Classification and visualization of web attacks using HTTP headers and machine learning techniques

Nicolas R Enciso[1] and Jorge E Camargo[2]

Universidad Nacional de Colombia jecamargom@unal.edu.co
nricardoe@unal.edu.co
www.unsecurelab.org

Abstract. This paper presents a methodology to identify web attacks such as XSS, CRLF and SQL injection using a data set that contains normal and anomalous items. The proposed methodology uses dimensional reduction techniques for visualization (PCA, t-SNE) and machine learning algorithms (SVM, Naive Bayes, random forest, logistic regression) to perform classification of URLs contained in HTTP headers. Results show that visualization is useful to present a general overview of attacks and classification experiments show an accuracy of 83% to detect attacks.

Keywords: HTTP header · URL · feature engineering · machine learning classifier · web attacks

1 Introduction

Web attacks are a growing problem worldwide, taking at the center of discussion security of information as a crucial part of the modern way of working from governments to private enterprises and industries [1] [2]. With the increasing of new threats that are being discovered, it is necessary to build new tools that allow to automatically detect attacks at the same time that threats appear.

In order to address this challenging problem, machine learning algorithms are a powerful tool that offer the capacity to extract patterns from data, as the case of intrusion detection systems (IDS), which allow to get detection rates of anomalous behaviours or outlier cases in considerable high accuracy percentages on real time scenarios [3] [4]. The same has been investigated using other techniques such as big data [5].

Nevertheless, to get high accuracy rates it is indispensable to apply special previous treatment over data that will be used as input to the machine learning algorithms. Feature extraction is a key process, which defines success on the application of it. Besides the correct selection of characteristics from data, it is necessary to perform an analysis about the characteristics (features) that differentiate the target class from the other classes in the dataset. Additionally, when it is obtained a large number of extracted variables as features, the complexity of analysis raises (curse of dimensionality), which suggests to select a subset of such features to reduce complexity without lose of much information.

In the feature extraction process of URLs it is possible to detect web attacks such as CRLF (Carriage return line feed), XSS (Cross-site scripting), and SQL injection, even if they are used in the detection of phishing [6] [7], catching the main characteristics with an analysis of different parts that forms an URL. This kind of web attacks are very usual and take advantage of bad software building, wrong configurations and naive mistakes from junior software engineers. According to OWASP [8], on the top 10 most common attacks appears injections and XSS as main types of attacks around the Internet.

In spite of visualization in high dimensional data presents difficulties, there are a variety of techniques that are able to reduce the number of dimensions on the data without lose of much information, as the case of non negative matrix factorization (NMF), which has been used with success in the detection of outliers in sensors [9] and the detection of anomalies in smart cards [10], and PCA, which has been used with success in classification of anomalous cases in intrusion detection systems [11].

This paper presents a methodology to detect web attacks using visualization techniques and machine learning. The rest of the paper is organized as follows: Section 2 presents a brief description of web attacks; Section 3 discuss how dimensionality reduction techniques are used to visualize data; Section 4 presents the machine learning algorithms used to train a classification model; In Section 5 we present the proposed methodology to detect web attacks; Section 6 shows the obtained results; and Section 7 concludes the paper.

2 Web attacks

This section presents a description of attacks that we addressed in this paper to validate the proposed attack detection methodology.

2.1 Cross-site scripting

Basically this is an attack that consists in the injection of Javascript code or scripts into web applications. Typically scripts are injected into an HTML document, and take advantage of the fact that when a web browser reads an HTML document, it executes all the code, so, if there is Javascript code inside the HTML document, it will be executed compromising the target user device [16]. The unique condition to execute the hidden script inside the HTML is turning on the execution of Javascript on the web browser, feature that is set on as default configuration in most of the popular web browsers. This attack is in the top 10 OWASP most common attacks through the Internet [8].

There are several types of XSS attacks, according to the OWASP project and guidelines about security on Internet [17]. one of them is The Non Persistent or reflected XSS attack, which runs in the user browser each time the user runs the script, commonly injected at the end of the URL. The other one is known as Persistent or stored XSS, which exploits the interaction between the website and a data base. Discovering the bridge between the website and the data base,

the attacker stores permanently the malicious code inside the server side, so any user that visits the website will get the malicious code, spreading the attack all over around.

2.2 Carriage Return Line Feed (CRLF)

This is a vulnerability exploited by attackers to introduce a code injection. On example of the attack consists in the introduction of a new line via URL, putting the hexadecimal %0d and %0a codes in the URL string [18]. These characters indicate a new line in the log of the server. The attacker puts for instance HTTP splitting code or phishing links on the new line. This take advantage of the characters that determine the end of a line depending on the operating system: Carriage return with ASCII 13 (\r) and line feed with ASCII 10 (\n). Other example is known as ''response splitting" attacks, where CRLFs are injected into an application and included in the response. The extra CRLFs are interpreted by proxies, caches, and browsers as the end of a packet, causing malfunctioning.

