International Journal of Networking
ISSN: 2249-278X & E-ISSN: 2249-2798, Volume 2, Issue 1, 2012, pp.-28-31.
Available online at http://www.bioinfo.in/contents.php?id=108

K-MEDIAN BASED NETWORK ATTACK DETECTION

PATNE J.D.[1] AND CHATARE S.S.[2]

M. Tech CSE REC, Bhopal, MP, India.
*Corresponding Author: Email- [1]jpatne@gmail.com, [2]samiksha709@gmail.com

Received: February 21, 2012; Accepted: March 15, 2012

Abstract- The clustered detection of network attacks represents an extremely challenging goal. Current methods rely on either very specialized signatures of previously seen attacks, or on expensive and difficult to produce labeled traffic datasets for profiling and training. In this paper we present a completely clustered approach to detect attacks, without relying on signatures, labeled traffic, or training. The method uses robust clustering techniques to detect anomalous traffic flows, sequentially captured in a temporal sliding-window basis. The structure of the anomaly identified by the clustering algorithms is used to automatically construct specific filtering rules that characterize its nature, providing easy-to-interpret information to the network operator. In addition, these rules are combined to create an anomaly signature, which can be directly exported towards standard security devices like IDSs, IPSs, and/or Firewalls. The clustering algorithms are highly adapted for parallel computation, which permits to perform the unsupervised detection and construction of signatures in an online basis. We evaluate the performance of this new approach to discover and to build signatures for different network attacks without any previous knowledge, using real traffic traces. Results show that knowledge-independent detection and characterization of network attacks is possible, opening the door to a whole new generation of autonomous security algorithms.

Citation: Patne J.D. and Chatare S.S. (2012) K-Median Based Network Attack Detection. International Journal of Networking, ISSN: 2249-278X & E-ISSN: 2249-2798, Volume 2, Issue 1, pp.-28-31.

Introduction

The detection of network attacks is a paramount task for network operators in today's Internet. Denial of Service attacks (DoS), Distributed DoS (DDoS), network/host scans, and spreading worms or viruses are examples of the different attacks that daily threaten the integrity and normal operation of the network. The principal challenge in automatically detecting and analyzing network attacks is that these are a moving and ever-growing target [1].

Two different approaches are by far dominant in the literature and commercial security devices: signature-based detection and anomaly detection. Signature-based detection systems are highly effective to detect those attacks which they are programmed to alert on. However, they cannot defend the network against unknown attacks. Even more, building new signatures is expensive and time-consuming, as it involves manual inspection by human experts. Anomaly detection uses labeled data to build normal-operation-traffic profiles, detecting anomalies as activities that deviate from this baseline. Such methods can detect new kinds of network attacks not seen before. Nevertheless, anomaly detection requires training to construct normal-operation profiles, which is time-consuming and depends on the availability of purely anomaly-free traffic data-sets. In addition, it is not easy to maintain an accurate and up-to-date normal-operation profile. In this paper we present a completely unsupervised method to detect and characterize network attacks, without relying on signatures, training, or labeled traffic of any kind. Our approach relies on robust clustering algorithms to detect both well-known as well as completely unknown attacks, and to automatically produce easy-to-interpret signatures to characterize them, both in an on-line basis. The analysis is performed on packet-level traffic, captured in consecutive time slots of fixed length ΔT and aggregated in IP flows (standard *5-tuples*). IP flows are additionally aggregated at 9 different *flow* levels *li*. These include (from finer to coarsergrained resolution): *source IPs* (*l*1: IPsrc), *destination IPs* (*l*2: IPdst), *source Network Prefixes* (*l*3,4,5: IPsrc/24, /16, /8), *destination Network Prefixes* (*l*6,7,8: IPdst/24, /16, /8), and *traffic per Time Slot* (*l*9: tpTS).

