Adaptive Framework for Network Traffic Classification Using Dimensionality Reduction and Clustering

Antti Juvonen, Tuomo Sipola
Department of Mathematical Information Technology
University of Jyväskylä
Jyväskylä, Finland
{*antti.k.a.juvonen, tuomo.sipola*}*@jyu.fi*

***Abstract*—Information security has become a very important topic especially during the last years. Web services are becoming more complex and dynamic. This offers new possibilities for attackers to exploit vulnerabilities by inputting malicious queries or code. However, these attack attempts are often recorded in server logs. Analyzing these logs could be a way to detect intrusions either periodically or in real time. We propose a framework that preprocesses and analyzes these log files. HTTP queries are transformed to numerical matrices using n-gram analysis. The dimensionality of these matrices is reduced using principal component analysis and diffusion map methodology. Abnormal log lines can then be analyzed in more detail. We expand our previous work by elaborating the cluster analysis after obtaining the low-dimensional representation. The framework was tested with actual server log data collected from a large web service. Several previously unknown intrusions were found. Proposed methods could be customized to analyze any kind of log data. The system could be used as a real-time anomaly detection system in any network where sufficient data is available.**

***Keywords*-intrusion detection; anomaly detection; n-grams; diffusion map; k-means; data mining; machine learning**

I. Introduction

Most web servers log their traffic. This log data is rarely used, but it could be analyzed in order to find anomalies or to visualize the traffic structure. Acquiring the data does not require any modifications to the actual web service, because data logging is usually done by default. Different kinds of log files are created, but for this study the most interesting log is the one containing HTTP queries.

One important application for network traffic analysis is anomaly detection. This is done using *intrusion detection systems* (IDS) [1]. Many of these analyze the transport layer, mostly TCP packet data. However, we try to find anomalies and other information from application layer log files. HTTP queries include this information. Many attacks, such as SQL injections, can be detected from this layer.

Log files are in textual form. Therefore, some preprocessing is needed to transform query strings into numerical matrices. This can be done using information about n-gram analysis, which is described in section III-A. Calculating the frequencies of individual substrings in the data results in a numerical data matrix.

After preprocessing, many data mining methods can be used to visualize and analyze the logs. We perform dimensionality reduction and clustering. After visualizing the results it is possible to interpret the findings and make more detailed analysis about the web service traffic.

We propose a framework that processes textual log files in order to visualize them. We are trying to find patterns and anomalies using only log files containing HTTP queries. The framework is adaptive, and individual parts of it can be changed. For example, the choice of dimensionality reduction method or clustering algorithm can be done based on current needs.

The proposed methods use data mining principles, and they work as an IDS and network traffic visualization and analysis tool. Using the framework, we are trying to find whether the textual HTTP query logs actually include some information about the traffic structure. This information could then be used to classify users and individual queries and to find anomalies and intrusion attempts.

II. Related work

We have previously researched log data preprocessing and anomaly detection [2], [3]. This research focused on finding intrusions from log data. We now extend this methodology to further analyze and cluster the structure of the traffic. This is done by adding more accurate clustering algorithms into the framework.

Principal component analysis has been widely used in network intrusion detection and traffic analysis. Xu et al. used PCA and support vector machine to reduce dimensions and classify network traffic in order to find intrusions [4]. Taylor et al. used PCA and clustering analysis to find network anomalies and perform traffic screening [5].

Diffusion methods have been applied in network traffic analysis. These studies have concentrated on low-level IP packet features. These features are numerical and the network architecture differs from our study [6] [7]. Network server logs have also been analyzed using diffusion maps and spectral clustering [2] [3].

This is the authors' postprint version of the article. The original print version appeared as: A. Juvonen and T. Sipola, "Adaptive Framework for Network Traffic Classification Using Dimensionality Reduction and Clustering," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on*. IEEE 2012, pp. 274–279. Available online at http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6459678&isnumber=6459638

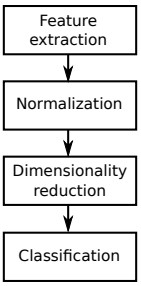

Figure 1. The data mining process

III. Methodology

Our overall approach is rooted in the data mining process [8], [9]. This approach is method-centric as our research is focused on the data processing and not business aspects. The data mining process of our study flows as follows:

1) Data selection.
2) Extract n-gram features from the text data.
3) Normalize the feature matrix.
4) Reduce the number of dimensions to obtain low-dimensional features.
5) Classify or cluster the low-dimensional data presentation.
6) Interpret the found patterns or anomalies.

The process is presented in figure 1.