2.3 SQL Injection

This is one of the most dangerous and popular along the history of web applications. This attack takes advantage of query languages used to access data bases. The main cause to be victim of an SQL injection attack is the absence of sanitization. Typically the attacker introduces in a text box especial characters and reserverd SQL words to be executed by the data base server [19]. Commonly, the attacker performs some tests to detect the types of user on the data base, if there is a direct connection with the data base, and others. To do that, it is used always true statements, taking advantage of the obvious use of the "WHERE" statement on the SQL query, making the query true and returning a full error text directly from the data base, giving to the attacker important information.

The attacks now consist on the modification of the data base, with the information got it from the previous step, and putting the queries on the string text that the web site performs from the text on the text box. This allows the attacker to delete information, changing user access, drop tables, getting sensitive information, destroying data bases, kidnapping of information and many others [20]. With all the attack understood, the principal way to find this kind of attack is detecting the use of reserve word of the query language on the parameter field on the URL, as the present work uses.

3 Dimensionality reduction for data Visualization

3.1 Principal Component Analysis

Principal component analysis (PCA) is a method that uses matrix operations through linear algebra and statistics to get a projection in a lower dimensional

space of the original data. This lower dimensional space allows to generate a trustful representation of the same data using the main components (2D or 3D representation). Such dimensions allow to plot data in a 2D/3D coordinates system highlighting the distribution of the data

In this paper we use PCA to visually see the the result of the perfomed analysis. Even though PCA could be used as a machine learning classifier, great results depends on the data itself and not for a learning method. This does not mean that the application of PCA as a classifier has not been performed with good results in the literature, but it needs of other methods to achieve good results [5].

The original data set is represented as a matrix with m rows and n columns as depicted in Figure 1.

$$\begin{matrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & ... & v_{1n} \end{bmatrix} \\ \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & ... & x_{1n} \\ x_{21} & x_{22} & x_{23} & x_{24} & ... & x_{2n} \\ x_{31} & x_{32} & x_{33} & x_{34} & ... & x_{3n} \\ ... & ... & ... & ... & ... & ... \\ x_{m1} & x_{m2} & x_{m3} & x_{m4} & ... & x_{mn} \end{bmatrix} \end{matrix} \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ ... \\ c_{m1} \end{bmatrix}$$

Fig. 1. Vector ***vi*** represents the variables of each column, the vector of ***ci*** represents the labels of the corresponding classes of each row (sample) on the data set.

On this resulting matrix, is applied a normalization. The normalization that was performed at the present work, was applied as it follows:

$$Z = \left(\frac{X - U}{S}\right) \tag{1}$$

Here is removed the mean and scaling to the unit variance. On the equation 1, Z is the standard score, X the sample numeric value, U the mean and S the standard deviation. With this procedure the data is centered and scaled to avoid miss understood point on the resulting graphic. In the next equation, X_std represents the standarized matrix, and X the original matrix.

$$X\_std = Standarize\left(X_{m*n}\right) \tag{2}$$

With the resulting matrix, is calculated the corresponding co variance matrix, getting a square matrix with n by n dimensions. This matrix represents the differences between each pair of variables, and the resulting diagonal contains the variance of each variable. This matrix contributes on the calculation of estimators of variance. In the next equation, Cov_X_std represents the covariance matrix from the standarized matrix.

$$Cov\_X\_std = Covariance\left(X\_std_{m*n}\right) \tag{3}$$

With the co-variance matrix, is calculated the eigenvalues matrix with dimension 1 by n and their corresponding eigenvectors matrix with dimensions n by n. This corresponds to the variance of each variable and their corresponding quantity of variance, in other words, the eigenvalues and eigenvectors represents how much is different the data. From a linear algebra point of view, the eigenvectors represents vectors that maintain their direction even when it is applied linear transformations and the eigenvalues the quantity of change, that explains why PCA maintains the essence of the original data, due to eigenvectors and eigenvalues keeps the information of the data.

From all the eigenvectors and their corresponding eigenvalues, it is sorted in decreasing order, from the eigenvalues with the highest value and their corresponding eigenvector, to the lower value of eigenvalue with their corresponding eigenvector. Each eigenvector is a principal component and their eigenvalue represents the quantity of "information" that was kept from the original data. With that in mind, is selected the first two principal components, each of them with dimensions n by 1, representing the two dimensions in which the data will be graphically showed. Also, it is calculated the explained variance and the individual variance of each principal component, this is performed to get an idea of how much "information" it is retained from the original data. A bigger summation between the variances of each principal component means more information explained (see Figure 2).

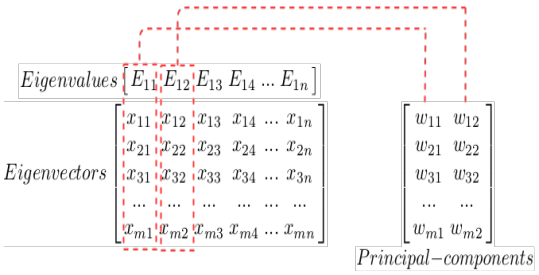

Fig. 2. The principal components are chosen by selecting the eigenvalues in decreasing order and its corresponding eigenvectors of W (matrix with principal components). We selected the 2 main components. .