The complete detection and characterization algorithm runs in three successive stages. The first step consists in detecting an anomalous time slot where an attack might be hidden. For doing so, time series *Zli t* are built for basic traffic metrics such as number of bytes, packets, and IP flows per time slot, using the 9 flow resolutions *l*1...9. Any generic anomalydetection algorithm *F*(.) based on time-series analysis [2]–[6] is then used on *Zli t* to identify an anomalous slot. Time slot *t*0 is flagged as anomalous if *F*(*Zli t*0) triggers an alarm for any of the *li* flow aggregation levels. Tracking anomalies at multiple aggregation levels provides additional reliability to the anomaly detector, and permits to detect both single sourcedestination and distributed attacks of very different intensities.

In our paper we using the K-median algorithm for the clustering. In our algorithm we have tried to use the algorithm for clustering data streams that results in a constant factor approximation in one pass using storage space O(k poly log n). Here, the data stream is divided into chunks of data which is mined in phases. In each phase, we find out the weighted medians (facilities).

The data that has been read is deleted from memory and
is replaced by the weighted medians. The facilities that have no increase in weight are considered as temporal candidate outliers for that phase. The next phasei+1 takes the data that has not been read along with the weighted medians found till phasei. The temporal outliers are checked for its outlierness for a given number of phases after which it is either declared as an inlier or a real outlier. After it has been declared as a real outlier it is further never used for clustering. The total cost of the product is the sum of all the facility costs(depending on
the number of medians) and the service costs (cost of assigning a point to an already opened closest facility).

Related Work and Contribution

The problem of network attacks and anomaly detection has been extensively studied in the last decade. Most approaches analyze statistical variations of traffic volume-metrics (e.g., number of bytes, packets, or flows) and/or other traffic features (e.g. distribution of IP addresses and ports), using either single link measurements or network-wide data. A non-exhaustive list of methods includes the use of signal processing techniques (e.g., ARIMA, wavelets) on single-link traffic measurements [2], [3], PCA [8], [9] and Kalman filters [5] for network-wide
anomaly detection, and sketches applied to IP-flows [4], [7]. Our approach falls within the unsupervised anomaly detection domain. Most work has been devoted to the Intrusion Detection field, targeting the well known KDD'99 data-set. The detection schemes
proposed in the literature are based on clustering and outliers detection, being [17]–[19] some relevant examples. In [17], authors use a single-linkage hierarchical clustering method to cluster data from the KDD'99 data-set, based on the standard Euclidean distance for inter-patterns similarity. [18]
reports improved results in the same data-set, using three different clustering algorithms: Fixed-Width clustering, an optimized version of *k*-NN, and one class SVM. [19] presents a combined density-grid-based clustering algorithm to improve computational complexity, obtaining similar detection results.

Our unsupervised algorithm has several advantages w.r.t. the state of the art: (i) first and most important, it works in a completely unsupervised fashion, which means that it can be directly plugged-in to any monitoring system and start to work from scratch, without any kind of calibration or previous knowledge.

Problem Definition

In this section, we discuss an algorithm (KORM) to find outliers in data streams which will be better than the already existing methods. Here we formally define data streams, kmedian objective and outlier detection over data streams.

Definition 1 (Data Stream): A Data Stream X = {x1 x2, . ..,x n } possibly infinite series of objects. xi is represented by n dimensional vector i.e., xi = (x1i, x2i, …………., xni) therefore data projected to the kth dimension from the data stream X is represented as

Xk= x1k, x2k, …………., xnk

Definition 2 (k-median): Given a set X of n points from some metric space, an integer k, and k members c1,.., ck of the metric space, the k–Median cost of using c1,. . . , ck as medians for X (or, simply, the k–Median cost of c1,. . . , ck on X) is Σ x _ X min1 ≤ i ≤ k { dist (x, ci)}.

So the cost of a set of medians is the value of the k–Median objective function. The k-median objective function is to minimize the sum of assignment distances. Each assignment distance over real spaces is replaced by their squares.

Definition 3 (Outlier): Given a data space in multiple dimensions A1,……,Am with domains D1,……,Dm respectively, let the data stream D be a sequence of data objects, where each data object t _ D1 X……X Dm.. Our task is to online detect if a new coming data is an outlier.