A. Feature extraction

The log files are in text format. Therefore, it is necessary to transform the log lines into numerical vectors which then can be used in further mathematical analysis. We use n-gram analysis to process log files into numerical matrices. It has been used e.g. in judging similarity in text documents [10], analyzing protein sequences [11] and detecting malicious code [12].

N-grams are consecutive sequences of n characters [10]. Each log line corresponds to a feature vector containing the frequencies of each individual n-gram found in the data. The list of n-grams appearing in the data can be found using n-character-wide sliding window moved along the string one character at a time [10].

Let us consider the following example. Having two strings containing the words `anomaly` and `analysis`, we can construct the feature matrix in the following way:

an	no	om	ma	al	ly	na	ys	si	is
1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1

In this study, 2-grams are used. However, it is possible to use longer n-grams as well. This will of course results in more dimensions in the matrix, because there are more unique n-grams. The theoretical maximum number of individual 2-grams using ASCII-characters is $256^2 = 65536$, but in practice this is usually not the case. This is due to the fact that many characters are never actually used [10].

B. Normalization

Normalization ensures that the features of the input data are in the same scale. We use logarithm for this purpose. To avoid complex numbers, the input must be above zero. The normalization function for a point x_i in the dataset is

$$f_n(x_i) = \log(x_i - X_{min} + 1),$$

where X_{min} is the minimum of all the values in the dataset.

C. Principal Component Analysis

Principal Component Analysis (PCA) [13] is perhaps the best-known dimensionality reduction technique. It has many practical applications, such as computer vision and image compression [14].

The PCA process is explained in more detail in [14]. First we must substract the mean from the original data to make the data have zero mean. Then the covariance matrix must be calculated. From the covariance matrix we can then calculate eigenvalues and the corresponding eigenvectors. If we choose d eigenvectors that contain most of the variance, we get a lower dimension representation of the original data with d dimensions. This is done by choosing the d eigenvectors as columns for a matrix, and multiplying the mean-centered data with this matrix. For visualization purposes it is necessary to choose either 2 or 3 dimensions, ie. eigenvectors.

Calculating PCA is relatively simple, but it will only work in linear cases. If the dataset is non-linear, some other dimensionality reduction method must be used. PCA can also give inaccurate results if there are outliers in the data.

D. Diffusion Map

Diffusion map (DM) reduces the dimensions while retaining the diffusion distances in the high-dimensional space as Euclidean distances in the low-dimensional space. This reduction is non-linear. The goal is to move from n-dimensional space to a low-dimensional space with d dimensions, when $d \ll n$ [15].

One measurement $x_i \in \mathbb{R}^n$ in this study corresponds to one line in the log file. Given the dataset $X = \{x_1,\ x_2,\ x_3,\ \ldots\ x_N\}$ the affinity matrix $W(x_i, x_j) = \exp\left(\frac{-||x_i - x_j||^2}{\epsilon}\right)$ describes the affinities between measurements. Here we have used the Gaussian kernel. Matrix $P = W^{-1}K$ represents the transition probabilities between the measurements. Next, the matrix D collects the row sums to its diagonal. Using the singular value decomposition (SVD) of matrix $\tilde{P} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ we obtain the eigenvectors v_k and eigenvalues λ_k.

The diffusion map maps the measurements x_i to low dimensions by giving each high-dimensional point coordinates in the low dimensions: $x_i \rightarrow [\lambda_1 v_1(x_i),\ \lambda_2 v_2(x_i) \dots\ \lambda_d v_d(x_i)]$. These new coordinates lose some of the information contained in the original dataset. However, the accuracy is usually good enough for later classification. Even though there is loss of information, the classification problem becomes easier.

E. Traffic clustering using k-means algorithm

We use cluster analysis to divide network traffic into meaningful groups. In this way we can capture the natural structure of the data [16].

K-means algorithm was introduced in 1955 and huge number of other clustering algorithms have been introduced since then, but k-means method is still widely used [17]. It is a prototype-based clustering technique [16]. Given the original data $X = x_i$, where $i = 1, .., n$, the goal is to cluster the data points into k clusters. The mean of cluster k is now μ_k, and the mean squared error (MSE) between a data point and the cluster mean is $||x_i - \mu_k||^2$. This leads to an optimization problem where the MSE for each datapoint in each cluster must be minimized.

The problem can be solved following these steps [18]:

1) Select initial centers for k clusters.
2) Assign each datapoint to its closest cluster centroid.
3) Compute the new cluster centers by calculating the mean of the datapoints in each cluster.

Steps 2 and 3 are repeated until a stopping criterion is met. Usually this means that the partitioning has not changed since the last iteration, and thus a local optimum solution for the problem has been found.