With the resulting principal components matrix with dimensions n by 2, it is performed a matrix multiplication between the original data with dimensions n by m and the principal components matrix. At the following equation, Y is the resulting new data representation matrix in two dimensions.

$$Y = \left(X\_std_{m*n}\right) * \left(W_{n*2}\right) \tag{4}$$

The resulting matrix Y represents the new data representation in a lower dimensional space. On the used methodology, the labels that represents the class

of each row or sample, is added as a new columns on the end of the resulting matrix. This is done because of a way to put each sample correctly with their corresponding class in the graphic image as a result (see Figure 3).

$$\begin{bmatrix} Y_{11} & Y_{12} & c_{13} \\ Y_{21} & Y_{22} & c_{23} \\ Y_{31} & Y_{32} & c_{33} \\ ... & ... & ... \\ Y_{m1} & Y_{m2} & c_{m3} \end{bmatrix}$$

Fig. 3. ***Yi*** elements represent the elements from the Y matrix of the resulting principal components that were selected, and the ***ci*** elements are the corresponding class labels of each row (sample) on the original data set that were removed on the first step of the methodology.

3.2 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a method that uses the student's t distribution to reduce a data set from a high dimensional space to a low dimensional space that retains the majority of the original information. It is a dimensionality reduction technique that preserves the local structure from the original data as well as learns the parametric mapping of which the data comes from. The method is described by the original author [15] as an unsupervised method that learns parametric mapping between the high dimensional space, to a low dimensional space.

The algorithm measures the distance between a point and each one of the other points in a score named "unscaled similarity". This measure is putted on a matrix, with the diagonal the measure of 0, the distance of the point with itself. This process is called the determination of the similarity of a point. Later, the point is putted on the center of the Student's distribution or t-distribution. Each of the other points are putted on the same curve of the t-distribution, and the distance between the point and the curve is the unscaled similarity. This process is made for all the points measuring the unscaled similarity with respect of the point on the center of the curve of the t-distribution. The next step is to scale the unscaled similarities so that they add up to 1. This is performed to have a curve width that depends on the density of points, avoiding very wide curves, and get an standardized measure all across the points.

Finally the points are putted randomly on a lower dimensional space, to later move them according to the unscaled similarity, with the idea that high similarity between points attracts and low similarity repels, so as the output, there is get a matrix with a diagonal of 2 or 0 depending of the order, that shows perfect similarity, so on the graphic, it is get a lower dimensional space trustful

with respect of the original data. The advantages of this method over PCA, are that it doesn't uses euclidean distance, instead, is a non-linear dimensionality reduction, that has as main idea, that the dimensionality of many data sets is only artificially high.

4 Machine learning classifiers

4.1 Naive Bayes classifier

It is based on a probability theory called the Bayes Theorem, that express the conditional probability of a random event, knowing previous probabilities. The key idea behind the method is the following equation:

$$P\big(A|B\big) = \frac{P\big(B|A\big)P\big(A\big)}{P\big(B\big)} \tag{5}$$

It shows how often the event A happens given that B happens. The probabilities of A and B alone are easily calculated, so the only condition is the probability given the other event. In the case of the classifier, it calculates the probabilities of every factor, then the algorithm gives as output the event with the highest score, in other words, the classifier shows as output the event that scores the higher number, meaning the higher probability to be of that class.

The algorithm assumes that all the features are independent, that is the reason of the naive on the name. With the equation, the algorithm classifies giving to it the probabilities of each class, and later determines the similarity of a given example compared to each class, getting as an output the class with the highest probability.

4.2 Support Vector Machine SVM classifier

It is a type of supervised machine learning like the other algorithms. Given labeled training data, the method creates a hyperplane that categorizes the classes, dividing the classes with the hyperplane: in a two dimensional space is a line, on three a plane and so on, the classifier is one dimension less than the data set. There are several ways to build this hyperplanes, this different types of building the classification are called "kernels". On the present work, the kernels used are: lineal, sigmoid and gaussian. This makes the separation of classes using different techniques. The common thing, is that all the kernels wants to separate the data classes on a way that the gap between the hyperplane built and the points on the space the higher the better.

There is a problem with this kind of classifier, and is the time consuming that could it be. This is caused by the fact that the algorithm tries to find a mathematical function that optimizes the gap of separation, and with that build the hyperplane. For that reason, sometimes is necessary make a regularization, this is named as C parameter on sklearn, the library that was used on the

present work. The tuning of this parameters is choosing a smaller margin gap that divides the different types of hyperplanes, so the time that is needed to find the best function to build the hyperplane is lower. Another parameter is the gamma value, which defines how far points are be consider to build the hyperplane, a higher gamma speed up the algorithm, using fewer points, but losing precision.

