Rho[j] > Rho[i]}(dist[i][j])$
end for
for *each* i *in Sample* **do**
if $Rho[i] > \theta 1 \cap Delta[i] > \theta 2$ **then**
$Center.insert(i)$
end if
end for
=====Part2:Cluster center selection=====
$Label = [N]$
for *each* i *in* D **do**
$Label[i] = min_{j \in Centers}(eculidean\_distance(D[i], Center[j]))$
end for
=====Part3:Labeling=====
return $Label$

VI. Experimental evaluation

A. Experimental Settings

KDDCup99 dataset is publicly available and used in our experiments. It contains five-million TCP connection records logged in a period of seven weeks. There are four main categories of attacks: information gathering attacks, DoS, user to root and remote to local. There are both labeled and unlabeled records in the dataset. Each labeled record contains of 41 features (attributes) as depicted in Table I and one target value. Target value indicates the attack category name.

There are continuous, discrete and symbolic attributes in the dataset, and they have different resolution and ranges. For discrete and symbolic attributes, their entropy and mutual information are computed without any preprocessing in the feature selection stage, and they are normalized and scaled in the clustering stage. For example, symbolic features like $protocol\_type$, $services$, $flags$ and $attack\_names$ are first mapped to integer values ranging from 0 to $N-1$ where N is the number of symbols. Then, min-max normalization process is implemented. Each of the feature is linearly scaled to the range of [0, 1.0].

The performance of our proposed algorithms are evaluated on a computer with an Intel I5 CPU and 4 GB main memory. The clock rate of the CPU is 3.2 GHz. The proposed algorithms are implemented using Python 2.7.9. Several valuable utilities, MINE package and Python open source machine learning libraries Scikit-learn, Numpy, SciPy, Matplotlib are also used.

B. Experiment for Feature Selection Stage

Comparison is carried out between our feature selection algorithm and two other well known ones: Recursive Feature Elimination (RFE) and ExtraTreeClassifier.

Five classification algorithms are employed to classify datasets before and after feature selection. They are the tree-based DecisionTreeClassifier, ensemble learning method ExtraTreesClassifier, Random Forest Classifier algorithm, AdaboostClassifier and optimal margin-based Support Vector Machine, respectively.

Each algorithm is run 100 times to obtain an average performance value.

After running our feature selection algorithm, 16 features are kept, i.e., a 60.97% reduction of original features. Fig. 2 shows the detection accuracy of the five classifiers applied on reduced datasets resulting from the three feature selection algorithms. For each classifier, the results of these feature selection algorithms appear in the same order, from left to right: the original data, our proposed feature selection algorithm, RFE and ExtraTreesClassifier. From the figure, we observe that:

1. Most classifiers achieve the highest accuracy when they operate on the original dataset. This is because all information contained in the dataset is preserved.
2. Our proposed algorithm and ExtraTreeClassifier can achieve an accuracy close to that obtained from original dataset. In most cases, ensemble learning model can achieve a higher detection accuracy than other models such as Decision Tree and Support Vector Machine.
3. Compared with other supervised approaches, our proposed algorithm achieves a detection accuracy which only has 0.4% difference with that of the ExtraTreesClassifier. Moreover, it achieves 3.3% better than RFE. The results show that even without labels, the detection accuracy of our algorithm is comparable with supervised approaches.

We have also recorded the runtime of each classifier when it is executed with and without feature selection. The results presented in Table II show that, with our proposed algorithm, the runtimes of the classification methods are significantly reduced.