Choosing the number of clusters is not trivial, but there are many methods for calculating the number of clusters, such as Davies-Bouldin index, described in [19]. This algorithm takes into account both scatter within a cluster and separation between different clusters. Davies-Bouldin index is used in this study to determine the number of clusters for each resource.

The algorithm can give different results depending on the initialization, because it only finds the local optimal solution. This can happen especially when using random initialization. However, this problem can be overcome by running k-means multiple times and choosing the clustering results that gives the smallest squared error [17]. There are also many other algorithms for choosing the initial cluster centroids.

IV. Experimental setup

Figure 2 shows the architecture of the web service that was analyzed. It contains many servers that offer the same service to users using load balancing. Proprietary log files were acquired from this service. These files then need to be preprocessed into numerical matrices. The data and this process are described in this section.

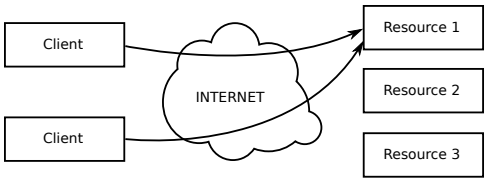

Figure 2. Experiment architecture.

A. Data acquisition

The data have been collected from a large web service. Apache web servers are used, and they log data using Combined Log Format, example of a single log line:

```
127.0.0.1 - -
[01/January/2012:00:00:01 +0300]
"GET /resource.php?parameter1=value1
&parameter2=value2
HTTP/1.1"
200 2680
"http://www.address.com/webpage.html"
"Mozilla/5.0
(SymbianOS/9.2;...)"
```

For this analysis, the HTTP query part is used because it contains the only information that a user can input. This offers possibilities for attackers. The other information, such as time, can be used when further analyzing individual log lines (e.g. for finding anomalies or attacks). On the other hand, HTTP query parameters and their values are dynamic and changing, offering valuable information about this dynamic web service. Analyzing this information will explain a lot about the structure of the traffic. The parameter values in data used in this study were dynamic and changing, and also not always human-readable. Therefore, analyzing these fields has to be done automatically with mathematical methods.

B. Data preprocessing

The first step is to select the data for analysis. The original log file contains approximately 4 million log lines. However, most of these lines contain only static queries. Static lines do not contain changing parameter values. These lines do not offer a lot of information, because they are practically identical in the used dataset. In addition, static lines do not contain information about user input, meaning it is not possible to detect attacks from those log lines alone. On the other hand, dynamic web resources are changing and also vulnerable, so dynamic lines containing parameters and parameter values are interesting and can offer more information about the web service. Therefore, static log lines are filtered out, leaving only approximately 221 000 lines to be inspected and clustered. This data selection reduces the size considerably and creates a database of the most interesting aspects of the log files.

After the first filtering stage, log files are divided into smaller files according to resource URI. This is because different resources accept different parameter values, so they do not have much to do with each other. This makes anomaly detection from full data very difficult and inaccurate. However, traffic structure inside single resource is more consistent. After this division, smaller logfiles can be analyzed independently. It makes sense to further analyze the largest log files, because some of the resources contain only a few lines. These lines have to be omitted.

Finally, in order to create data matrices out of textual log data, n-gram analysis is performed. This process is explained in III-A.

V. Results

For this research, 3 relatively large resources are selected for further analysis and clustering. Resource 1 contains 10935 lines and 414 dimensions, and is the simplest in terms of HTTP query parameters. Resource 2 contains only 2982 lines, but the number of dimensions is 3866, which makes analysis challenging. Also, the parameters are clearly not human-readable, i.e. it is impossible to say anything about the queries by looking at the parameter string alone. Resource 3 is the largest, including 21406 lines and 991 dimensions.

All the resources are analyzed using the proposed framework. The feature data are normalized with the logarithm function. PCA and diffusion map reduce the dimensionality of the normalized feature matrices. Clustering then reveals the structure of the data and facilitates the interpretation of the log files.

Resource 1 contains 10935 lines and 414 dimensions. The results for diffusion map and principal component analysis are presented in figures 3 and 4, respectively. This resource is a simple example, mainly useful in validating that the methods do give satisfactory results. The only difference is that DM separates the data points more clearly. Due to this separation we get 3 clusters, instead of 2 as in PCA. The biggest cluster contains varying parameter values. The parameters in smaller clusters are almost the same within that cluster. However, this behavior is easy to see directly from the log lines. The framework visualizes the traffic well, but in this case we do not obtain any new information about the data.

