Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Layer-7 DDOS Flood Attack Detection Based On Document Rank

RameshBanoth[1]
PG Scholar, Dept. of CSE
TKR College of Engineering and Technology, Hyderabad, India.

T. Radhika[2]
Assistant Professor, Dept. of CSE
TKR College of Engineering and Technology, Hyderabad, India.

P.V.S. Srinivas[3]
Professor & Head, Dept. of CSE
TKR College of Engineering and Technology, Hyderabad, India.

Abstract—Anomaly detection is an important problem that has been researched within diverse research areas and application domains. Many anomaly detection techniques have been specifically developed for 11certain application domains, while others are more generic. In this paper, we used a different Page Rank algorithm at peak in proximity graph collection of data points indicated by vertices, gives results a score quantifying the extent to which each data point is anomalous. In this way to achieve the target, firstly we require forming a density calculation using the training data and it is highly calculative intensive for sets of high-dimensional data. In this paper, we explored page Rank probability in point-wise consistent density that imagines the data points in an asymptotic sense and decreased computational effort only in the case of mild assumptions and appropriately chosen parameters. With the betterments in executing time, while maintaining similar detection performance, this way is computationally well brought-up and scales perfectly to an enormous high-dimensional data sets.

Keywords—Anomaly Detection, Proximity Graph, Personalized Page-Rank

I. INTRODUCTION

Distributed Denial of service (DDoS) attacks is the most undetectable attacks in the internet. DDoS attacks such as ICMP flooding, SYN flooding, and UDP flooding, can be detected by the firewall in general [3] at network layer of TCP/IP. Anomaly detection, also known as outlier detection, refers to the problem of discovering data points or patterns in a given dataset that do not conform to some normal behavior. These techniques are applied in a variety of domains, including credit card fraud prevention, financial turbulence detection, virus or system intrusion discovery, and network monitoring[6]. We can view anomaly detection as a binary classification problem, with one class being anomalous and the other normal. The existing systems work on IP log analysis and the paper is on graph based detection [2]. Here, the normal user profile is represented as a graph with document as a node. In the testing phase the current user profile is represented as a graph path. In the later stages it deals with the case where both classes are of relatively equal size; this is not the case in anomaly detection [8]. Since anomalies, by definition, deviate from the normal pattern, they usually represent situations where something goes wrong with the system (e.g., a malfunction, misuse, or malevolent behavior), and thus they are rarely observed.

II. RELATED WORK

The standard approach in unsupervised statistical anomaly detection has been to assume that the data are drawn from a mixture of outlier and nominal distributions, and to estimate level sets of the nominal density. Scott and Nowak extend the Neyman-Pearson hypothesis testing framework to general supervised learning problems. Based on this extension, they derive a decision region using minimum volume (MV) sets in, providing false alarm control. Later, Scott and Kolaczyk generalize this hypothesis testing framework to the unsupervised case, where measurements are no longer assumed to come from the nominal distribution alone. Meanwhile, they incorporate a multiple testing framework, where the false discovery rate is controlled rather than false alarm errors. All of the methods mentioned above involve intensive computation, which is undesirable especially for large, high-dimensional data. We address this problem by taking an alternative graph-based approach. Another line of previous work is based on forming a graph from the data using the distances between data points. For example, a k- nearest neighbor (kNN) graph is constructed first, and then the distances from each data point to its k^{th} nearest neighbor are used to identify anomalies. These distances are ranked in descending order, and either a threshold is applied or the top m candidates are declared anomalous.

Breunig defined a related quantity called local outlier factor, which is a degree depending on how isolated one data point is with respect to the surrounding neighborhood, to better accommodate heteroscedastic data sources. Pokrajac extended the local outlier factor approach in an incremental online fashion. Zhao and Saligrama propose a non-parametric anomaly detection algorithm based on kNN graphs trained using only nominal data points, which provides optimal false alarm control asymptotically. Our work is motivated by both directions mentioned above. We combine the graph approach together with random walk models, providing false alarm controls in an asymptotic sense.

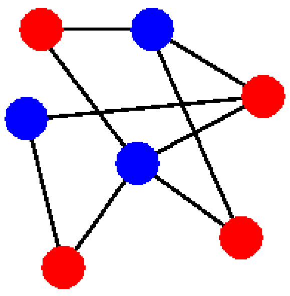

Fig.1: Sample Bipartite Graph

Janeja and Atluri proposed a graph-based anomaly detection algorithm in an active learning setting, where the density information is used to reduce the number of inquiries made to the oracle and their algorithm builds on earlier work which uses graph-based methods for density estimation. Cheng et al. exploited random walks for finding anomalies in time sequences. Sun et al. also investigated anomalous patterns using a Page Rank-like method. However, they focus mainly on bipartite graphs shown in figure 1, while we are discussing much more general distributions and graphs. In this paper, we aim to find anomalous nodes in a graph induced by high dimensional measurements. The partition-based algorithm proposed in this section prunes out points whose distances from their Kth nearest neighbors are so small that they cannot possibly make it to the top n outliers.