C. Experiment for Clustering Stage

Four mainstream clustering algorithms, Birch, Kmeans, DBSCAN and MeanShift, are compared with our proposed algorithm (referred to as Sampled-DP). The algorithms are ranked under each of the following metrics: 1) Runtime,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

TABLE I
SUMMAY OF THE 41 ATTRIBUTES IN KDDCUP99 DATA SETS

No.	Feature Name	Type	No.	Feature Name	Type
1	duration	C	22	is_guest_login	D
2	protocol_type	D	23	count	C
3	service	D	24	src_count	C
4	flag	D	25	serror_rate	C
5	src_bytes	C	26	srv_serror_rate	C
6	dst_bytes	C	27	rerror_rate	C
7	land	D	28	srv_rerror_rate	C
8	wrong_fragment	C	29	same_srv_rate	C
9	urgent	C	30	diff_srv_rate	C
10	hot	C	31	srv_diff_host_rate	C
11	num_failed_logins	C	32	dst_host_count	C
12	logged_in	D	33	dst_host_srv_count	C
13	num_compromised	C	34	dst_host_same_srv_rate	C
14	root_shell	D	35	dst_host_diff_srv_rate	C
15	su_attempted	D	36	dst_host_same_src_port_rate	C
16	num_root	C	37	dst_host_srv_diff_host_rate	C
17	num_file_creations	C	38	dst_host_serror_rate	C
18	num_shells	C	39	dst_host_srv_serror_rate	C
19	num_access_files	C	40	dst_host_rerror_rate	C
20	num_outbound_cmds	C	41	dst_host_srv_rerror_rate	C
21	is_hot_login	D			

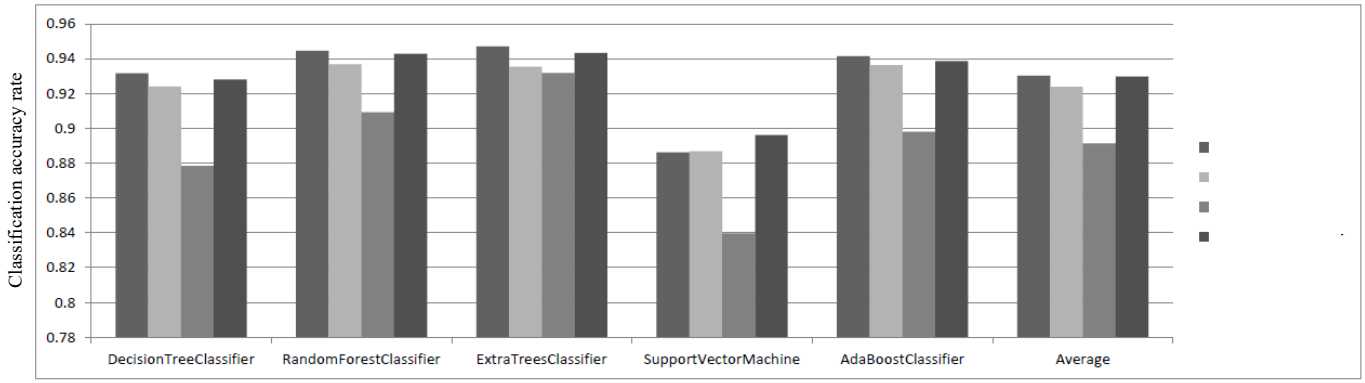


Fig. 2. Classification accuracy rate over different feature selection methods.

TABLE II
RUNTIME COMPARISON BETWEEN TWO DATASETS

	Orignal data	Reduced data	Time reduced
DecisionTreeClassifier	0.2520	0.1748	30.63%
SupportVectorMachine	6.6171	5.8828	11.09%
ExtraTreesClassifier	0.3782	0.3370	10.89%
RandomForestClassifier	0.3969	0.3537	10.88%
AdaBoostClassifier	22.4513	20.4912	8.73%
Average	6.0191	5.4479	14.44%

running time of the algorithm, counted by seconds. 2) Adjusted_mutual_information, the normalized mutual information between predicted labels and true labels. 3) Adjusted_rand_score, a similarity measure between two clusters by considering all pairs of samples. 4) Completeness_score, the degrees of all members of a given class are assigned to the same cluster. 5) Homogeneity_score, the degree of each cluster only containing members of a single class. 6) Mutual_info_score, the normalized mutual information between predicted labels and true labels. 7) Detection accuracy, the rate of samples clustered to the right cluster.

Comparison of the five clustering algorithms are carried out for original data and reduced data, respectively. Given our downsampling operation, the experiments are carried out on the same machine over 100 times to obtain the average values.