4.3 Random Forest classifier

Random forest is a machine learning algorithm that uses the decision trees theory to build classifiers. The main idea is apply queries that makes narrower the number of possibilities in which the entry data could be classify. The tree is a binary tree, driving the answer path through True of False statements. This use of binary trees and simple True or False question makes this algorithm Log(n) on speed, taking advantage of the binary trees benefits. During the training, the random forest model build a structure that makes the best queries that ends with a correct answer. The name random forest came from the fact that this technique builds some decision trees randomly, creating a "forest" of decision trees.

The final result from a random forest, is the average of answers of all the decision trees, concluding with a unique output. The first step on the algorithm is build a bootstrap data set, that is choosing randomly samples of the original data set, and putting in on other data set, it is possible to put the same sample twice or more times. Later, is chosen two columns or features of the bootstrap data set, building the root of the decision tree with two possible paths, the value of the sample on that feature. Later, there is selected other two features, different ones, making the leafs of the tree, and so on until all the features are used. This process is repeated several times, building several trees. In the case of the present work, the number was 100. With all the trees built, the test sample is drive on all the decision trees, and the results are counted, so for voting, the final output is determined, taking as the answer the class with the majority of votes from thr decision trees.

4.4 Logistic regression classifier

This method is based on the statistical origin of the logistic function and the traditional linear regression used for making predictions through the fit of a function across the points on the space. The method fits the logistic function "S" shape, dividing two classes, on True or False, so this method only works for binary types of data and can work with continues or not variables. The logistic function plotted on the dimensional space of the data set, determines probabilities, so an entry data can be classified using the projection on the logistic function curve to determined the class that is most probably belongs to.
The logistic curve is fitted using the maximum likelihood, that depends on the training data, so the curve is graphed according to the given data, the likelihood numbers of each point is multiplied, getting the likelihood of all the curve. This

process is repeated until it is get the maximum value of the likelihood for the curve. The basic function used to start with the calculation of the likelihood is the following:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

That is the logistic basic function. On this x is the actual numerical value that is wanted to be transform. On the present work, the logistic regression classifier doesn't take any parameter, and the time used for the execution of this algorithm is low, because of the simple operations required to be executed.

5 Methodology

5.1 Data set

The data set that was used, comes from the Information Security Institute of CISC (Spanish Research National Council) from the Government of Spain [12]. It contains more than 60000 web request in form of HTTP headers generated automatically, containing the generated traffic targeted an e-commerce web application. The data set contains 36,000 normal requests and more than 25,000 anomalous requests, in which there are web attacks such as SQL injection, buffer overflow, files disclosure, CRLF injection, XSS and information gathering.

The data set was already used on other investigations, one of them on the building of intrusion detections systems (IDS) for WAFs (Web application frameworks), to detect web attacks [13]. Each HTTP header sample contains several fields as it is shown in the following:

Sample HTTP header
GET http://localhost:8080/tienda1/publico/anadir.jsp?id=2&nombre=Jam
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
*Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5*
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=B92A8B48B9008CD29F622A994E0F650D
Connection: close

From all the fields, the URL was the one selected to be the object of analysis and feature extraction, due to the fact that the other fields on the HTTP headers didn't represent essential information that differentiate the anomalous cases from

normal cases, also, because of continues repetitions of same data on the same fields, making possible bias feature extraction, consequently, wrong results on visualization and classification.

The data was given by two files in plain text format: one for anomalous cases and the other for normal cases. The anomalous file with 25,000 HTTP headers and the normal file with 35,000 cases. Both files on the HTTP headers have GET and POST types of HTTP requests, as a result, some of the HTTP headers have one more field, the ones of POST type which have the content type field extra.

5.2 Feature engineering extraction

The goal for this part of the work, was the extraction of essential information from the URLs, taking relevant characteristics that represents faithfully the differences between the anomalous cases and the normal cases.

Uniform Resource Locator Structure "URL" As it is specified on the standard of the RFC, specifically the RFC 1738 [14], is a string of characters that represents a resource that is available through the Internet, and has a specific order to show the location of a resource on a server online. A URL has the following structure:

$$< scheme >:< scheme - specific - port > \tag{7}$$

The scheme part, has the access mechanism or network protocol that uses the server that has the resource that wanted the user. The second part, the scheme specific and port are dependent of the server configuration and network DNS setup. A typical URL has as specific the domain name of the owner of the wep page, the port is the network port that uses the internet browser to get access to the web page, and the other parts, are the path on the server, or routes, that has the specific resources on the server.

The main focus was the characterization of the anomalous cases, that is, the features that an URL has when it is part of a web attack: SQL injection, CRLF injection or XSS. Some of the characteristics that were took into consideration, came from the techniques for detection of phishing from URLs in other paper work [7].

