Online Anomaly Detection Using Dimensionality Reduction Techniques for HTTP Log Analysis

Antti Juvonen[a,1,*], Tuomo Sipola[a,1], Timo Hämäläinen[a]

[a]*Department of Mathematical Information Technology, University of Jyväskylä, Finland*

Abstract

Modern web services face an increasing number of new threats. Logs are collected from almost all web servers, and for this reason analyzing them is beneficial when trying to prevent intrusions. Intrusive behavior often differs from the normal web traffic. This paper proposes a framework to find abnormal behavior from these logs. We compare random projection, principal component analysis and diffusion map for anomaly detection. In addition, the framework has online capabilities. The first two methods have intuitive extensions while diffusion map uses the Nyström extension. This fast out-of-sample extension enables real-time analysis of web server traffic. The framework is demonstrated using real-world network log data. Actual abnormalities are found from the dataset and the capabilities of the system are evaluated and discussed. These results are useful when designing next generation intrusion detection systems. The presented approach finds intrusions from high-dimensional datasets in real time.

Keywords: cyber security, anomaly detection, intrusion detection, principal component analysis, random projection, diffusion map

*Corresponding author. Tel. +358 40 357 3875. Postal address: c/o Timo Hämäläinen, Department of Mathematical Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland.

Email addresses: antti.k.a.juvonen@jyu.fi (Antti Juvonen), tuomo.sipola@iki.fi (Tuomo Sipola), timo.hamalainen@jyu.fi (Timo Hämäläinen)

[1]Current affiliation is with CAP Data Technologies, Finland.

1. Introduction

Web applications and services have become more featured and therefore more complex in recent years. At the same time, the number of different vulnerabilities and intrusion attempts has become more and more common. For detecting these attacks, *intrusion detection systems* (IDS) are used. There are many different types of systems, and they can be divided according to, e.g., detection principle, detection time and the system's place in the network or host. The attacks are changing over time, and therefore IDSs need to adapt to new threats as well.

Intrusion detection systems can generally be divided into two categories based on the detection principle: signature-based and anomaly-based detection [1, 2]. In signature-based systems, manually created rules are created based on known attack patterns. Network behavior is then compared to these rules, and alarm is created if there is a match. The advantages include computational simplicity and being able to determine which type of attack is taking place. However, only previously known attacks can be found using this methodology, while unknown intrusions remain undetected. Another option is to use anomaly-based detection, where any new behavior is compared to the normal behavior patterns in the network. Alarms are created if a deviation from the norm is found. Using this detection principle, it is possible to detect new and unknown intrusion attempts and other anomalies. On the other hand, the biggest possible problem with anomaly detection is high number of false alarms. Intrusion detection systems can be implemented using method based on different approaches, such as statistics, patterns or rules [2].

Most of the traditional intrusion detection systems are based on signature detection. Anomaly detection systems work best when used together with traditional systems, not on their own. Figure 1 shows an example of the placement of intrusion detection system components in a small network. One option is to use anomaly detection to analyze potential intrusions that were not detected by the signature-based system. This will improve the security of critical infrastructure. The scenario represents the test setting used in this research. There are of course many other options for the placement of the anomaly detection system in the network.

In recent years, many machine learning methods have been used to facilitate anomaly detection. The challenge with this approach is that machine learning methods are better at finding similarities than abnormalities. How-

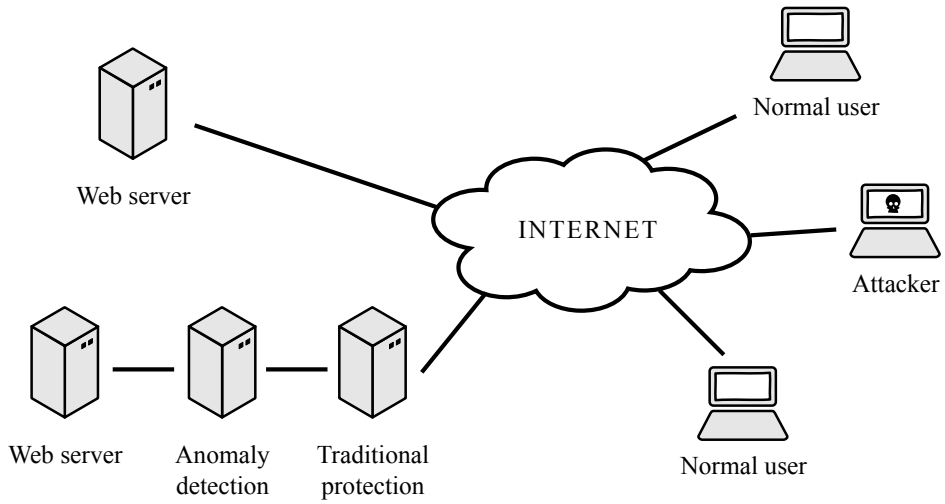

Figure 1: IDS topology.

ever, this does not mean that using machine learning is always unfeasible. To overcome this problem, data selection must be performed with care so that there is a clear context [3]. With careful data selection, preprocessing and feature extraction it is possible to facilitate anomaly detection using machine learning algorithms.

Intrusion detection systems can also be classified as either network-based or host-based [1]. Network-based systems monitor certain parts of the network and scan for suspicious activity from the network traffic. The best place for network-based systems is at the boundary of different network segments. On the other hand, host-based intrusion detection systems monitor a single host. Different kinds of activity can be monitored from a single host, e.g., log files and application activity.

In addition, intrusion detection systems can operate in offline or online mode [4]. Offline systems scan the network behavior periodically to find out if intrusive traffic has occurred since previous scan. Online systems scan new traffic as it arrives, essentially in real time.

