information

MDPI

Article

Anomaly-Based Method for Detecting Multiple Classes of Network Attacks

Anastasia Gurina [1,2,†] and Vladimir Eliseev [1,2,*,†]

[1] Moscow Power Engineering Institute, National Research University, Krasnokazarmennaya 14, Moscow 111250, Russia; asya.gurina001512@yandex.ru
[2] InfoTeCS JSC, Stary Petrovsko-Razumovsky Proyezd, 1/23, building 1, Moscow 127287, Russia
* Correspondence: vlad-eliseev@mail.ru; Tel.: +7-916-914-9889
† These authors contributed equally to this work.

Received: 17 January 2019; Accepted: 20 February 2019; Published: 26 February 2019

Abstract: The article discusses the problem of detecting network attacks on a web server. The attention is focused on two common types of attacks: "denial of service" and "code injection". A review and an analysis of various attack detection techniques are conducted. A new lightweight approach to detect attacks as anomalies is proposed. It is based on recognition of the dynamic response of the web server during requests processing. An autoencoder is implemented for dynamic response anomaly recognition. A case study with the MyBB web server is described. Several flood attacks and SQL injection attack are modeled and successfully detected by the proposed method. The efficiency of the detection algorithm is evaluated, and the advantages and disadvantages of the proposed approach are analyzed.

Keywords: anomaly detection; autoencoder; one-class classification; dynamic response model; vulnerability exploitation; lightweight intrusion detection; web server; SQL injection; flood attack

1. Introduction

Many computer attacks are carried out through the global Internet. It is becoming a phenomenon that affects not only companies and corporations, but also the policies of leading countries of the world [1]. There are many types of attacks with unique features. Some attacks are classified by their effect on the victim system (denial of service (DoS), website deface, code injection); others are named attack implementation techniques (distributed DoS (DDoS), buffer overflow, protocol flood, protocol amplification, etc.). The protection against all variations of attacks requires elaborate intrusion detection systems (IDS).

The problem of the creation of a perfect network attack detection method is highly complex. It is difficult to ensure high quality classification, low computational efficiency, low maintenance overhead, and universal application for a wide range of attacks at the same time. The methods, implemented in modern network intrusion detection systems are far from perfect, so there are many directions for their optimization and replacement with more efficient ones. The use of an approach to detect attacks as anomalies looks promising.

Online stores and marketplaces are the frequent targets of DoS attacks for obvious reasons: ransom and unfair competition. Code injection attack is also feasible due to potential theft of credit card numbers and valuable personal information. As a typical example of a network service, a bulletin-board web server was taken because it has the structure and scenarios of an online marketplace site. To model the code injection attack, we chose the version of the bulletin-board application with SQL injection vulnerability. Our anomaly-based method was used to detect network attacks on the MyBB web server.

The article is organized into six sections. The first one is the Introduction. The second section gives the related works and research results. The third section presents the main provisions of the proposed method: the dynamic response model of the web server and the autoencoder used to classify the dynamic responses. The anomaly detection method is introduced in the third section as well. The fourth section presents the results and analysis of the experiments. The fifth section discusses the results and identifies the limitations of the method. The final section concludes the paper.

2. Background

The most well-known representatives of DDoS attacks are TCP flood, SYN flood, UDP flood, ICMP flood, and HTTP flood. There are many methods to protect against DDoS attacks. However, DDoS attacks pose a serious danger and continue to cause enormous damage. A number of recent events [2–4] confirm this fact.

The proliferation of unsecured Internet of Things (IoT) devices has led to an increase in the number and power of DDoS attacks [4]. At the same time, attacks coming from IoT devices are difficult to detect. This is because the devices generate traffic over the TCP protocol, which is practically indistinguishable from the legitimate one; for example the Mirai botnet-generated TCP ACK + push and TCP SYN flood. Experts predict the emergence of even larger botnets capable of flood attacks even without the use of amplification protocols [5,6]. Thus, the task of timely detection of signs of the beginning of a flood attack is fundamentally important.

Various methods for detecting flood attacks were described in [7]. Furthermore, there are signature methods, statistical methods [8], and detection methods based on anomalies [9]. The DPI-system is able to analyze, monitor, and filter traffic. Therefore, it is often used to detect and protect against flood attacks.