III. SYSTEM STUDY

The necessity of a Web page is based on subjective, which depends on the users thinking and attitudes. This gives Page Rank, a method for rating Web pages objectively and mechanically, perfectly calculating the user's choice [1]. We differentiate Page Rank to an idealized random Web surfer. We explore how to efficiently compute Page Rank for large numbers of pages and the way of using Page Rank to search and to user navigation. To calculate the relative importance of web Pages, we often choose Page Rank, a method for computing a ranking for each web page based on the graph. Page Rank has applications in search, browsing, and traffic guess [1]. It gives a mathematical description of Page Rank and provides some intuitive justification. We provide an efficient way to compute Page Rank for as many as 518 million hyperlinks and to test the utility of Page Rank for search; we built a web search engine called Google. In this we also demonstrate how Page Rank can be used as a browsing aid [9].

3.1) Link Structure of the Web

While estimates vary, the current graph of the crawl able Web has roughly 150 million nodes and 1.7 billion edges. Every page has some number of forward links and back links [5]. We can never know whether we have found all the back links of a particular page but if we have downloaded it, we know its entire forward links at that time. Web pages vary greatly in terms of the number of back links they have [3]. For example, the Netscape home page has 62,804 back links in our current database compared to most pages which have just a few back links. Page Rank provides a more sophisticated method for doing citation counting. The reason that Page Rank is interesting is that there are many cases where simple citation counting does not correspond to our common sense notion of importance [11]. For example, consider a web page that has a link on the Yahoo home page and this page should be ranked higher than many pages with more links but from obscure places. Here, Page Rank is an attempt to see how good an estimate to significance can be obtained just from the link structure [2].

3.2) Propagation of Ranking through Links

Based on the discussion above, we give the following intuitive description of Page Rank: a page has high rank if the sum of the ranks of its back links is high. This covers both the case when a page has many back links and when a page has a few highly ranked back links [5].

Definition of PageRank: Let u be a web page. Then let Fu be the set of pages u points to and Bu be the set of pages that point to u. Let Nu = jFuj be the number of links from u and let c be a factor used for normalization [6].We begin by defining a simple ranking, R which is a slightly simplified version of Page Rank:

$$R(u) = c\sum_{v \in Bu} R(v)/Nv$$

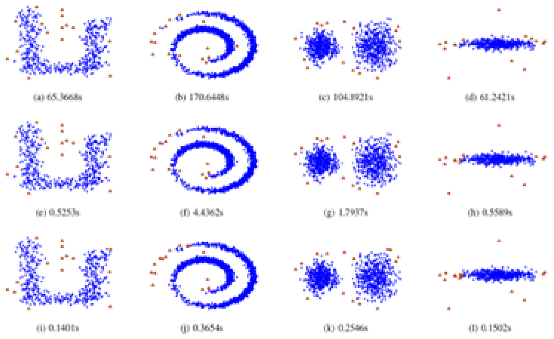

Fig.2: Demonstrating the propagation of rank

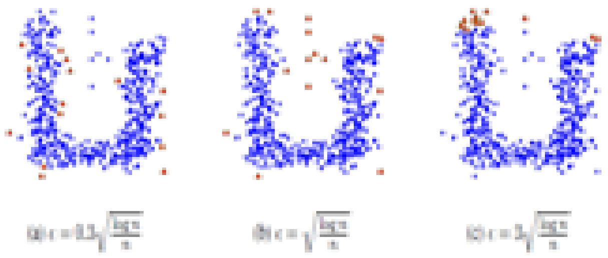
Fig.3: Representing the consistent steady state solution

The rank of a page is divided among its forward links evenly to contribute to the ranks of the pages they point to. Note that c < 1 because there are a number of pages with no forward links and their weight is lost from the system [10].The equation is recursive but it may be computed by starting with any set of ranks and iterating the computation until it converges. Figure 2 demonstrates the propagation of rank from one pair of pages to another [11]. Figure 3 shows a consistent steady state solution for a set of pages. Stated another way, let A be a square matrix with the rows and column corresponding to web pages. Let Au; v = 1=Nu if there is an edge from u to v and Au; v = 0 if not. If we treat R as a vector over web pages, then we have R = cAR. So R is an eigenvector of A with Eigen value c[4].In fact, we want the dominant eigenvector of A. It may be computed by repeatedly applying A to any non degenerate start vector. There is a small problem with this simplified ranking function. Consider two web pages that point to each other but to no other page. And suppose there is some web page which point to one of them [8]. Then, during iteration, this loop will accumulate rank but never distribute any rank. The loop forms a sort of trap which we call a rank sink.

IV. SYSTEM DEVELOPMENT

During the development, we undergo two different phases. They are training phase and testing phase.