A system using the anomaly detection principle was first introduced by Denning [5]. Since then, there has been a lot of research and many different methods and algorithms have been used to facilitate anomaly detection. Very common approaches to intrusion detection in the literature include statistical methods, machine learning and anomaly detection [6]. Examples of methodologies used in this context include neural networks [7, 8, 9], self-organizing

maps (SOMs) [10] and support vector machines (SVMs) [11]. HTTP traffic can be analyzed extracting certain features to build an anomaly-based intrusion system [12]. PCA methodology has been used even quite recently for intrusion detection [13]. A recent study also modified PCA to be able to analyze new data by using online updating technique [14]. This extends the system to large-scale problems, the system does not apply any other dimensionality reduction algorithms and might suffer from nonlinear data. Many of the new research still focuses on intrusion detection using signatures. Anomaly detection continues to be a challenging subject.

The authors of this paper have previously explored network anomaly detection using diffusion map methodology for mostly offline detection [15, 16, 17]. We have also applied a rule extraction algorithm to the framework to create an online detection system [18]. This approach works independently of the anomaly detection algorithms used. Random projection methodology applied to web anomaly detection framework has also been used by the authors [19].

Our new proposed system reads web server log files, extracts the features from the raw logs and finds anomalies using several dimensionality reduction techniques. In addition, we get visualizations that make in-depth analysis easier. The system is capable of online detection when new data points are dynamically added. We use three different dimensionality reduction techniques, some of which are widely used in intrusion detection research, while others are not as commonly applied in this context.

2. System architecture

At first, the system is trained using existing data. The system architecture takes log data as input. These could theoretically be any structured text files. The preprocessing part extracts n-gram features. The training n-gram profile is saved for later. These features are then transformed to a low-dimensional space. Machine learning training uses several dimensionality reduction methods and produces a low-dimensional representation of the data.

When new streaming data arrives, existing n-grams are counted and new ones added to the profile. This way the n-gram dictionary stays up-to-date. These features are then transformed again to the low-dimensional space created in the previously explained machine learning training. The new data

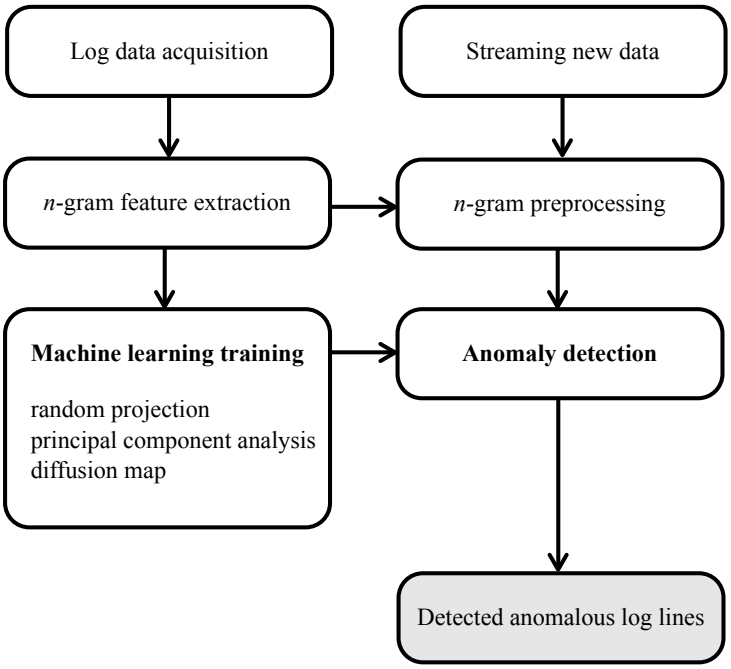

Figure 2: System architecture flow diagram.

points are classified either normal or anomalous in the low-dimensional space. Afterwards, the detected anomalies are output to the user.

Refer to Figure 2 for flow diagram. The left column shows the training and the right column shows new streaming data coming. The n-gram profiles benefit the feature extraction on the right, and the low-dimensional model is used when projecting new streaming data for anomaly detection.

3. Data acquisition and preprocessing

Data acquisition is crucially important for any IDS, because it is the first phase of any intrusion detection framework. Different types of data can be used. In this paper we focus on Hypertext Transfer Protocol (HTTP) log data. It has become a universal transfer protocol and will also be heavily used in the future [20]. After acquiring the data, they must be preprocessed and features must be extracted to transform it into numerical form that can be used for later analysis phases. The data and used preprocessing methods are described in the following subsections.

3.1. Data acquisition

The data used in this research are acquired from real-world company web servers. The servers are running the widely-used Apache web server software. The logs are in *combined log format* [21], which include different kinds of information such as IP address, timestamp, the actual HTTP request, Apache server response code and user agent header field. An example log line is presented below:

```
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;...)"
```

These logs might contain several actual intrusions, especially inside the HTTP requests that are not static, i.e., they contain dynamic parameters that depend on the user input. This is why we focus on analyzing the request strings. Different kinds of attack attempts, such as SQL injections, can be found from web server logs. Even though many intrusions cannot be found without having access to the actual payload data, web server logs are widely available and used by default in most web servers around the world, which is a huge advantage.

3.2. Preprocessing

Acquired log files contain essentially strings describing requests sent from the user to the server. In the preprocessing phase, textual logs are transformed into numerical matrices to facilitate the subsequent analysis phases. In this research, we use n-gram analysis for extracting meaningful features from the data.