Currently, network attacks have moved upward in the OSI model. They reside at the presentation layer [10] and the application layer [11]. Their goal now is not only denial of service, but also penetration into the system, resulting in data theft, data change, and control of the conquered service. This situation in general is caused by the imperfection of the software used for network data processing. One of the most dangerous and common vulnerabilities is "code injection" [12]. Code injection vulnerabilities are caused by software bugs and insufficient software testing [13]. According to [14], 25% of all vulnerabilities in service-oriented architecture (SOA) software systems are code injections. The most popular type of code injection attack is SQL injection (about 18% in the SOA class). Currently, nothing has changed. In fact, injection attacks have made the OWASP top ten list for the past 14 years and have been listed as the number one attack for the past eight years. Injection attack remains at the A1 position on the latest 2017 OWASP Top 10 list of most prevalent security threats [15]. This is caused by the widespread use of SQL queries to databases in application code based on user input values without proper control. Thus, the problem of detecting SQL injection attacks is important, which is regularly noted in the reports on the information security of web applications [16,17].

A standard well-known way to detect code injection attacks is to find a sequence of data (so-called, signature) that uniquely characterizes the attack against the background of normal requests to the web server [18]. It should be guaranteed that the signature matches only the attack and does not match any normal traffic, otherwise a false detection event will be produced. It is important to mention that the signature may match only the already detected attack, and the matching rule is usually written by a highly qualified specialist, who should pay attention to the generality and the performance simultaneously. The database of signatures is replenished with a delay, so there is some time gap for large-scale network attacks. This is a significant drawback of the signature approach. Another drawback is the large size of the intrusion detection system's signature database. Many attack signatures in the database are frequently working for nothing since they relate to attacks that are not dangerous for the protected network service. The large size of the attack signature database requires significant resources to analyze the traffic on the database and does not allow the signature method to be applied totally at the Internet service provider (ISP) level.

The problem of detecting code injections attack is actively investigated by the computer security research community. Hundreds of papers have been published to offer various methods for detecting SQL injection attacks, for example [19,20]. Most of the works are based on a signature approach enriched with methods to adopt polymorphic representation of injections and to eliminate the insignificant information from the analyzed query. The comprehensive information on the types of SQL injection attacks and the tools to combat them are presented in numerous reviews, for example in [21–23]. However, the problem continues to be relevant, because SQL injection attacks rely on the individual features of the attacked web application, the programming language on which it is written, the database structure, and the database that serves this database. Not surprisingly, new SQL injection attacks are not always recognized by intrusion prevention systems. Attempts have been made to reduce the individuality of queries, which should simplify the recognition of SQL injection attacks [19], but this does not solve the problem in principle.

One of the promising approaches to identifying attacks of various kinds is to use machine learning methods. In particular, such methods have been used for a long time to detect SQL injection attacks and XSS (cross-site scripting) attacks similar to them. The motivation for using machine learning is the ability to detect implicit dependencies in the data. Presumably, it will allow detecting not only known, but also unknown attacks. Let us consider some research works on this topic. All machine learning-based methods differ in the parameters selected to compose the area of interest and to find the proper characteristic of real world events in this area.

In [24], as a distinctive feature of malicious requests, the frequency of symbols, classes of symbols, and keywords of SQL language was considered. Training of the neural network recognizer was performed on a synthetic sample containing both normal SQL queries received during the operation of a particular site and queries with known attacks. The peculiarity of the implemented approach was the individual configuration of the neural network recognizer on the protected site.

A similar approach was demonstrated in [25]. It used a probabilistic model to detect the symbol of a particular class in the arguments of requests to the database server. Code injection attacks will obviously change the structure of the query and the distribution of the characters' probabilities. Building a probabilistic model requires individualization for a specific application. In this work, the PHP-Nuke system was used as the target application, which at that time had several known vulnerabilities of the SQL injection type.

The construction of a model of valid arguments in the form of intervals was the basis of the approach proposed in [26]. The method demonstrated good efficiency in terms of errors of the first and second kind, but required a manually-created specification of the exchange protocol. This reduces the applicability of this approach.