4.1) Training Phase:
Module 1: *Access Log Parsing*
The user accesses are stored in the access log file. The files cannot be used for direct comparison [9]. The file is preprocessed to identify Client IP, Request, and Referrer from each user access log.
Module 2: *Document Matrix*
The module identifies the access frequency for each document
It can be calculated as:
= (No.Of.Hits for a page per user)/ (Total Number of Logs)
The value always ranges in between 0 to 1. Training time Document Matrix represents the standard user access behavior.
4.2) Testing Phase
A. User Request Access
i) The module identifies the user requested (URI). It also identifies the referrer URI.
ii) The user profile is stored for further processing
B. Document Matrix
i) For every fixed interval of time, The user-profiles are processed for calculating the DM.
ii) Each individual user DM prepared.
iii)The DM rank indicates the document rank [11].
C. Anomaly Detection
i) User DM is cross compared with the training time DM. ii) If any URI crosses or under flows the Training Time DM.
ii) For a predefined threshold.
iii) The user is treated as an anomalous user.
iv) The anomalous users are reported to the administrator.
D. Administration Interface
i) The system monitors the anomalous activity
ii) The anomalous behavior of any user is reported to administrator.
iii) It allows login, view the anomalous activity.

V. SYSTEM EVALUATION

We call our framework Anomaly Detection using Proximity Graph and Page Rank (ADPP). The steps of this framework are outlined in the below given algorithm. The algorithm takes three input arguments each measurement xi is itself, a d-dimensional point and the weight function f. We consider the identity weight and the Gaussian weight, both of which are non-increasing functions with respect to distances between nodes. Here, the teleport vector t which specifies the jumping probability.
Algorithm 1: Procedural steps for ADPP Algorithm
Input: the observations {xi}, the weight function 'f 'and the teleport vector't'.
Output: the Page Rank vector's'.

Step1: Compute pair wise distances among measurements.
Step 2: Determine vicinity criteria to form a proximity graph.
Step 3: Apply the wait function 'f' to obtain similarity matrix 'W'.
Step 4: Normalize 'W' to get transition matrix 'P'.
Step 5: Solve's' for $s^T=\alpha s^T P+(1-\alpha)t^T$.
Step 6: Sort's' in ascending order and output the top few points as anomalies.

VI. CONCLUSION

In this work, we propose a framework for anomaly detection using proximity graphs and the PageRank algorithm. This is an unsupervised, nonparametric, density estimation-free approach, readily extending to high dimensions. Various parameter selection, time complexity guarantees and possible extensions are discussed and investigated. One straightforward extension is to formalize the problem of semi-supervised anomaly detection, when partial labels are available. The label information can be adapted into our framework without difficulty by changing the teleport vector t accordingly in a more deliberate way. Another direction is to make the framework online. At this stage, our algorithm operates in a batch mode. Given a set of observations, after announcing the potential anomalies once, the algorithm terminates. Once a new observation is available, we do not want to run the whole algorithm from been explore n to be O(n2), which is not desirable in the online fashion. In reality, especially in high dimension cases, not all of them are helpful. The inclusion of noisy dimensions may even hurt the performance. Therefore, it will be better if our framework has some feature selection ability support built in, to filter out those unwanted dimensions.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, ―Anomaly detection: A survey,‖ACM Computing Surveys, vol. 41, no.3, pp. 15:1–15:58, 2009.
[2] L. Page, S. Brin, R. Motwani, and T. Winograd, ―The PageRank citation ranking: Bringing order to the web,‖Stanford InforLab, Tech. Rep. 1999-66, 1999.
[3] B. Sch¨olkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R.Williamson, ―Estimating the support of a high-dimensional distribution,‖ Neural computation, vol. 13, no. 7, pp. 1443–1471, 2001.
[4] C. Scott and R. Nowak, ―A Neyman-Pearson approach to statistical learning,‖ IEEE Transactions on Information Theory, vol. 51, no. 11,pp. 3806–3819, 2005.
[5] Learning minimum volume sets,‖ Journal of Machine Learning Research, vol. 7, pp. 665–704, 2006.
[6] C. Scott and E. Kolaczyk, ―Nonparametric assessment of contamination in multivariate data using generalized quantile sets and fdr,‖ Journal of Computational and Graphical Statistics, vol. 19, no. 2, pp. 439–456,2010.
[7] A. Hero III, ―Geometric entropy minimization (GEM) for anomaly detection and localization,‖ in Proc. Advances in Neural Information Processing Systems, vol. 19, Vancouver, BC, Canada, 2006, pp. 585–592.
[8] S. Byers and A. Raftery, ―Nearest-neighbor clutter removal for estimating features in spatial point processes,‖ Journal of the American Statistical Association, vol. 93, no. 442, pp. 577–584, 1998.
[9] S. Ramaswamy, R. Rastogi, and K. Shim, ―Efficient algorithms for mining outliers from large data sets,‖ ACM SIGMOD Record, vol. 29,no. 2, pp. 427–438, 2000.
[10] M. Breunig, H. Kriegel, R. Ng, and J. Sander, ―OPTICS- OF: Identifying local outliers,‖ in Proc. European Conference on Principles of Data Mining and Knowledge Discovery, Prague, Czech Republic, 1999, pp.262–270.
[11] LOF: Identifying density-based local outliers,‖ ACM SIGMOD Record, vol. 29, no. 2, pp. 93–104, 2000.