An n-gram can be defined as a consecutive sequence of n characters [22]. It could also be described as a substring with the length n. For example, the string *ababc* contains unique 2-grams *ab*, *ba* and *bc*. The 2-gram *ab* appears twice, thus having frequency of 2. A list of tokens of text can be represented with a vector consisting of n-gram frequencies [22]. Feature vector describing this string would be $\mathbf{x}_{ababc} = [2,\ 1,\ 1]$. Similar feature vectors will form the whole feature matrix.

Here is an example of constructing the feature matrix using the n-gram analysis process with two words, *anomaly* and *analysis*. From these words we get the unique 2-grams `an`, `no`, `om`, `ma`, `al`, `ly`, `na`, `ys`, `si` and `is`. From this information we can construct a matrix with the n-gram frequencies.

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

The feature matrix $\mathbf{X}$ is constructed in a similar way from the request string n-gram frequencies. The occurrences of specific n-grams are summed for each log line. In practice, n-gram tables generated from real-life log data are very sparse, because most of the n-grams do not actually appear on all the log lines. The n-grams that do not appear in the data at all as well as the corresponding columns in the feature matrix can be excluded. This will reduce the number of dimensions (columns) in the matrix.

Apache log files are ascii-coded, which means that 256 different unique characters are possible. If we choose $n = 1$ for the n-gram analysis, we get a simple character distribution with 256 theoretical maximum dimensions. For this analysis, we choose $n = 2$, which means that the maximum number of dimensions is $256^2 = 65,536$. However, in a normal real-world situation the actual number is much lower. With $n = 3$ or higher, the dimensionality of the data is massively increased. Since the performance of many of the algorithms used depend on the size of the individual feature vector, high input dimensions would slow performance. With the chosen value $n = 2$, we get a good balance between contained information and the size of the feature matrix.

4. Dimensionality reduction

Dimensionality reduction methods try to map high-dimensional data to fewer dimensions while retaining the internal structure of the data. Usually this structure is defined by distances between the data points. Several mathematical approaches can be found [23], three of which are presented in the next three sections: random projection (RP), principal component analysis (PCA) and diffusion maps (DM). Each section presents the training step and the out-of-sample extension of new data points.

RP	Very fast, sometimes unstable, cannot reduce dimensions very much, preserves distance
PCA	Average speed, preserves variance, cannot handle non-linear data
DM	Slowest, can deal with non-linear data, can facilitate spectral clustering

Table 1: Different dimensionality reduction methods in this paper.

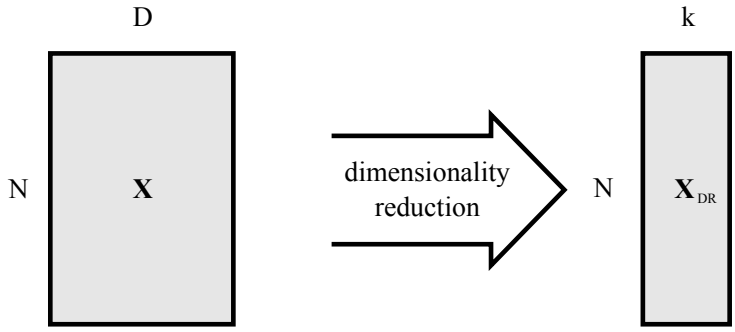

Figure 3: Dimensionality reduction reduces the number of variables describing the data points while retaining most of the information.

These three algorithms perform the same reduction but from different theoretical viewpoints. Even though they pursue the same goal, their performances may differ. Table 1 briefly summarizes the differences between the methods. Furthermore, various datasets exhibit behaviors that are appropriate for only some of the methods.

The basic idea behind dimensionality reduction is expressed in Figure 3. The data points on the rows of $\mathbf{X}$ are described by D dimensions, or features on the columns. The dimensionality reduction algorithm reduces the number of dimensions to k while retaining sufficient information in the new dimensions. The most common information to preserve is the distance between different data points The most common information to preserve is the distance between different data points.

5. Random projection

5.1. Training

Random projection (RP) is a dimensionality reduction method based on Johnson–Lindenstrauss lemma [24], for which a proof is available in the literature [25]. It states that if points in a vector space are projected onto a randomly selected subspace with high-enough dimensions, the distances between these points are preserved approximately provided that the vectors have unit lengths. In other words, the goal is to use a randomly generated matrix to lower the number of dimensions in the data.

Let us assume that we have the original data matrix $\mathbf{X}$ with N data points and D dimensions. The number of dimensions in the low-dimensional subspace is k so that $k \ll D$. The randomly generated matrix is $\mathbf{R}^{k \times D}$. The matrix containing data points projected onto the low-dimensional subspace is obtained with the following multiplication [26]:

$$\mathbf{X}_{\mathrm{RP}}^{k \times N} = \mathbf{R}^{k \times D} \mathbf{X}^{D \times N}.$$

Random projection is not actually a projection because the matrix $\mathbf{R}$ is not strictly orthogonal [26]. In addition, orthogonalization of a matrix can be computationally expensive. However, we can use the result by Hecht-Nielsen [27]: *"There exists a much larger number of almost orthogonal than orthogonal directions in a high-dimensional space"*. Therefore, RP is close to a projection and can be used for practical applications.

We can use a very simple probability distribution for choosing the elements r_{ij} or random matrix $\mathbf{R}$. Using the following distribution, we get *sparse random projection* as originally proposed by Achlioptas [28]:

$$r_{ij} = \sqrt{3} \times \begin{cases} +1 & \text{with probability} \frac{1}{6} \\ 0 & \text{with probability} \frac{2}{3} \\ -1 & \text{with probability} \frac{1}{6} \end{cases}.$$

Alternatively, we can use a more general distribution [29]:

$$r_{ij} = \sqrt{s} \times \begin{cases} +1 & \text{with probability} \frac{1}{2s} \\ 0 & \text{with probability} 1 - \frac{1}{s} \\ -1 & \text{with probability} \frac{1}{2s} \end{cases}.$$

If we choose $s = 3$, we get the original distribution for Achlioptas' sparse random projection. It is also possible to choose $s \gg 3$, which leads to *very sparse random projections* [29].