Another way to detect database queries containing code injections is to build a profile of normal queries [27]. Requests containing injections will be marked as not matching the profile. This makes it possible to detect a wide range of SQL injection attacks, including previously unknown ones, but requires individual configuration for each application accessing the database.

To identify the constant part of the text of SQL queries to the database and their arguments automatically, in [28], a genetic algorithm was used. Training of the anomaly detector was conducted according to the protocol of the normal operations of the DBMS. There was a high efficiency for the method of detecting anomalous queries, but also a high rate of false positives recorded, which in general, is typical of many algorithms for anomaly detection.

The original method of constructing a set of correct SQL queries was proposed in [29]. The method uses a neural network with feedback. The neural network is trained to recognize chains of a fixed number of tokens that form requests. The recursive nature of connections in the neural network makes it possible to recognize chains of tokens of any desired length. The PHP-Nuke system was used as an object for experiments.

Sometimes, to build generalizing rules for classification of query texts, a representation in the form of N-grams is used [30,31]. In the resulting feature space, different methods of single-class classification are used: support vector machines and neural networks.

We can see that many researchers offer machine learning algorithms to detect SQL injection attacks based on anomaly detection. In some works, the anomaly detection algorithm is individualized to a specific application that needs to be protected from SQL injection attacks.

The main thesis of the majority of works on the detection of network attacks is that all necessary information about the fact of the attack is contained in one direction of traffic. Typically, the signs of an attack are to be found in incoming traffic from the Internet, but it is also known that the signs, for example of an infection of computers in the local network, manifest themselves as specific packets coming from the local network to the command centers located somewhere on the Internet.

In previous works [32,33], authors have shown that the joint analysis of input and output data allows detecting anomalies. A promising approach to detect SQL injection attacks [34] is based on accounting for the execution time of normal SQL queries. These approaches can make the detection of network attacks efficient and generalized to detect several kinds of attacks.

3. Method

3.1. Principles of the Proposed Approach

Based on the most general considerations, we can postulate that a network attack is a violation of the normal functioning of a computer system. Obviously, by functioning, it is meant the logic of data processing and interaction with other systems. In particular, for a web server, processing of data consists of forming a response to a request received from outside. Thus, a violation of the normal functioning of the web server is not every "attack" request, even if it contains the signs of a known network attack, but only the one that caused the web server to function incorrectly.

Let us introduce the concept of the correct functioning of a web server. This is a collection of request-response pairs that describe all possible normal scenarios of data processing by the server. Since many classes of web servers provide very limited functionality, the incoming requests and responses to them are mostly typical. In this case, it is possible to create some template for normal interaction with the web server, which generates a limited number of unique request-response pairs of different types. Having such a set will allow us to check the similarity of each request and response with a request-response pair from the set and identify network attacks, the response to which would go beyond this set. The formation of a template of normal interaction with the web server can be automated.

To create such a set, it is necessary to formalize the characteristics of requests and responses to them, containing information about the main parameters and features of the web server response to the request. Obviously, the assumption is that the characteristics calculated during normal interaction with the web server will be similar to the characteristics of the request-response pairs from the set, and the implementation of malicious actions that cause an incorrect reaction of the web server will affect the characteristics of the request-response pairs, which will allow us to distinguish one from the other.

In this paper, for the identification of request-response pairs, it is proposed to use the relationship between the size of the data exchanged between the server and the client and the timing of this exchange. This choice was made due to the following considerations. Each request and each response are characterized by their size. Since information processing takes some time, the timing of the response data stream also should be considered as an important characteristic. The cross-correlation function (XCF) for the time series of the incoming and outgoing web server traffic intensity was adopted as a formalized characteristic that combines both sizes and timing [33].

The XCF form shows the uniqueness of the request-response pair when the web server processes a normal request. Determining the space of normal XCFs will make it possible to identify anomalous XCFs that go beyond its limits. Figure 1 shows a scheme of an anomaly detection method based on XCFs. The scheme also shows the difference in the form of XCFs when the web server processes normal requests and when processing anomalous requests with attacks.