That was used as a result of the similarities between the URLs that are part of phishing and the URLs that are part of web attacks (SQL-CRLF injection, XSS). The web attacks have as main traits the randomness of the words in the path, the use of many characters, reserved words from specialized IT languages like SQL or from typical default configuration on network devices or computers and of course the main characteristics of each web attack that is inside the anomalous cases of the data set. With that in mind, the characteristics extracted from the URLs are illustrated in Figure 4.

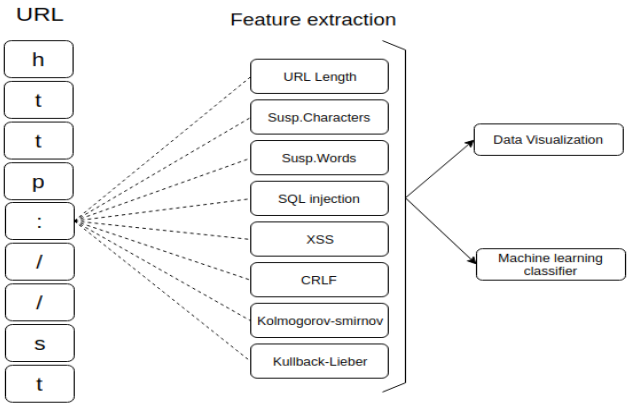

Fig. 4. Representation of the feature extraction performed on the URL given by each HTTP header. These 8 features are used as input to visualization and classification algorithms. .

URL length This takes the URL's length. A long URL increases the probability that the URL comes from an attack, this is because the attackers typically wants to confuses the user.

Suspicious characters A common characteristic on an URL that is part of a web attack, is the use of some characters, that confuses the user, who think that the URL is a trustful site when actually, this has minor differences from the original web site, using characters like "@", "#", "-" and so on.

Suspicious words Into the web attacks, it is typical sign the use of reserved words commonly used on configuration of servers, network devices and login functions on web sites. Here some examples of the words: "confirm", "account", "secure", "webscr", "login", "admin", "password" etc.

SQL injection words According to OWASP and the explanation given before about the SQL injection attack, the use of reserved words that comes from the SQL language is used on this kind of attack. For this reason, words like "SELECT", "WHERE", "FROM", "DROP", "TABLE", "LIKE", are taking into consideration, making a count of each of this words on each URL.

XSS attack words Following the guideline of OWASP about this kind of web attack, a cross site scripting injection uses scripts into the web browser to send malocious scripts. The typical way to do this, are the use of alerts and parameters into the HTML code. As a result, here is extracted words like: "alert", "script" and "param".

CRLF injection attack words Theoretically this is a vulnerability that is exploited by the attackers. In this is used two special characters, which confuses the end line signal on the ASCII end line character. The first character is the "%", which makes the rest of the string on the URL as a ignored part for the HTTP protocol, for that case, is typical case the use of trustful domain names after the %, the user thinks that it is the original web site, but actually is the ignored part for the browser. The second one is the "0". This is used like the %. For that case, the "%" and "0" characters are capture as a feature on the URL.

Kolmogorov-Smirnov test score This is a special statistic score, which wants to measure, through the non-parametric comparison of a list of values, determines a score if it comes from a determined probabilistic distribution. In other words, this want to measure how much a determined list of values are similar to pretend that all of it comes from the same distribution. At the present work, only the letters from the URL are compared to the distribution of characters on the Spanish language, because of the data set comes from Spain and uses Spanish as language, resulting on a score that could be a decimal number between 0 and 1, with 1 the score for the most similar to the language, meaning that the URL has a "logical" structure, with a possible lower "machine origin" or "random generated" URL that is commonly used by the attackers.

Kullback-Leibler divergence This is a measure that compares two distributions, and defines if this two are similar. This metric is very similar to the cross entropy, but in this case, it is measure how much similar is the distribution A and the distribution B. For this work, it is compared the distribution of letter on the URL, with the distribution of letter on the Spanish language, giving the score of how much the URL is similar to the Spanish, so the URL has a logical language usage, meaning that the URL has a user friendly path, which could be took as a sign of trustful web sites.

Resulting data set With the eight characteristic extracted, the data is putted on a new file with CSV format, because of the easy way to manage this kind of files with the Pandas Python library. In each HTTP header is so extracted the URL field, and later with the URL, the eight values captured, giving to the CSV file a vector of numbers, and a final one for the class it belongs to: 1 for anomalous and 0 for normal. On the following it is presented an example of the resulting sample on the CSV data set of one of the URLs:

284,16,5,0,0,4,0.115384615385,0.0334788954498,1

Each value on the sample corresponds to a feature as it follows:

URL length: 284
Suspicious characters: 16

Suspicious words: 5
SQL injection words: 0
XSS attack words: 0
CRLF words: 4
Kolmogorov-Smirnov score: 0.115384615385
Kullback-Liebler divergence: 0.0334788954498,1

5.3 Hardware tools

On the hardware, the work took place on two devices: the first one a laptop with an AMD A10 quad core processor @ 2.5Ghz, with 8GB of RAM memory, a Radeon HD8670M 2GB GPU, and the second one a desktop computer, with an AMD FX 8350 octa core processor @ 4Ghz, with 8GB of RAM memory and a NVIDIA GT840 2GB GPU.