In this case, given the original data matrix $i\mathbf{X} \in \mathbb{R}^{N \times D}$ and random matrix $\mathbf{R} \in \mathbb{R}^{D \times k}$, we can obtain the randomly projected matrix by [29]:

$$\mathbf{X}_{RP} = \frac{1}{\sqrt{k}}\mathbf{X}\mathbf{R} \in \mathbb{R}^{N \times k}, k \ll \min(N, D).$$

The main advantage of random projection methodology is its speed and computational efficiency. Even though the method is very fast, it is accurate enough not to create too much distortion in the data [26]. This makes it usable for applications where computationally expensive algorithms are not feasible.

It is important to note that in this case "training" simply means the generation of the random matrix. The matrix $\mathbf{R}$ does not depend on the used training data. However, random projection contains a parameter ϵ so that $0 < \epsilon < 1$. Using this, the minimum number of dimensions for the projection can be calculated for the chosen value of ϵ using the Johnson–Lindenstrauss lemma. This can be considered a training stage, since we are making sure that the dimensionality is not reduced too much for the given dataset, so that the distances are still approximately preserved.

5.2. Out-of-sample extension

In this context, out-of-sample extension means projecting any new data points to the same subspace as all the points in the data matrix $\mathbf{X}$. Since generation of the random matrix $\mathbf{R}$ does not depend on data points in the matrix $\mathbf{X}$, projecting any new data points does not require any special steps other than matrix multiplication. If we get a new preprocessed data vector $\mathbf{y}_i$, projecting it onto the low-dimensional subspace can simply be done performing the following:

$$\mathbf{y}_{RP} = \mathbf{y}_i\mathbf{R}$$

This will give us the original data vector projected to the subspace. As can be seen, random projection facilitates out-of-sample extension, making it feasible for online anomaly detection systems due to it's simplicity and speed.

6. Principal component analysis

6.1. Training

Principal component analysis is probably the most popular dimensionality reduction technique. The goal is to represent the information included in the original correlated variables using a smaller number of independent variable called *principal components*. The principal components are linear combinations of the original variables [30].

To perform PCA for the original data matrix $\mathbf{X}$, the matrix is first centered to form the matrix $\mathbf{X}_c$. Covariance matrix $\mathbf{C}$ is then calculated from the centered data. From this, we can use the following decomposition for real-valued matrices:

$$\mathbf{C} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\intercal}.$$

Now we can obtain the eigenvectors in the matrix $\mathbf{U}$. To map the data points into the low-dimensional subspace we only need to perform the multiplication

$$\mathbf{X}_{\mathrm{PCA}} = \mathbf{X}\mathbf{U}.$$

The new principal components are in the direction of most variance in the data and thus represent the most differentiating combination of features [23, 30, 31]. Normally the principal components containing 95% of the variance are selected and the rest dropped out because they do not include much information. In an optimal situation, the first few components are enough.

The principal components are linear combinations, and PCA can only find linear dependencies in the data. It has initial assumptions that restrict its use for latent variable separation and nonlinear dimensionality reduction [23].

The calculation of covariance matrix $\mathbf{C}$ and the subsequent calculation of $\mathbf{U}$ can be considered the training stage, since both of these need some original data to be calculated. If we want to retrain the algorithm with new data, these have to be calculated again unless some more complicated update algorithm is used. It might also be sufficient to recalculate PCA periodically.

6.2. Out-of-sample extension

When the system gets new data points, using the same projection as used in the training stage is very simple, as it requires only a multiplication operation.

Given a new data point $\mathbf{y}_i$, we can project it into the same subspace as other points by doing the following:

$$\mathbf{y}_{PCA} = \mathbf{y}_i \mathbf{U}$$

With this multiplication we get the new projected data point $\mathbf{y}_P CA$.

7. Diffusion maps

7.1. Training

Diffusion map is a function from multi-dimensional space to a space with lower dimensions while the information content is only slightly distorted. It can be described using the taxonomy of dimensionality reduction methods as a nonlinear geometric method that preserves the diffusion distance as Euclidean distance in the lower dimensions [23, 32]. The underlying assumption in such manifold learning methods is that the data is situated on a manifold that is embedded to the ambient space [33].

Recall that the measurements $\mathbf{x}_i \in \mathbb{R}^D, i = 1 \dots N$ lie on a D-dimensional space, where N is the number of measurements and D is the number of measured variables. The measurements should be normalized in order to make the variables comparable. One way of doing this is simply taking the logarithm of each value in the data matrix.