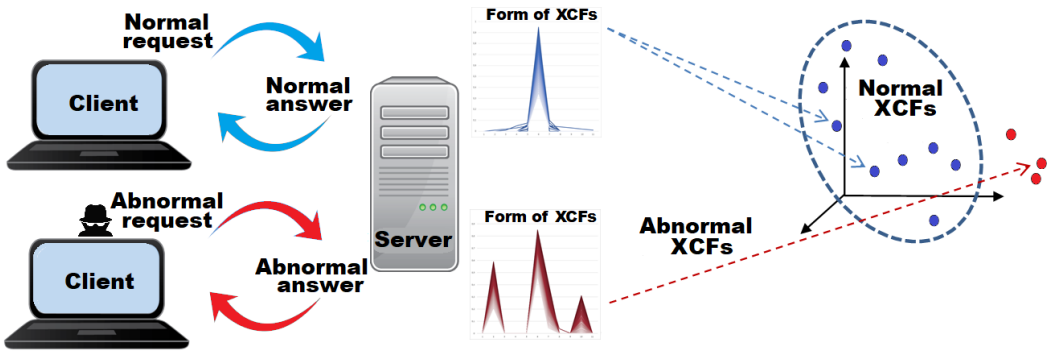

Figure 1. Scheme of detection of abnormal responses of the web server to requests. XCF, cross-correlation function.

To detect anomalies in the multidimensional feature space, one can use well-known classification methods, such as neural networks, support vector machines, etc. [35,36].

In this paper, for single-class classification, we used an autoencoder trained on the set XCF, calculated from the time series of the intensity of incoming and outgoing server traffic in the normal interaction with it. The anomaly detection criterion is based on the error of XCF recognition submitted to the input of a trained autoencoder. If the error exceeds the set threshold for an XCF submitted to the autoencoder input, the XCF and its corresponding request-response pairs are considered abnormal.

3.2. Dynamic Model of Request Processing by the Web Server

Consider the formal model of a web server that processes incoming requests from the network and issues back responses. In this case, we will characterize the queries and web server responses only by their size. We impose a restriction on the flow of requests so that it is ordinary, for example Poisson arrivals, that is, at any moment, no more than one request arrives. This assumption with a high accuracy is valid for web servers with a small number of visitors and small traffic. In this case, it is also a fair assumption that the web server response will be fast enough and in most cases will end before the next request comes.

We also postulate that the server's response to the request does not depend on the order of requests, but depends only on the information contained in the request. Of course, in the general case, the server can accumulate information in its memory, and this affects its responses at the following times. However, the general logic behind the development of network protocols and web servers limits the size of the answer: the search engine is always limited to 10 results found; there are no more than 20 items in the storefront at a time; there is only one large picture on the photo gallery page; there is a limited number of thumbnails of other photos, etc. The reason for this feature is that the web server must respond to the request within a short and guaranteed time. The answer, not limited in size, will slow down the responses to other requests coming to the web server.

Figure 2a shows an example of the intensity of incoming and outgoing traffic when the web server processes requests of the same type. Figure 2b shows an example of the intensity of incoming and outgoing traffic when the web server processes requests of two different types. Requests of different types have a different form of intensity and are indicated by different colors for clarity.

One can see an analogy between the example shown of processing network requests and the input-output of the transfer function of the dynamic plant used in the theory of automatic control. The traffic intensity measured at discrete instants of time, according to this analogy, can be considered as a signal, and the web server acts as a dynamic plant that converts the signal. Developing this analogy, we see that the web server's processing of requests of different types can be represented as a scheme, which is shown in Figure 3. In this case, the web server performs a query type determination and then processes each request using its own block: G_1 or G_2.

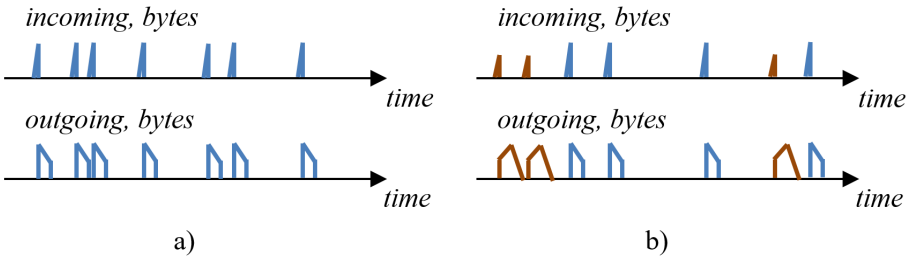

Figure 2. Intensity of incoming and outgoing traffic, if the web server processes requests of one type (**a**) and requests of two types (**b**).