The distribution of work was taking the laptop as the main tool for development and the desktop computer for execution of the software developed for the present work. The limitation of more computational power was presented by the time that was required to execute the software, taking several hours to achieve the results, principally on the machine learning classifiers, showing the necessity to have more hardware resources to get less time developing machine learning algorithms, specially on the support vector machine classifiers.

5.4 Software tools

We conducted the experiments in a machine with Linux 16.04 LTS for the laptop and 18.04 LTS for the desktop computer. The programming language used was Python version 2.7. The choice was Python due to the easy integration between libraries, simple structure and the possibility to use several machine learning and mathematical frameworks. We used the implementation of machine learning algorithms provided by Sklearn, pandas, matplotlib and the native libraries that comes with Python right out the box.

6 Results

6.1 Data visualization

The data set was passed through the two visualization techniques. All the data set was visualized in a 2D coordinates system. For the PCA technique, the algorithm was developed by us using the math library and pandas for matrix manipulation. In the case of t-SNE it was used the manifold library of scikit-learn. On both cases, the original data set contains 8 variables in 8 columns, with more than 60,000 rows, each one a sample on the data set, getting as the output on both algorithms, a two dimensional graph with two variables, reducing the original dimensionality. Table 1 shows the relation between each extracted features and the two principal components.

PCA results

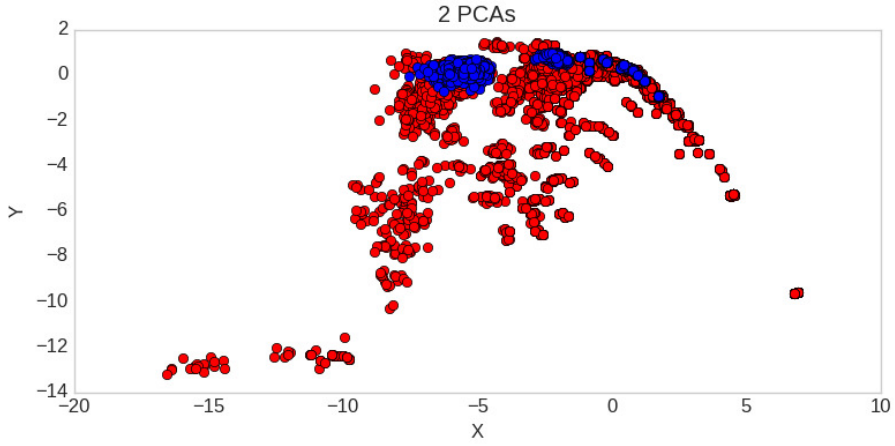

Fig. 5. PCA applied to the data set reducing from 8 dimensions to 2 dimensions (2 PCAs). On blue, the point corresponds to anomalous cases, and red points to normal cases.

In Figure 5 anomalous points form clusters in two regions making a clear identification of the anomalous cases, whilst normal cases are spread across the visualization space. The resulting PCA output graphic shows a clear classification and identification of the two classes that are on the data set, having a good graphic distribution of the points, allowing to say that it is possible to identify the attacks on the data set on a graphic point of view using PCA.

Figure 6 shows the variance from the resulting PCAs. As the previous section explained, the bigger the variance, the higher is the rate of information that is retained. In this case the first two PCAs retain more than 60% of the original information.

t-SNE results

For t-SNE was performed with 2 components and a random state of 0, using the Sklearn library. The results were from 8 dimensions to 2 dimensions.

As it can be seen in Figure 7, the t-SNE shows clusters of both classes on the center, allowing to identify the two classes of data. The algorithm produces clusters of the same type on different zones, showing an easy way to identify the majority of points, with a circular distribution all across the graph. Also, some minimum clusters can be seen on 3 different zones corresponding to anomalous cases, giving another point of view of the data with PCA.

Table 1. Linear relation between feature and principal component. Each coefficient represents correlation between a pair of feature and component.

Feature	PC1	PC2
URLlong	**0.4868**	0.1688
Characters	0.4669	0.0351
SuspWord	0.4404	0.0050
SQL	0.0497	0.0389
XSS	0.1086	0.5365
CRLF	0.2728	0.5070
Kolmogorov	0.4355	0.3168
Kullback	0.2693	**0.5685**

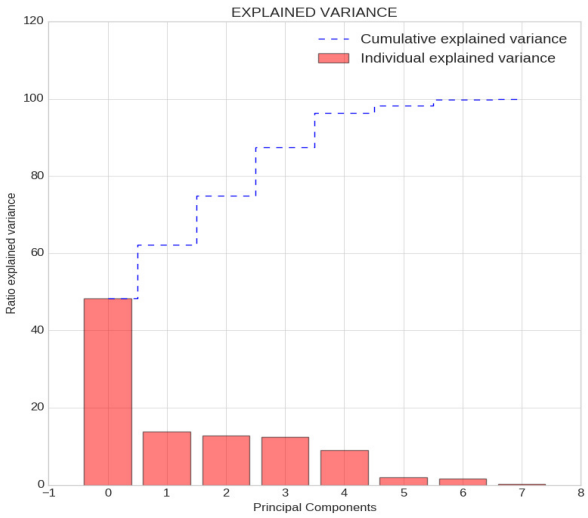

Fig. 6. Explained and individual variance of each PCA, sorted in decreasing order. The two selected PCAs were the two first ones, taking more than 60% of the total information from the original data set. .