In the diffusion map method, at first, the pairwise distances between the data points are calculated. These distances are exaggerated using a kernel function. Here, the Gaussian kernel is used with Euclidean distance measure:

$$\mathbf{W}_{ij} = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\epsilon}\right).$$

The degree of each point can be calculated from $\mathbf{W}$ by summing the weights that connect them to the other points. This means that the kernel matrix rows are summed: $\mathbf{D}_{ii} = \sum_{j=1}^{N} \mathbf{W}_{ij}$. The rows of $\mathbf{W}$ are normalized by the row sums: $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. Matrix $\mathbf{P}$ can now be understood as containing the transition probabilities between the data points. Symmetric matrix $\tilde{\mathbf{P}}\mathbf{D}^{\frac{1}{2}}\mathbf{P}\mathbf{D}^{-\frac{1}{2}}$ is simplified by substituting the original $\mathbf{P}$ with its definition:

$$\tilde{\mathbf{P}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}.$$

The decomposition of this real-valued normal matrix is expressed as $\tilde{\mathbf{P}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$. Singular value decomposition (SVD) performs the operation,

resulting in matrix $\mathbf{U}$ that contains eigenvectors on its columns and the diagonal of $\mathbf{\Lambda}$ contains the corresponding eigenvalues of $\tilde{\mathbf{P}}$. However, the real interest is in the eigenvectors of the transition matrix $\mathbf{P}$. Those eigenvalues of the two matrices are the same, but the eigenvectors are obtained by calculating the right eigenvectors:

$$\mathbf{V} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}.$$

The low-dimensional coordinates can now be formed by multiplying each eigenvector column with the corresponding eigenvalue. The resulting matrix contains N rows, each corresponding to the data points, and k columns, each representing the new dimensions.

$$\mathbf{X}_{\mathrm{DM}} = \mathbf{V}\mathbf{\Lambda}$$

Only some of these coordinates are needed to represent the data to a certain degree of error [34]. The data can be reconstructed using only some of the eigenpairs while the error stays small enough. Due to the graph theoretical calculations, the first eigenvector is constant, so it is not used.

7.2. Nyström extension

Nyström extension takes new points data points and extends them to the low-dimensional space mapped earlier by diffusion map. The goal is to interpolate the coordinates of unknown points based on the coordinate mapping of the training data. With this kind of projection, the new points can be compared with the training dataset. Many dimensionality reduction methods can use the general Nyström extension framework for out-of-sample extension [35, 36, 37]. The same features are used as the ones used during the training, with the same normalization.

Let us assume that a new data point $\mathbf{y}_j \in \mathbb{R}^D$ is extended. The distances between the data point and the training points are collected to $\bar{W}$, which is defined as

$$\bar{\mathbf{W}}_{ij} = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{y}_j||^2}{\epsilon}\right).$$

Similarly to the training, diagonal matrix of $\bar{\mathbf{D}}_{ii} = \sum_{i=1}^{N} \bar{\mathbf{W}}_{ij}$ contains the column sums of $\bar{\mathbf{W}}$. The transition probabilities are then calculated with

$$\mathbf{B} = \bar{\mathbf{W}}\bar{\mathbf{D}}^{-1}.$$

Now, the eigenvectors in the columns of $\bar{\mathbf{V}}$ can be found:

$$\bar{\mathbf{V}} = \mathbf{B}^\intercal \mathbf{V} \mathbf{\Lambda}^{-1}$$

The eigenvalues $\mathbf{\Lambda}$ are the same as in the training. The low-dimensional coordinates for the new points can be found with $\bar{\mathbf{Y}} = \bar{\mathbf{V}}\mathbf{\Lambda}$, and the last two steps are combined as

$$\bar{\mathbf{Y}}_{\mathrm{DM}} = \mathbf{B}^\intercal \mathbf{V}.$$

Matrix $\bar{\mathbf{Y}}_{\mathrm{DM}}$ now contains the extended coordinate approximations in its columns for the new points $\mathbf{y}_j$.

8. Anomaly detection

Anomaly detection is performed using statistical analysis. We assume that the data follows a Gaussian distribution. We calculate the mean point from the training data. The average distance of training points from the mean point is μ_{train}, which is used as an estimate for μ_{test} for the testing data. We use a simple anomaly detection method, where any data points with a distance greater than $n\sigma$ from μ_{test} will be classified as an anomaly with the anomaly indicator function

$$g(y) = \begin{cases} 1, & \text{if } n\sigma < \|y - \mu_{\mathrm{test}}\|. \\ 0, & \text{otherwise.} \end{cases}$$

Here y is the low-dimensional representation of a data point and σ is the standard deviation of the training data points from the mean. The choice of n is not trivial, but a common option is $n = 3$ [33]. In practice, with large or non-Gaussian datasets a different threshold should be selected.

9. Results

The proposed framework is tested using several different datasets received from companies. The logs contain simulated test data as well as real-life network logs from actual production web servers. All of the files include only HTTP server access logs. This is feasible because most or all of the traffic will be encrypted in the near future, making it very difficult to analyze the payload. The header information is still going to be available. The format and contained information is introduced in Section 3.1. It is important to

note that we did not get access to payloads of the packets, just the header information contained in normal HTTP log files.

9.1. Simulated test data

For the first initial experiment, we received two datasets with simulated test traffic and manually injected intrusion attempts. The intrusions fall under two general categories. Firstly, some attacks try to access vital files in the server, e.g., the file `/etc/passwd` on a Linux server. Secondly, some cross-site scripting (XSS) attacks have been injected. These attacks attempt to execute malicious foreign scripts when the user visits a web page. These are especially difficult, since these attacks do not contain many uncommon or encoded characters, and are difficult to find with access logs alone. In addition, all of the analysis phases are done using a normal laptop computer, and the execution times are presented so the efficiency of different methodologies can be compared. In an actual scenario with finished software, the analysis can be performed on a specialized server with multiple cores, making the analysis dozens of times faster. More accurate analysis of the execution times and scalability can be found below in Section 9.2.

The first and smaller log contains only a few intrusions, and mostly consists of normal traffic. It contains 2,693 lines. RP methodology analyzes the data in 2.4 seconds, finds two actual intrusions and gives one false alarm. PCA-based analysis finds the same two intrusions with no false positives, and takes 11.2 seconds. Using DM, we discover the same two attacks plus one extra intrusion which was not found using the other methods. This analysis takes considerably longer, 196.6 seconds. A low-dimensional representation of the data using DM can be seen from Figure 4. In this case RP is fast and efficient, PCA is quite fast and more precise, and DM is the most accurate but also the slowest. An example plot of the anomaly levels is presented in Figure 5.