From Table III we obtain the ranks of different clustering algorithms under seven metrics. The numbers in brackets indicate the ranks of the algorithms. It is noted that our proposed algorithm ranks first in detection accuracy and second in runtime, respectively, which are the most important factors considered in practice. The value of the overall rank metric is computed as the average of the total seven metrics. The overall rank shows that our proposed algorithm outperforms other four clustering algorithms generally. Compared with Sampled-DP, Birch, KMeans, DBSCAN achieves better performance in adjusted_MI, adjusted_rand_score, which demonstrates that those methods perform better in clustering purity, however, they are not as fast and accurate as the proposed method. The overall rank shows that Sampled-DP outperforms other four clustering algorithms generally.

Table IV also marks the ranks of different clustering algorithms under seven metrics with feature selected dataset. This time KMeans clustering algorithm ranks 1st on overall ranks, showing its high suitability and performance over reduced

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal

TABLE III
CLUSTERING METHODS COMPARISON ON ORIGINAL DATASETS

Approaches	Runtime(s)	adjusted_MI	adjusted_rand	completeness	homogeneity	mutual_info	detection_accuracy	overall_rank
Birch	19.14(5)	0.742(2)	0.814(1)	0.743(2)	0.892(4)	1.996(4)	0.93488(2)	2.857(4)
Kmeans	2.96(3)	0.716(3)	0.795(3)	0.718(3)	0.896(2)	2.003(2)	0.93358(3)	2.714(3)
DBSCAN	4.09(4)	0.745(1)	0.808(2)	0.748(1)	0.894(3)	1.998(3)	0.91633(4)	2.571(2)
MeanShift	0.70(1)	0.321(5)	0.147(5)	0.591(5)	0.326(5)	0.729(5)	0.68175(5)	4.428(5)
Sampled-DP	1.68(2)	0.673(4)	0.795(3)	0.677(4)	0.904(1)	2.02(1)	0.94564(1)	2.286(1)

TABLE IV
CLUSTER METHOD COMPARISON ON REDUCED DATASETS

Approaches	Runtime(s)	adjusted_MI	adjusted_rand	completeness	homogeneity	mutual_info	detection_accuracy	overall_rank
Birch	4.63(5)	0.695(2)	0.78(1)	0.697(2)	0.86(3)	1.924(3)	0.89602(3)	2.714(3)
Kmeans	1.65(3)	0.695(2)	0.78(1)	0.697(2)	0.873(2)	1.952(2)	0.90848(2)	2.0(1)
DBSCAN	2.01(4)	0.777(1)	0.74(4)	0.791(1)	0.778(4)	1.74(4)	0.85966(4)	3.143(4)
MeanShift	0.41(1)	0.539(5)	0.378(5)	0.639(5)	0.542(5)	1.211(5)	0.81449(5)	4.428(5)
Sample-DP	1.59(2)	0.659(4)	0.773(3)	0.662(4)	0.899(1)	2.011(1)	0.92248(1)	2.286(2)

data. But considering crucial factors such as runtime and accuracy, Sampled-DP is slightly faster than KMeans and more precise. Besides, the detection accuracy of Sampled-DP over reduced data is comparable to other methods over original data.

According to Tables III and IV, Sampled-DP achieves the best performance in terms of runtime and detection accuracy against other four clustering algorithms, which are the two most significant factors in performance evaluation. Also, Sampled-DP ranks 1st and 2nd on overall ranks before and after feature selection, respectively.

In Fig. 3, we obtain 4 different decision graphs of sampled data by setting different sample numbers. Observations show that with the growth of sample number, the maximum value of X-axis ρ is becoming larger, which indicates the growth of overall local density. Meanwhile, detection accuracy grows with the scale of sample points. With fixed thresholds for δ and ρ, it is likely to select more centers in a more densely distributed decision graph.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have reviewed the capabilities of SDN that can be leveraged to develop efficient traffic anomaly detection methods, and discussed recent progress in this direction. However, new challenges are imposed on the design of traffic anomaly detection methods. To address these challenges, we have proposed refined unsupervised feature selection and density peak clustering algorithms, for anomaly detection of large-scale and high-dimensional network data without labels. When a hierarchy of controllers is used, our proposed approach can be used to analyze traffic data locally in each controller. The intermediate data can then be merged to generate the final result. This reduces the amount of traffic shuffled across the network.