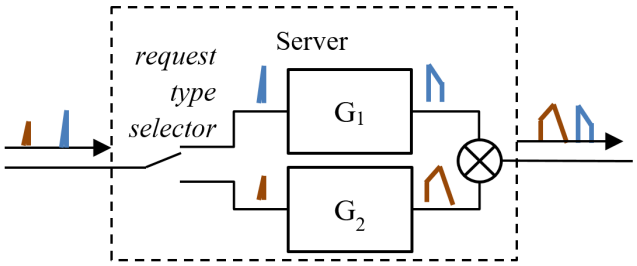

Figure 3. Dynamic model analogue to highlight two types of request processing.

In the case of a larger number of query types, this model is easily extended by adding parallel processing units. This model allows us not only to describe the functioning of the anomaly detection algorithm formally, but also offers methods for modeling abnormal situations.

From the theory of automatic control, it is known that in the case of a linear model of a signal processing unit, its transfer function can be equivalently represented by means of XCFs. For a web server, the linearity of the dynamic response model is not obvious. For example, the same request to a web server can be executed quickly if the necessary information is in the cache of the DBMS, and slowly if you have to read this information from the hard disk of the server. This difference is manifested by two different XCFs. Accordingly, in the model, this behavior of the web server can be represented by two different processing units.

3.3. Features of the Normal Processing

As noted earlier, the set of features on which it is proposed to detect anomalous behavior is the correlation characteristics between the intensity of incoming and outgoing traffic, characterizing the processing of incoming requests. In this case, both Pearson correlation coefficients [32] and the correlation functions [33] can be used. The number of points in the XCF in discrete time is thus the dimension of the feature space in which one can attempt to construct a region of normal queries. For example, if XCF has a width of three, then the object space is three-dimensional.

The combination of these features at any time of the web server in the absence of other influences characterizes the processing of requests. Let us call the vector of such features an instant correlation response of the web server. In order to determine the whole set of these responses, that is the profile of normal functioning, the following approach is proposed. Based on the description of the site structure or user's guide, it is possible to fully test all web server functions, including working with user data of different sizes and simulating situations that cause error handling. In the process of such testing, two time series are recorded that determine with the required frequency the number of bytes transferred during incoming requests and their processing by the web server. These time series are converted into instant correlation responses, forming a base with the profile of the normal functioning of the web server.

With full testing of the web server's capabilities, it is necessary to provide it with different types of normal load, including periods of intensive receipt of requests. This will provide a variety of dynamic web server responses and realism of the created profile in terms of real operation.

Full testing can be replaced by a period of trial operation, within which users themselves create a normal operation profile by their requests. The main thing is that at this time, the web server should be under the supervision of the system administrator, and during this period, the web server should not be successfully attacked from outside, because it is undesirable to include information about abnormal events in the profile of normal operation.

3.4. One-Class Classification for Anomaly Detection

The profile of normal functioning in implicit form contains information about the region of admissible correlation responses. If we construct a classification rule that will allow us to distinguish points within this region from points outside, then the problem of detecting anomalies will be solved. To construct such a one-class classifier, there are different techniques [35,36].