Let data stream objects are elements of a metric space on which we can define a distance function. We consider a data stream as chunks of data

X= X 1, X 2, …………, X m

where every chunk contains specified number of n points.

In order to deal with the processing of data stream to find outliers we proposed an algorithm, that would
for a given chunk of data find weighted medians (the weight of the median is the number of points assigned to it) and temporary outliers and delete that summary chunk of data from memory. By this way the freed memory could be used for the next upcoming data chunk.

In Fig. 1 below, only the weighted medians marked with black are passed to the next phase and the points allotted to the medians are deleted.

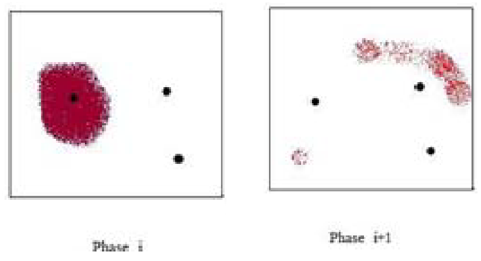

Fig.1- Weighted medians passed from Phase i to Phase i+1

In our method we do not declare a point as an outlier for the data stream but call it a temporal outlier for the given chunk of data. This point may be an outlier for the present data but may not be an outlier for the next data chunk as the data stream is dynamic. We would check this temporal candidate outlier for a given number of data chunks and if it is an outlier for the next given fixed number of stream chunks we declare it as an outlier for the data stream and is not included further for clustering.

Most of the existing work use k-NN approaches for outlier detection which involves parameters R and k, to be provided by the user and require pair wise distance calculations which makes it computationally expensive. The clustering approach for outlier detection using k-mean provides a better solution than k-NN but it is very rigid about the value of k. It provides good clustering results and at the same time deserves good scalability. However, our algorithm for detecting outliers relaxes the value of k and uses minimum k-medians(facilities) and the maximum k log(n). The flexibility in the value k, assures better stability to our k-median solution than the k-means. It runs in polylogarithmic space i.e. O(k poly log(n)) and results in a constant factor O(1) approximation, the algorithm is randomized and has high probability.

Automatic Characterization of Attacks

The following task after the detection of a group of anomalous flows is to automatically produce a set of *K* filtering rules *fk*(**Y**), *k* = 1,. .,*K* to characterize them. In the one hand, such filtering rules provide useful insights on the nature of the anomaly, easing the analysis task of the network operator. On the other hand, different rules can be combined to construct a

signature of the anomaly, which can be used to easily detect its occurrence in the future. To produce filtering rules *fk*(**Y**), the algorithm selects those sub-spaces **X***i* where the separation between the anomalous flows and the rest of the traffic is the biggest. We define two different classes of filtering rule:

absolute rules *fA*(**Y**) and *relative* rules *fR*(**Y**). Absolute rules are only used in the characterization of small-size clusters, and correspond to the presence of dominant features in the flows of the anomalous cluster. An absolute rule for feature *j* has the form *fA* (**Y**) = *{***y***i* ∈ **Y** : *xi*(*j*) == *λ}*. For example,

in the case of an ICMP flooding attack, the vast majority of the associated flows use only ICMP packets, hence the absolute filtering rule *{*nICMP*/*nPkts == 1*}* makes sense (nICMP*/*nPkts corresponds to the fraction of ICMP packets).

On the other hand, relative filtering rules depend on the relative separation between anomalous and normal-operation flows. Basically, if the anomalous flows are well separated from the rest of the traffic in a certain partition *Pi*, then the features of the corresponding sub-space **X***i* are good candidates to define a relative filtering rule. A relative rule defined for feature *j* has the form *fR*(**Y**) = *{***y***i* ∈ **Y** : *xi*(*j*) < *λ* or *xi*(*j*) > *λ}*. We shall also define a *covering relation* between filtering rules: we say that rule *f*1 *covers* rule *f*2 ↔ *f*2(**Y**) ⊂ *f*1(**Y**). If two or more rules overlap (i.e., they are associated to the same feature), the algorithm keeps the one that covers the rest.