Our future research direction will focus on real-time packet clustering so as to defend against intrusions timely. Also, we can consider how to utilize some big data ecosystem frameworks in the design of traffic anomaly detection in order to improve its performance.

VIII. ACKNOWLEDGEMENT

This research is supported by the National Science Foundation of China (Grants: 51477056 and U1636216), the Shanghai Rising-Star Program (No. 15QA1401700), the CCF-Venustech Hongyan Research Initiative, the State Grid Corporation Science and Technology Project The pilot application on network access security for patrol data captured by unmanned planes and robots and intelligent recognition based on big data platform (Grant No. SGSDDK000KJJS1600065), the Pearl River Nova Program of Guangzhou (No. 2014J2200051) and a strategic research grant from City University of Hong Kong (No. 7004615).

REFERENCES

[1] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communication Surveys & Tutorials*, vol. 17, no. 1, pp. 27-51, First Quarter 2015.

[2] S. Khan, A. Gani, A. Wahab, A. Abdelaziz, K. Ko, M. Khan, and M. Guizani, "Software-defined network forensics: Motivation, potential locations, requirements, and challenges," *IEEE Network*, vol. 30, no. 6, pp. 6-13, Nov.-Dec. 2016.

[3] Y. Zhang, "An adaptive flow counting method for anomaly detection in SDN," in *Proc. ACM CoNEXT*, pp. 25-30, 2013.

[4] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. IEEE LCN*, pp. 408-415, Oct. 2010.

[5] W. Kumari and D. McPherson, "Remote triggered black hole filtering with unicast reverse path forwarding," *RFC5635*, Internet Engineering Task Force, Aug. 2009.

[6] K. Giotis, G. Androulidakis, and V. Maglaris, "Leveraging SDN for efficient anomaly detection and mitigation on legacy networks," in *Proc. EWSDN*, pp. 85-90, Sept. 2014.

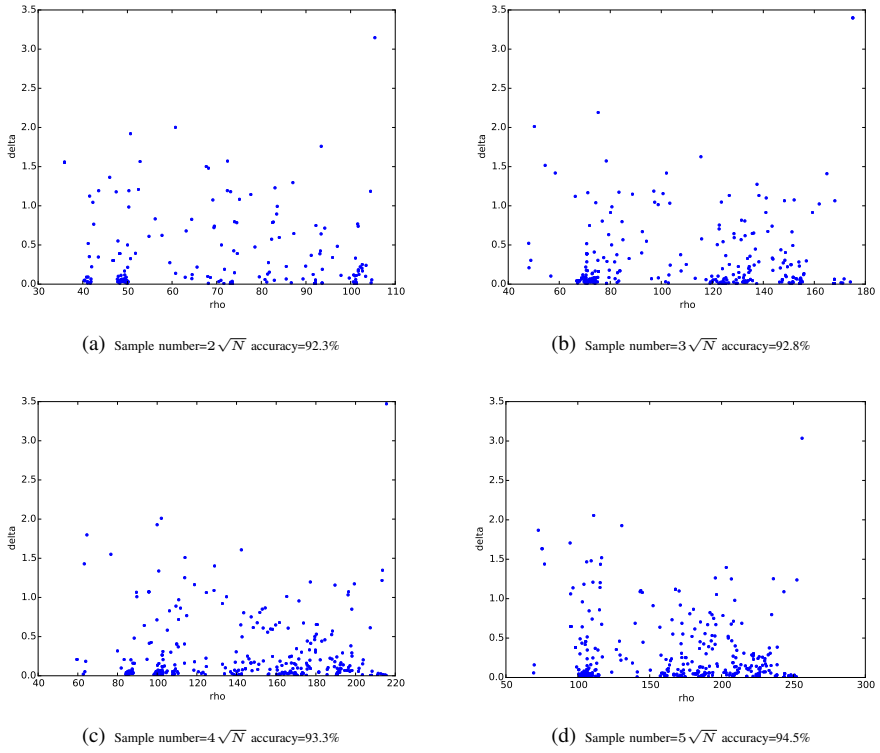
[7] S. Mehdi, J. Khalid, and S. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proc. RAID*, pp. 161-180, 2011.