One of the options for implementing a one-class classifier is the autoencoder. The feature of the autoencoder consists of the special structure of the neural network, when the number of inputs is equal to the number of outputs. In the average hidden layer of the autoencoder, the number of neurons must be less than the number of inputs. This layer is called a "bottleneck", in which the incoming information is compressed. In the "bottleneck" of the autoencoder are identified the most common features of the input vector. In the process of learning, the autoencoder learns to reproduce at the output of the vector (point) of the training set the profile of normal operation. Thus, for points that are present in the profile of normal operation and close to them, the autoencoder at the output will receive approximately the same points. However, the points that are significantly distant from the normal ones will not be recognized by the autoencoder, and the output value will be very different from the input one. Thus, the criterion of anomaly can be the exceeding of a certain threshold value of the instantaneous reconstruction error (IRE) of the autoencoder. The IRE is calculated as the Cartesian distance between the coordinates of the points at the input and output of a trained autoencoder. The threshold value of the IRE for the detection of anomalies is defined as the maximum value of the IRE on the data of the normal operation profile. The IRE threshold is required to enable anomaly detection and analysis of all detected anomalies in the web server.

3.5. The Web Server Implementation

As an object for testing the proposed algorithm for detecting anomalies, consider a web server with a deployed site on the basis of the bulletin-board application MyBB. This software product is a fairly popular engine, using the standard tools: PHP, MySQL, Apache2.

We are interested in the version of MyBB 1.8.6, in which the SQL injection [37] vulnerability was discovered in 2016. The vulnerability allows entering in a numerical field a fragment of SQL code that will be executed by the DBMS because of incomplete control over the input values. Thus, we are dealing with a usual SQL injection vulnerability. A small peculiarity is that when one fills the web form with code injection and sends it to the web server, everything in the reply looks normal. However, when performing an operation for accessing the forum with an incorrect identifier, instead of the usual request error message, a text appears in the web browser window as a result of executing the injected code, as well as an error message about failed MySQL operation. The result of the SQL injection attack is shown in Figure 4.

Thus, this vulnerability allows the embedded code to be started indefinitely, which can potentially be very dangerous. It is interesting to note that even six months after the vulnerability was discovered, Version 1.8.6 continued to be used on many sites and was part of some docker containers that simplified the procedure for creating new sites.

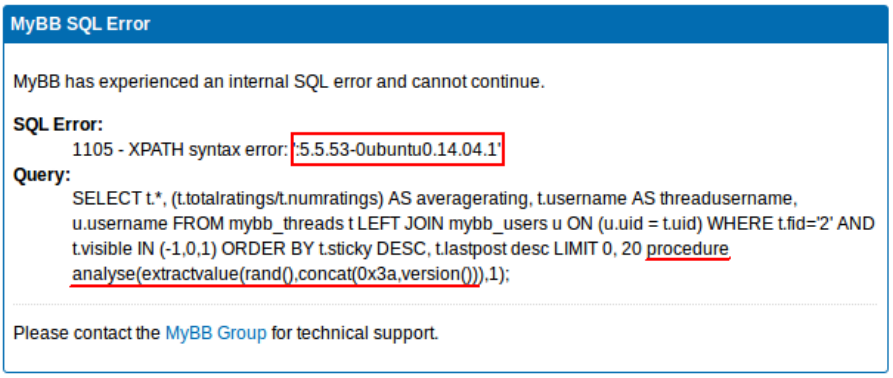

Figure 4. SQL injection effect with produced output (red box) and injected code (red underscore).

3.6. The Classification Algorithm Implementation

To protect the site based on the selected version of MyBB, a training set must first be created. In our case, it is composed of network traffic with usual operations with the forum including usage of communication functionality (creation of a short message, a picture, a paragraph, several paragraphs), change of settings, user mistakes, and incorrect actions. Further on in this scenario, several sessions of work with the forum were simulated. At the same time, using the network sniffer **tshark**, all incoming and outgoing web server traffic was recorded. The received data on communications between the client and the server via the HTTP protocol were converted into time series of the data transmission intensity, which was represented by the number of bytes per unit of time. The time sampling step was 0.1 s. This allowed seeing instant changes in the processing network device behavior.

Two synchronous time series with the intensity of incoming and outgoing traffic was used to calculate the XCF with a window width of 11 time samples. Thus, it was possible to register a server's dynamic response to the request within 0.5 s. A significant number of XCFs should have been calculated for intervals with zero traffic intensity. Such intervals were excluded from the calculations. The obtained 4830 cross-correlation functions formed the normal operation profile of the web server: a training set.