Resource 2 is the smallest in this research, containing only 2982 requests. However, the number of dimensions is 3866. This means that there are more dimensions than data points, which is always a problem in classification tasks. Despite that, we obtained clear results. The DM and PCA results presented in figures 5 and 6. The results are essentially identical, figures look slightly different but the clustering is exactly the same. This might mean that variables are linearly dependent, otherwise PCA would not work well. The log lines themselves are not human-readable, containing error

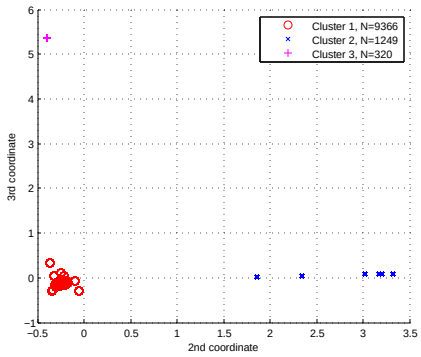

Figure 3. Resource 1, diffusion map.

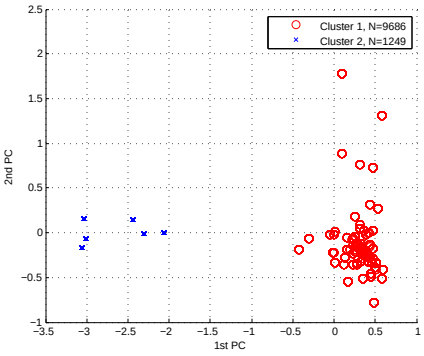

Figure 4. Resource 1, PCA.

tickets that have a seemingly random code as the parameter value. However, as can be seen from the figures, there are clearly two distinct clusters that can be seen using both dimensionality reduction methods. This behavior was not previously known and requires more detailed analysis with the administrator of the web service.

Resource 3 is the largest with 21406 lines and 996 dimensions. It also shows that DM (in figure 7) and PCA (in figure 8) can sometimes give very different results. Normal parameter values in this resource are long and varied. This results in PCA not being able to clearly distinguish any clusters. For this reason, k-means clustering was not performed for resource 3 PCA datapoints. However, with DM the results are very meaningful. Normal traffic clearly forms it's own cluster, while 2 other groups are apparent. Cluster 2 with 5 datapoints does not contain anything malicious, but is slightly different from other normal datapoints. The most interesting finding in this data is cluster 3, which contains 4 lines. All of these lines contain an SQL injection attack, where an attacker tried to include malicious SQL queries as parameter values. The 2nd DM coordinate clearly separates attacks from rest of the data, meaning that in this case only one dimension is needed for anomaly detection.

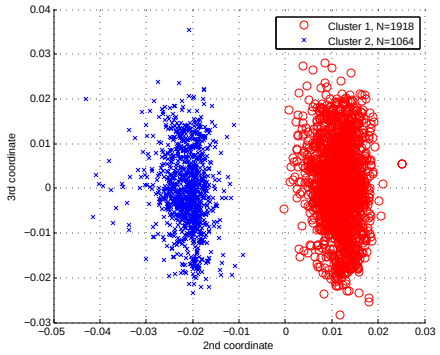

Figure 5. Resource 2, diffusion map.

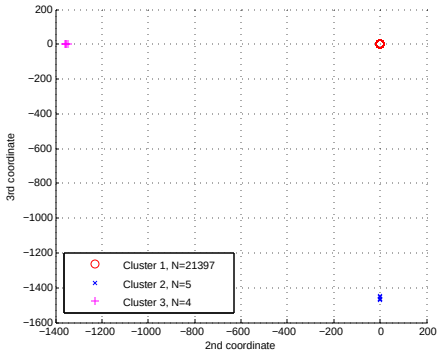

Figure 7. Resource 3, diffusion map.

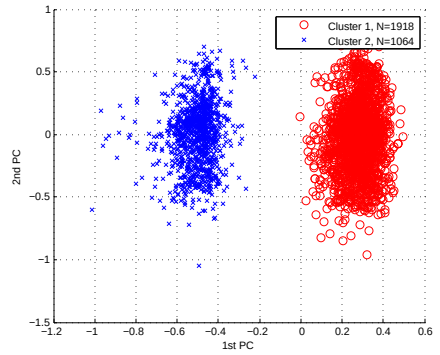

Figure 6. Resource 2, PCA.