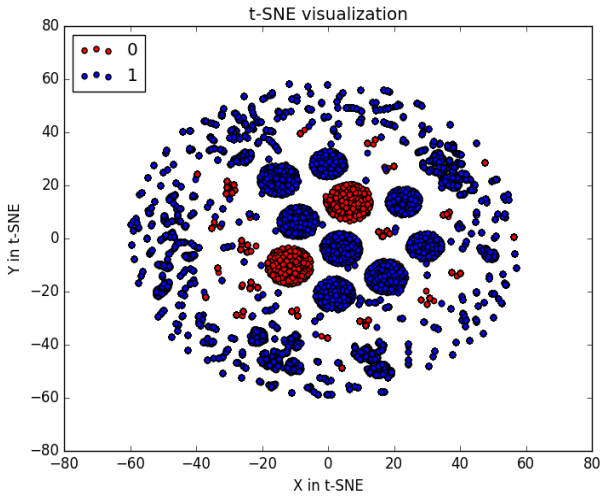

Fig. 7. t-SNE output on two dimensions. Blue points indicate anomalous cases and red points normal cases. Note that there are clusters of both classes on the center.

6.2 Machine learning classifiers

Taking into consideration the fact that all the used algorithms were of the type supervised learning, the data used the labeled tag of each sample, and the training set was split on 30% for testing and 70% for training or fitting on the classifier model choosing randomly the samples to be used on the two sections of data (training and testing). For the SVM Gaussian algorithm, they were used as parameters a C equal to 1.11 and a *gamma* equal to 0.09. For random forest, the number of estimators parameter was set to 100. All the algorithms were took from the Sklearn library for Python. According to Figure 8 and Table 1, the best algorithm was the random forest, with 83.2% of accuracy and an area under the curve (AUC) of 93.3%, followed by SVM Gaussian with 79.4% of accuracy and 87.09% of AUC. The others like Naive Bayes on both types, gives no concluding results with 65% of precision. Important to discuss is the null operational result of SVM sigmoid that has a very low score over all.

According to Figure 8, it can be said that the classification of anomalous and normal cases from the data set, can be improved by the tuning of parameters, with the fact that the feature engineering methodology was correct according to the visualization techniques and the results on the random forest and SVM Gaussian followed by the logistic regression, giving the idea that a better performance is possible on the improvement of parameters on the machine learning algorithms. That means, for better results, it is important to tune the algorithms and not to change the feature extraction process.

A key idea that can be extracted from the results is the possibility to make combinations of algorithms to improve outputs, as a result, on a production environment, a real case could be classified accurately using a combination of the machine learning algorithms. This is the case of the use of SVM Gaussian to get a better precision, but combining it with random forest and logistic regression to get better accuracy.

Table 2. Performance evaluation on classification anomalous cases

ML algorithm	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**AUC**
NB+Gaussian	0.72	0.20	0.32	0.64	0.65
NB+Multinomial	0.75	0.23	0.35	0.65	0.65
SVM+Sigmoid	0.00	0.00	0.00	0.59	0.39
SVM+Linear	0.81	0.26	0.39	0.67	0.69
SVM+Gaussian	**0.83**	0.62	0.71	0.79	0.87
Logistic Regression	0.70	0.41	0.52	0.68	0.70
Random Forest	0.75	**0.88**	**0.81**	**0.83**	**0.93**

7 Conclusions

Results show a good classification precision and accuracy, especially on the random forest and SVM Gaussian cases. Nevertheless, the rates of classification are not too high so as to said that they are very precise and can be used on a production software environment. However, it is a general view of the potential of the usage of visualization techniques that works fine to determine if the feature extraction was correct from a graphical point of view and the potential usage of other machine learning algorithms on the detection of web attacks, showing the previous results a possibility to use this kind of algorithms on a real environment on real users, with a good possibility to be improve and combined with other techniques, such as big data and other more powerful algorithms like neural networks and deep learning.