The second simulated log contains 5,369 lines and contains the two different attack types mentioned above. RP analysis takes 4.7 seconds, and it finds 62 attacks trying to access important server files. PCA takes 25 seconds, and only finds 51 of these attack attempts. Both of these methods give zero false positives, but completely fail to detect the XSS attacks. The actual contents of the scripts are not present in the log files, making them difficult to be detected. However, DM analysis finds 141 lines of XSS attacks , as well as 47 of the other types of attacks. The execution time is 285 seconds in this experiment. RP finds the first type of attacks better and faster than others,

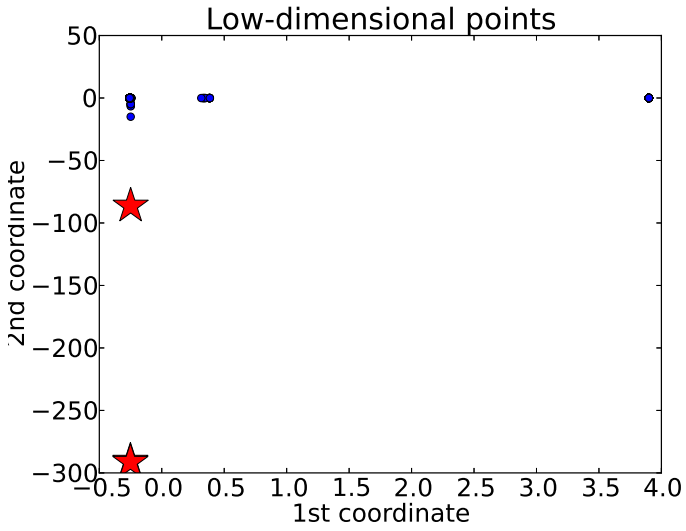

Figure 4: Low-dimensional points and anomalies using DM. Red stars are the anomalies, while blue dots are normal HTTP requests.

but once again DM is more accurate at finding difficult intrusion attempts that cannot be detected using the other methodologies. Figure 6 shows the normal points as well as the two main attack types in a low-dimensional visualization created using DM.

While these manually created logs are useful for initial testing, they are much too small for more practical testing. Therefore, more testing data is analyzed in the next section.

9.2. Speed and scalability tests using real-world log data

This dataset contains HTTP queries to a real web server. In this log there is much more traffic. Various subsets of the data are used to test the speed and especially the scalability of the three different methodologies to compare the efficiency on larger datasets. The variability of this kind of real data is higher than in the simulated case. For more accurate and realistic results, it would be essential to have better log data. However, the size of the data is big enough to test the scalability aspects. We tested up to 300,000 log lines since that provided enough evidence in terms of linear scalability. All of the methods find actual intrusions attempts from the data, but it is impossible to say how many potential intrusions are left undetected, since the test is completely unsupervised.

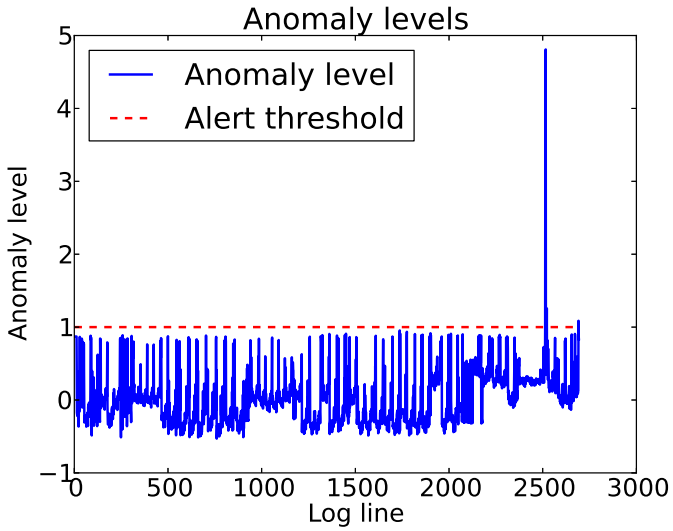

Figure 5: Anomaly levels for test3 log file with RP.

Figure 7 shows the speed and scalability using different sizes of data. There are two main things to note here. Firstly, all of the methods scale linearly. This is because the training phase is done using a limited number of log lines, and analyzing new streaming data is a linear operation. Secondly, the differences in scaling between the methods are massive. As expected, RP can analyzed much more data than other methods. The processing time increases rapidly when using DM. It seems that RP works well when analyzing bigger datasets, and DM can be used for more accurate analysis on smaller sets of data, since the experiments in Section 9.1 showed that DM can find some intrusions more accurately.

Another real-world dataset is used for testing RP methodology. Figure 8 shows the computation time against log file size. The test was run on two Intel® Core™ i5-2520M CPU @ 2.50 GHz cores using hyper-threading. As expected, the time taken by online anomaly classification is linearly dependent on the size of data with this data as well. The 300 MB dataset corresponds to the amount of daily traffic of a small web service. It happens acceptably fast even on a very low-powered computer used in this experiment. Using more cores will significantly speed up the processing. The most important thing to note here is the linear scaling of the system.