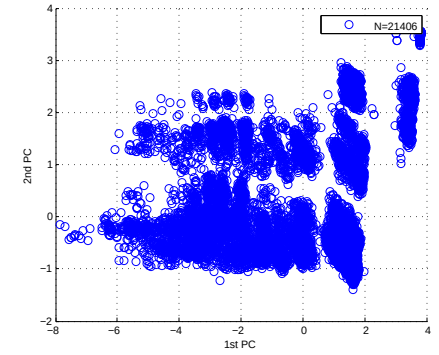

Figure 8. Resource 3, PCA.

In all of the figures, except PCA for resource 3, it can be seen that the separation of clusters is clear. A simple clustering method such as spectral clustering or decision tree could be used.

VI. Conclusion

We presented a framework for preprocessing, clustering and visualizing web server log data. This framework was used for anomaly detection, visualization and explorative data analysis based only on application layer data. Individual parts of the architecture can be changed for different results. For example, k-means clustering can be replaced with hierarchical linkage clustering method.

The results clearly indicate that there are traffic structures that can be visualized from HTTP query information. The data forms distinct clusters and contains anomalies as well. The sensitivity for outliers creates some problems for PCA, which means that it can be challenging to use it for anomaly detection. Diffusion maps give good results, but more research would have to be done to get more information about performance issues. In some cases the results for PCA and DM are nearly identical, while in other cases they differ greatly. PCA is faster but cannot be used with non-linear data. DM seems to work in most situations but can be too slow.

Traffic clustering can give new information about the users of a web service. This information could be used to categorize users more accurately. This gives opportunities for more accurate advertising or offering better content for users. Finding anomalies gives information about possible intrusion attempts and other abnormalities.

To make the framework more usable, it should be automatic and work in real-time. More research is needed to find the most generally usable algorithms for each phase in the architecture. In addition, log data tends to be high in volume, so performance issues might become a problem. For dimensionality reduction the number of dimensions is not trivial. Also, the number of clusters must be determined depending on the chosen clustering algorithm. Real-time functioning requires changes in preprocessing and limits the dimensionality reduction options. For this purpose, PCA might be a good method, since projection of new points into lower dimensions is simply a matter of matrix multiplication. However, the limitations mentioned previously still apply.

Using data mining methods, underlying structure and anomalies are found from HTTP logs and these results can be visualized and analyzed to find patterns and anomalies.

REFERENCES

[1] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," *NIST Special Publication*, vol. 800, no. 2007, p. 94, 2007.

[2] T. Sipola, A. Juvonen, and J. Lehtonen, "Anomaly detection from network logs using diffusion maps," in *Engineering Applications of Neural Networks*, ser. IFIP Advances in Information and Communication Technology, L. Iliadis and C. Jayne, Eds. Springer Boston, 2011, vol. 363, pp. 172–181.

[3] ——, "Dimensionality reduction framework for detecting anomalies from network logs," *Engineering Intelligent Systems*, 2012, forthcoming.

[4] X. Xu and X. Wang, "An adaptive network intrusion detection method based on pca and support vector machines," *Advanced Data Mining and Applications*, pp. 731–731, 2005.

[5] C. Taylor and J. Alves-Foss, "Nate: N etwork analysis of a nomalous t raffic e vents, a low-cost approach," in *Proceedings of the 2001 workshop on New security paradigms*. ACM, 2001, pp. 89–96.

[6] G. David, "Anomaly Detection and Classification via Diffusion Processes in Hyper-Networks," Ph.D. dissertation, Tel-Aviv University, 2009.

[7] G. David and A. Averbuch, "Hierarchical data organization, clustering and denoising via localized diffusion folders," *Applied and Computational Harmonic Analysis*, 2011.

[8] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," *Commun. ACM*, vol. 39, pp. 27–34, November 1996. [Online]. Available: http://doi.acm.org/10.1145/240455.240464

[9] ——, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[10] M. Damashek, "Gauging similarity with n-grams: Language-independent categorization of text," *Science*, vol. 267, no. 5199, p. 843, 1995.

[11] M. Ganapathiraju, D. Weisser, R. Rosenfeld, J. Carbonell, R. Reddy, and J. Klein-Seetharaman, "Comparative n-gram analysis of whole-genome protein sequences," in *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., 2002, pp. 76–81.

[12] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based detection of new malicious code," in *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, vol. 2. IEEE, 2004, pp. 41–42.

[13] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, pp. 417–441, 1933.

[14] L. Smith, "A tutorial on principal components analysis," *Cornell University, USA*, vol. 51, p. 52, 2002.

[15] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[16] P. Tan, M. Steinbach, and V. Kumar, "Cluster analysis: Basic concepts and algorithms," *Introduction to data mining*, pp. 487–568, 2006.

[17] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[18] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice Hall., 1988.

[19] D. Davies and D. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 2, pp. 224–227, 1979.