The present work could be used as an approach of a practical use of reduction of dimensionality to visualize from a graphic if the distribution of the classes inside the data set are correctly characterized and if the formation of clusters indicates a correct extraction of features in order to get better results on classification. Although, on the machine learning classifiers, further works related should focus on the improvement of parameters on the algorithms used on this work, especially on the ones that presented better results such as the random forest, SVM Gaussian and logistic regression. Even though the used classifier al-

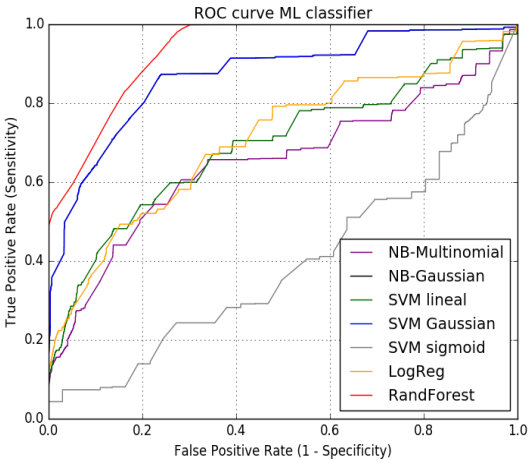

Fig. 8. Receiver Operating Characteristics - Area under the curve ROC-AUC graph that represents the overall performance of the machine learning classifiers, in this case seven types of them. The higher the curve, the better the performance.

gorithms presented some good results, in future work we will use other machine learning algorithms such as neural networks and deep learning.

References

1. Verizon Data breach 2018 investigations report, https://www.verizonenterprise.com/resources/reports. Last accessed 2 Jan 2019
2. Kaspersky Security Bulletin 2018. Threat Predictions for 2019, https://securelist.com/kaspersky-security-bulletin-threat-predictions-for-2019/88878/. Last accessed 2 Jan 2019
3. Sanjay Kumar, Ari Viinikainen, Timo Hamalainen, Machine learning classification model for network based intrusion detection system, in 11th International conference for internet technology and secured transactions (ICITST), 2016.
4. Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset , IEEE Communications and tutorials survey volume 18, Aegean University, Samos, Greece , 2015.
5. Kalyan Veeramachaneni, Alfredo Cuesta-Infante, Vamsi Korrapati, Costas Bassias, Ke Li Ignacio Arnaldo. (2016). AI 2: Training a big data machine to defend of MIT Massachusetts Institute of Technology.
6. Doyen Sahoo, Chenghao Liu, and Steven C.H. Hoi. (2017). Malicious URL Detection using Machine Learning: A Survey of Administration University of Singapore Information systems school.

7. Alejandro Correa Bahnsen, Eduardo Contreras Bohorquez, Sergio Villegas, Javier Vargas, Fabio A. Gonzlez. (2017). Classifying Phishing URLs Using Recurrent Neural Networks. of Easy Solutions Research, MindLab Research Group National University of Colombia.
8. OWASP top ten project 2017, https://www.owasp.org. Last accessed 2 Jan 2019
9. Hamoud Alshammari, Oussama Ghorbel, Mohammed Aseeri and Mohamed Abid. (2018). Non-Negative Matrix Factorization (NMF) for outlier detection in Wireless Sensor Networks of Al Jouf (Kingdom of Arabia Saud) University, CES Investigation center Sfax University (Tunisia).
10. Emeric Tonnelier, Nicolas Baskiotis, Vincent Guigue, Patrick Gallinari. (2014). Anomaly detection and characterization in smart card logs using NMF and Tweets of Paris University, La Sorbonne, Pars Francia.
11. Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, LiWu Chang. (2003). A Novel Anomaly Detection Scheme Based on Principal Component Classifier de Miami University, computational systems naval laboratory of investigations Washington DC.
12. Carmen Torrano Gimnez, Alejandro Prez Villegas, Gonzalo lvarez Maran. (2010). HTTP DATASET CSIC 2010. National Investigation Conselour, Informatio security institute Spain Government http://www.isi.csic.es/dataset/ . Last accessed 4 Jan 2019.
13. C. Torrano-Gimenez, A. Perez-Villegas, G. Alvarez. An anomaly-based approach for intrusion detection in web traffic. Journal of Information Assurance and Security, vol. 5, issue 4, pp. 446-454. ISSN 1554-1010 (2010).
14. T. Berners-Lee, L. Masinter, and M. McCahill, Uniform resource locators (url), Tech. Rep., 1994.
15. L.J.P. van der Maaten. Learning a Parametric Embedding by Preserving Local Structure. In Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AI-STATS), JMLR W and CP 5:384-391, 2009.
16. OWASP Cross-site Scripting (XSS), https://www.owasp.org/index.php/Cross-site_Scripting_(XSS). Last accessed 4 Jan 2019
17. OWASP Types of Cross-Site Scripting, https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting. Last accessed 4 Jan 2019
18. OWASP CRLF Injection, https://www.owasp.org/index.php/CRLF_Injection. Last accessed 8 Jan 2019
19. OWASP Guide Project, https://www.owasp.org/index.php/OWASP_Guide_Project. Last accessed 8 Jan 2019
20. OWASP SQL injection, https://www.owasp.org/index.php/SQL_Injection. Last accessed 8 Jan 2019