In order to construct a compact signature of the anomaly, we have to devise a procedure to select the most discriminant filtering rules. Absolute rules are important, because they define inherent characteristics of the anomaly. Regarding relatives rules, their relevance is directly tied to the degree of separation between flows. In the case of outliers, we select the *K* features for which the normalized distance to the normaloperation traffic (statistically represented by the biggest cluster in each sub-space) is among the top-*K* biggest distances. In the case of small-size clusters, we rank the degree of separation to the rest of the clusters using the well-known Fisher Score (FS) [16], and select the top-*K* ranked rules. The FS basically measures the separation between clusters, relative to the total variance within each cluster. To finally construct the signature, the absolute rules and the top-*K* relative rules are combined into a single inclusive predicate, using the covering relation in case of overlapping rules.

Experimental Evaluation

We evaluate the ability of the unsupervised algorithm to detect and to automatically construct a signature for different attacks in real traffic from the WIDE project data repository [20]. The WIDE network provides interconnection between different research institutions in Japan, as well as connection to different commercial ISPs and universities in the U.S.. Traffic consists of 15 minutes-long raw packet traces; the traces we shall work with consist of packets captured at one of the trans-pacific links between Japan and the U.S.. Traces are not labeled, thus our analysis will be limited to show how the unsupervised approach can detect and characterize different network attacks without using signatures, labels, or learning. We shall begin by detecting and characterizing a distributed SYN network scan directed to many victim hosts under the same */*16 destination network. Packets in **Y** are aggregated using IPdst*/*24 flow resolution, thus the attack is detected as a small-size cluster. The length of each time slot is Δ*T* = 20 seconds. As we explained in section III, the SSC-EAbased clustering algorithm constructs a new similarity measure between flows in **Y**, using the multiple clustering results obtained from the different sub -spaces. Let us express this new similarity measure as a *n*×*n* matrix *S*, in which element *S*(*i, j*) represents the degree of similarity between flows *I* and *j*. Figure 1.(a) depicts a histogram on the distribution of inter-flows similarity, according to *S*. The structure of flows in **Y** provided by *S* evidences the presence of a small isolated cluster in multiple sub-spaces. Selecting this cluster results in 53 anomalous IPdst*/*24 flows; a further analysis of the packets in these flows reveals multiple IP flows of SYN packets with the same IPsrc address and sequential IPdst addresses, scanning primary the same TCP port. Such a behavior is characteristic of a worm in the spreading phase. Regarding filtering rules, figures 1.(b,c) depict some of the partitions *Pi* where both absolute and top-*K* relative rules were produced. These involve the number of sources and destinations, and the fraction of SYN packets. Combining them produces a signature that can be expressed as (nSrcs == 1) ∧ (nDsts > *λ*1) ∧ (nSYN*/*nPkts > *λ*2),where both *λ*1 and *λ*2 are obtained by separating clusters at half distance. Surprisingly enough, the extracted signature matches quite closely the standard signature used to detect such an attack in current signature-based systems [10]. The beauty and main advantage of our unsupervised approach relies on the fact that this new signature has been produced without any previous information about the attack or baseline traffic, and now it can be directly exported towards any security device to rapidly detect the same attack in the future. IP flows are now aggregated according to IPsrc resolution. The distribution analysis of inter-flows similarity w.r.t. *S* selects a compact cluster with the most similar flows, corresponding to the set of attacking hosts. The obtained signature can be expressed as (nDsts == 1) $\wedge$ (nSYN/nPkts > λ3) $\wedge$ (nPkts/sec > λ4), which combined with the large number of identified sources (nSrcs > λ5) confirms the nature of a SYN DDoS attack. This signature is able to correctly isolate the most aggressive hosts of the DDoS attack, i.e., those with highest packet rate.