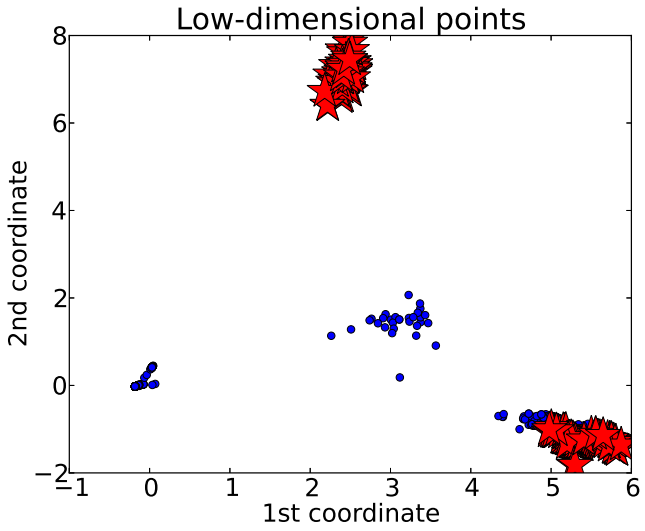

Figure 6: Low-dimensional presentation of the data using DM. Red stars are anomalies, while blue dots are normal HTTP requests.

10. Conclusion

Web log analysis can be done with anomaly detection. This paper presents results from three methods that can be used for dimensionality reduction before anomaly detection: random projection, principal component analysis and diffusion maps. These results show that web attacks can be captured using this type of framework.

The results suggest that an ensemble system could be built upon the methodologies described. Based on the experimental results, we propose that RP and DM should be used together. RP methodology is efficient for daily analysis of huge amounts of traffic, while DM produces better visualizations and more accurate analysis of smaller amounts of data when needed. PCA falls in between of the other methods, but does not seem to offer any major advantages in our experiments.

These results are relevant to new intrusion detection services for web servers. Moreover, analyzing any log files produced by various applications should be easier using dimensionality reduction. The usefulness of anomaly detection in any text mining task is also obvious. The results show that new data point extension happens in linear time. The analysis can be performed in sufficient time on huge volumes of data.

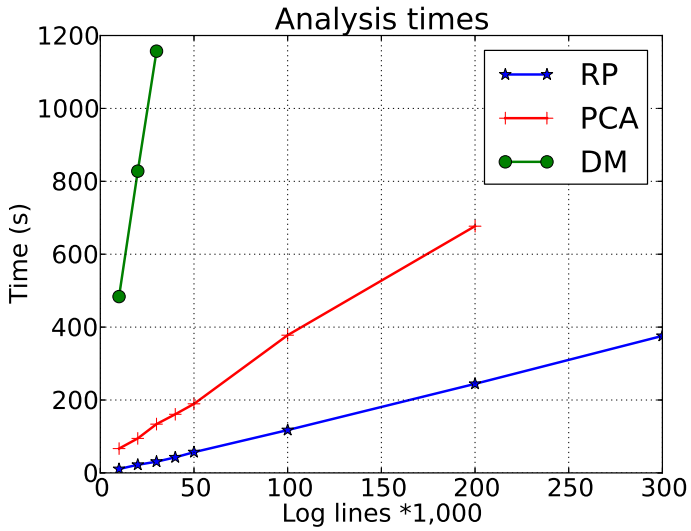

Figure 7: Computational times for RP, PCA and DM.

One important question is sampling the data for training. The data sampling should strive to represent the variability in the dataset but with a limited number of samples. It is not trivial to choose the right training data size. In addition, the selection of anomaly threshold can be challenging. More robust automatic parameter selection must be developed.

For future research, larger volumes of data must be analyzed to ensure that the scaling is efficient. It is possible that the system will be throttled by memory or I/O instead of CPU, which will create new challenges. In addition, it would be useful to try different formats of data to test the feasibility of the framework more generally. Furthermore, some optimization could be made to the implementation to ensure a better performance in a realistic network application.

Acknowledgment

The authors would like to thank Pardco Group Oy and Second Nature Security Oy for co-operation.

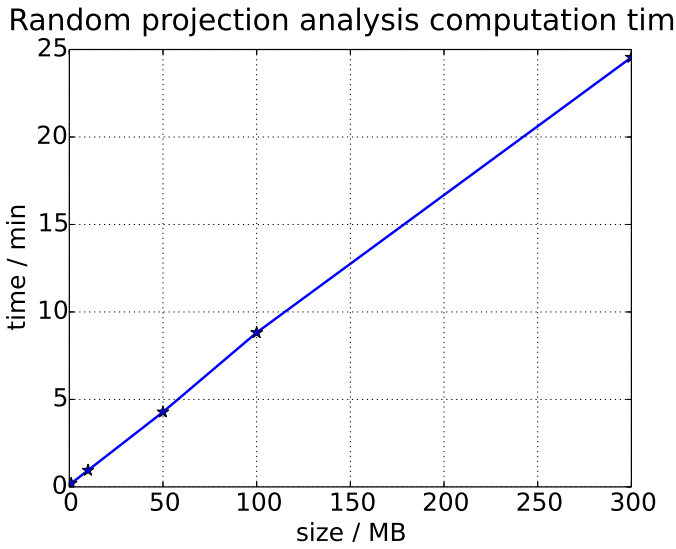

Figure 8: Random projection computation time with two Intel® Core™ i5-2520M CPU @ 2.50 GHz cores using hyper-threading.

References

[1] K. Scarfone, P. Mell, Guide to intrusion detection and prevention systems (IDPS), NIST Special Publication 800 (2007) 94.

[2] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, K.-Y. Tung, Intrusion detection system: A comprehensive review, Journal of Network and Computer Applications 36 (2013) 16–24.

[3] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: Security and Privacy (SP), 2010 IEEE Symposium on, IEEE, pp. 305–316.

[4] P. Sangkatsanee, N. Wattanapongsakorn, C. Charnsripinyo, Practical real-time intrusion detection using machine learning approaches, Computer Communications 34 (2011) 2227–2235.