[8] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Comm. Surveys Tutorials*, vol. 16, no. 1, pp. 303-336, First Quarter 2014.

[9] D. Jiang, Z. Yuan, P. Zhang, L. Miao, and T. Zhu, "A traffic anomaly detection approach in communication networks for applications of multimedia medical devices," *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14281-14305, Nov. 2016.

[10] Y. Luo, B. Wang, Y. Sun, B. Zhang, and X. Chen, "FL-LPVG: An approach for anomaly detection based on flow-level limited penetrable visibility graph," in *Proc. ICINS*, pp. 1-7, Nov. 2013.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2694702, IEEE Internet of Things Journal



(a) Sample number=$2\sqrt{N}$ accuracy=92.3%

(b) Sample number=$3\sqrt{N}$ accuracy=92.8%

(c) Sample number=$4\sqrt{N}$ accuracy=93.3%

(d) Sample number=$5\sqrt{N}$ accuracy=94.5%

Fig. 3. Decision graph of different sample numbers

[11] X. Tong, Z. Wang, and H. Yu, "A research using hybrid RBF/Elman neural networks for intrusion detection system secure model," *Computer physics communications*, vol. 180, no. 10, pp. 1795-1801, Oct. 2009.

[12] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proc. the Twenty-eighth Australasian conference on Computer Science (ACSC)*, pp. 333-342, 2005.

[13] H. Egilmez and A. Ortega, "Spectral anomaly detection using graph-based filtering for wireless sensor networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1085-1089, 2014.

[14] A. AlErouda and I. Alsmadib, "Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach," *Journal of Network and Computer Applications*, vol. 80, pp. 152-164, Feb. 2017.

[15] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 1-14, 2013.

[16] D.N. Reshef, *et al*, "Detecting novel associations in large data sets," *Science*, vol. 334, no. 6062, pp. 1518-1524, Dec. 2011.

[17] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

Daojing He (S'07-M'13) received the B.Eng. (2007) and M. Eng. (2009) degrees from Harbin Institute of Technology (China) and the Ph.D. degree (2012) from Zhejiang University (China), all in Computer Science. He is currently a Professor in the School of Computer Science and Software Engineering, East China Normal University, P.R. China. His research interests include network and systems security. He is an associate editor or on the editorial board of some international journals such as *IEEE Communications Magazine* and *IEEE/KICS Journal of Communications and Networks*. Contact him at djhe@sei.ecnu.edu.cn.

Sammy Chan (S'87-M'89) received his B.E. and M.Eng.Sc. degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and a Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Australia, in 1995. From 1989 to 1994 he was with Telecom Australia Research Laboratories, first as a research engineer, and between 1992 and 1994 as a senior research engineer and project leader. Since December 1994 he has been with the Department of Electronic Engineering, City University of Hong Kong, where he is currently an associate professor. Contact him at eeschan@cityu.edu.hk.

Mohsen Guizani (S'85-M'89-SM'99-F'09) received the B.S. (with distinction) and M.S. degrees in electrical engineering, and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a professor and Chair of the Electrical and Computer Engineering Department at the University of Idaho, USA. He was a professor and the Associate Vice President for Graduate Studies at Qatar University, Doha, Qatar. His research interests include Computer Networks, Wireless Communications and Mobile Computing, and Optical Networking. He currently serves on the editorial boards of six technical Journals and the Founder and EIC of "Wireless Communications and Mobile Computing" Journal published by John Wiley (http://www.interscience.wiley.com/jpages/1530-8669/). Dr. Guizani is an IEEE Fellow and a Senior member of ACM.