The detection of an ICMP flooding DoS attack. Traffic is aggregated in IPdst flows, thus the attack is now detected as an outlier rather than as a smallsize cluster. Absolute rules are not applicable in the case of outliers detection. Relative rules correspond to the separation of the outlier from the biggest cluster in each subspace, which statistically represents normal-operation traffic. Besides showing typical characteristics of this attack, such as a high packet rate of exclusively ICMP packets from the same source host, both partitions show that the detected attack does not involve the largest elephant flows in the time slot. This emphasizes the ability of the algorithm to detect attacks that are not necessarily different from normal-operation traffic in terms of volume, but that they differ in other, less evident characteristics. The obtained signature can be expressed as (nICMP/nPkts > λ6) $\wedge$ (nPkts/sec > λ7).

More evaluation results can be found at [12], including an evaluation of true-positives/false-alarm rates, as well as a comparison against other methods for unsupervised anomaly detection. These results confirm the outperforming ability of our approach in the unsupervised anomaly detection domain.

Computational Time and Parallelization

The last issue that we analyze is the Computational Time (CT) of the algorithm. The SSC-EA-based algorithm performs multiple clusterings in *N*(*m*) low-dimensional sub-spaces $\mathbf{X}_i \subset \mathbf{X}$. This multiple computation imposes scalability issues for on-line detection of attacks in very-high-speed networks. Two key features of the algorithm are exploited to reduce scalability problems in number of features *m* and the number of aggregated flows *n* to analyze. Firstly, clustering is performed in very-low-dimensional subspaces, $\mathbf{X}_i \in$ R2, which is faster than clustering in high-dimensional spaces [15].

Secondly, each sub-space can be clustered independently of the other sub-spaces, which is perfectly adapted for parallel computing architectures. Parallelization can be achieved in different ways: using a single multi-processor and multi-core machine, using network-processor cards and/or GPU (Graphic Processor Unit) capabilities, using a distributed group of machines, or combining these techniques. We shall use the term ”slice” as a reference to a single computational entity.

Conclusion

The completely unsupervised algorithm for detection of network attacks that we have presented has many interesting advantages w.r.t. previous proposals. It uses exclusively unlabeled data to detect and characterize network attacks, without assuming any kind of signature, particular model, or canonical data distribution. This allows to detect new previously unseen network attacks, even without using statisticallearning. By combining the notions of Sub-Space Clustering and multiple Evidence Accumulation, the algorithm avoids the lack of robustness of general clustering approaches, improving the power of discrimination between normal-operation and anomalous traffic. We have shown how to use the algorithm to automatically construct signatures of network attacks without relying on any kind of previous information. We claim that such an approach can be used do devise autonomous network security systems, in which the SSC-EA-based algorithm runs in parallel to any standard security device, producing specific signatures to unknown anomalous events. Finally, and contrary to previous work on clustering for detection of network attacks, we have evaluated the computational time of our algorithm. Results confirm that the use of the algorithm for on-line unsupervised detection and automatic generation of signatures is possible and easy to achieve for the volumes of traffic that we have analyzed. Even more, they show that if run in a parallel architecture, the algorithm can reasonably scale-up to run in high-speed networks, using more traffic descriptors to characterize network attacks.

References

[1] Hansman S., Hunt R. (2005) *Computers and Security*, 24(1), 31-43.

[2] Barford P., Kline J., Plonka D., Ron A. (2002) *ACM IMW*.

[3] Brutlag J. *Aberrant Behavior Detection in Time Series for Network Monitoring*.

[4] Manzoor Elahi, Kun Li, Wasif Nisar, Xinjie Lv, Hongan Wang (2008) *Fifth International Conference on Fuzzy Systems and Knowledge Discovery*.

[5] Angiulli F. and Fassetti F. (2007) *Sixteenth ACM Conf. on information and Knowledge Management*.

[6] Cormode G. and Muthukrishnan S. (2005) *IEEE Trans. on Net.*, 13(6), 1219-1232.