[5] D. E. Denning, An intrusion-detection model, Software Engineering, IEEE Transactions on (1987) 222–232.

[6] A. Patcha, J.-M. Park, An overview of anomaly detection techniques:

Existing solutions and latest technological trends, Computer Networks 51 (2007) 3448–3470.

[7] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, J. Ucles, HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification, in: Proc. IEEE Workshop on Information Assurance and Security (2001), pp. 85–90.

[8] M. Amini, R. Jalili, H. R. Shahriari, RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks, Computers & Security 25 (2006) 459–468.

[9] M. Govindarajan, R. Chandrasekaran, Intrusion detection using neural based hybrid classification methods, Computer networks 55 (2011) 1662–1671.

[10] K. Labib, R. Vemuri, NSOM: A real-time network-based intrusion detection system using self-organizing maps, Networks and Security (2002) 1–6.

[11] W. Hu, Y. Liao, V. R. Vemuri, Robust anomaly detection using support vector machines, in: Proc. International Conference on Machine Learning (2003), pp. 592–597.

[12] J. M. Estévez-Tapiador, P. Garcıa-Teodoro, J. E. Dıaz-Verdejo, Measuring normality in http traffic for anomaly-based intrusion detection, Computer Networks 45 (2004) 175–193.

[13] M. Ahmadi Livani, M. Abadi, A pca-based distributed approach for intrusion detection in wireless sensor networks, in: Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on, IEEE, pp. 55–60.

[14] Y.-J. Lee, Y.-R. Yeh, Y.-C. F. Wang, Anomaly detection via online oversampling principal component analysis, Knowledge and Data Engineering, IEEE Transactions on 25 (2013) 1460–1470.

[15] T. Sipola, A. Juvonen, J. Lehtonen, Anomaly detection from network logs using diffusion maps, in: L. Iliadis, C. Jayne (Eds.), Engineering

Applications of Neural Networks, volume 363 of *IFIP Advances in Information and Communication Technology*, Springer Boston, 2011, pp. 172–181.

[16] T. Sipola, A. Juvonen, J. Lehtonen, Dimensionality reduction framework for detecting anomalies from network logs, Engineering Intelligent Systems 20 (2012) 87–97.

[17] A. Juvonen, T. Sipola, Adaptive framework for network traffic classification using dimensionality reduction and clustering, in: IV International Congress on Ultra Modern Telecommunications and Control Systems 2012 (ICUMT 2012), St. Petersburg, Russia, pp. 274–279.

[18] A. Juvonen, T. Sipola, Combining conjunctive rule extraction with diffusion maps for network intrusion detection, in: The Eighteenth IEEE Symposium on Computers and Communications (ISCC 2013), Split, Croatia, pp. 411–416.

[19] A. Juvonen, T. Hämäläinen, An efficient network log anomaly detection system using random projection dimensionality reduction, in: New Technologies, Mobility and Security (NTMS), 2014 6th International Conference on, IEEE, Dubai, United Arab Emirates, 2014, pp. 1–5.

[20] K. L. Ingham, H. Inoue, Comparing anomaly detection techniques for HTTP, in: Recent Advances in Intrusion Detection (2007), Springer, pp. 42–62.

[21] Log files - Apache HTTP server, 2014.

[22] M. Damashek, Gauging similarity with n-grams: Language-independent categorization of text, Science 267 (1995) 843.

[23] J. Lee, M. Verleysen, Nonlinear dimensionality reduction, Springer Verlag, 2007.

[24] W. B. Johnson, J. Lindenstrauss, Extensions of lipschitz mappings into a hilbert space, Contemporary mathematics 26 (1984) 1.

[25] S. Dasgupta, A. Gupta, An elementary proof of a theorem of johnson and lindenstrauss, Random Structures & Algorithms 22 (2003) 60–65.

[26] E. Bingham, H. Mannila, Random projection in dimensionality reduction: applications to image and text data, in: C. Kruegel, R. Lippmann, A. Clark (Eds.), Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (2001), ACM, pp. 245–250.

[27] R. Hecht-Nielsen, Context vectors: general purpose approximate meaning representations self-organized from raw data, Computational intelligence: Imitating life (1994) 43–56.

[28] D. Achlioptas, Database-friendly random projections, in: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2001), ACM, pp. 274–281.

[29] P. Li, T. Hastie, K. Church, Very sparse random projections, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (2006), ACM, pp. 287–296.

[30] H. Abdi, L. Williams, Principal component analysis, Wiley Interdisciplinary Reviews: Computational Statistics 2 (2010) 433–459.

[31] J. Han, M. Kamber, Data mining: concepts and techniques, Morgan Kaufmann, 2006.

[32] L. J. P. van der Maaten, E. O. Postma, H. J. van Den Herik, Dimensionality reduction: A comparative review, Journal of Machine Learning Research 10 (2009) 1–41.

[33] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (2009) 15:1–15:58.

[34] R. R. Coifman, S. Lafon, Diffusion maps, Applied and Computational Harmonic Analysis 21 (2006) 5–30.

[35] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, M. Ouimet, Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering, Advances in neural information processing systems 16 (2004) 177–184.

[36] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the Nyström method, Pattern Analysis and Machine Intelligence, IEEE Transactions on 26 (2004) 214 –225.

[37] S. Belongie, C. Fowlkes, F. Chung, J. Malik, Spectral partitioning with indefinite kernels using the Nyström extension, in: A. Heyden, G. Sparr, M. Nielsen, P. Johansen (Eds.), Computer Vision ECCV 2002, volume 2352 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2002, pp. 531–542.