The structure of the autoencoder for the classification of XCFs was chosen based on the width of the correlation window and expert ideas about the possibilities of the autoencoder. The experiments used an autoencoder with 11 neurons in the input layer, which is equal to the width of the discrete cross-correlation function, and successively located fully-connected neuron layers of 15, 10, 6, 3, 5, 10, and 15 neurons and 11 neurons in the output layer. The structure of the autoencoder for the one-class classification is presented in Figure 5.

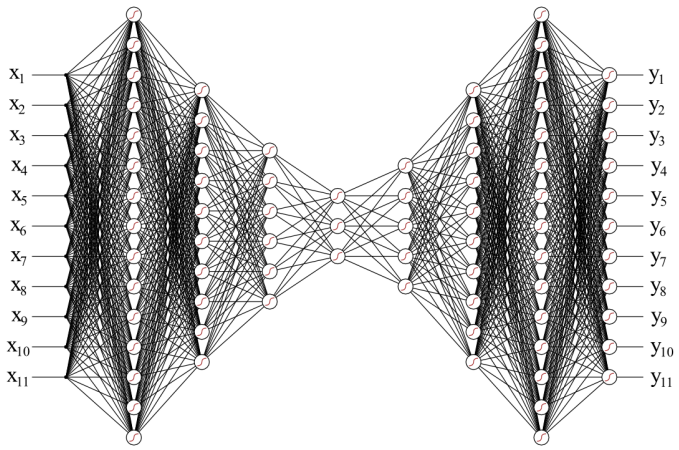

Figure 5. Autoencoder structure.

The input of the autoencoder was fed the vector of values of the XCF $x_1, ..., x_N$. The autoencoder was trained so that the outputs formed a vector of values $y_1, ..., y_N$, as close as possible to the vector of input values of the XCF $x_1, ..., x_N$. The training used the criterion for minimizing the mean squared error (MSE). The autoencoder was trained in the MATLAB package by the Levenberg–Marquardt method in a training set.

In the middle of the autoencoder neural network, there was a "bottleneck" layer, which consisted of 3 neurons. The output of these neurons can be plotted in 3D to highlight the compression of the multidimensional ($N = 11$) input vector data by the neural network. Sometimes, such visualization can help to understand how the neural network works and to distinguish different datasets.

To find out whether the trained autoencoder recognizes the input XCF or not, the instant reconstruction error (IRE) for a given input vector x and produced output vector y is defined by the formula:

$$IRE = \sqrt{\sum_{i}^{N} (x_i - y_i)^2}. \quad (1)$$

The value of IRE is closer to zero if the input vector is closer to one of the input vectors in the training set and is higher if the input vector differs from all vectors from the training set. Therefore, IRE characterizes the novelty of the input vector x. The threshold value for the IRE allows implementing the one-class classification algorithm as separate vectors, a low IRE value from vectors with a higher IRE value.

3.7. The Attacks' Implementation

To test the method of detecting attacks, different scenarios for working with MyBB were prepared, including not only regular queries, but also incorrect actions with the site: SQL injection, the generation of flood attacks.

To implement various types of DDoS attacks, the utilities and tools of the Kali Linux distribution were used, designed for information security professionals and used for testing web servers. The hping3 utility was used to generate the SYN flood, TCP flood, UDP flood, and ICMP flood, and the siege utility was used to generate the HTTP flood to the MyBB web server.

According to the network traffic recorded during the implementation of the scenarios, 6 sets of XCFs were obtained in the same way: test set with SQL injection attacks (SQLIAs) and 5 test sets with SYN flood, TCP flood, UDP flood, ICMP flood, and HTTP flood attacks.

4. Case Study

The developed autoencoder-based anomaly detection method was used to detect several types of network attacks on a MyBB web server with the known SQL injection vulnerability. A template for regular interaction with the MyBB web server was prepared and implemented, according to the time series of the intensity of the incoming and outgoing traffic of which a training set was formed, consisting of XCFs, ass well as test sets containing various attacks.

4.1. SQL Injection Attack Detection

After training the autoencoder for each vector of the XCF of the training set, an IRE value was obtained that characterizes the quality of input XCF recognition. The graph of IRE values for the normal operations with MyBB is shown in Figure 6. The maximum value of IRE was equal to 0.0068.

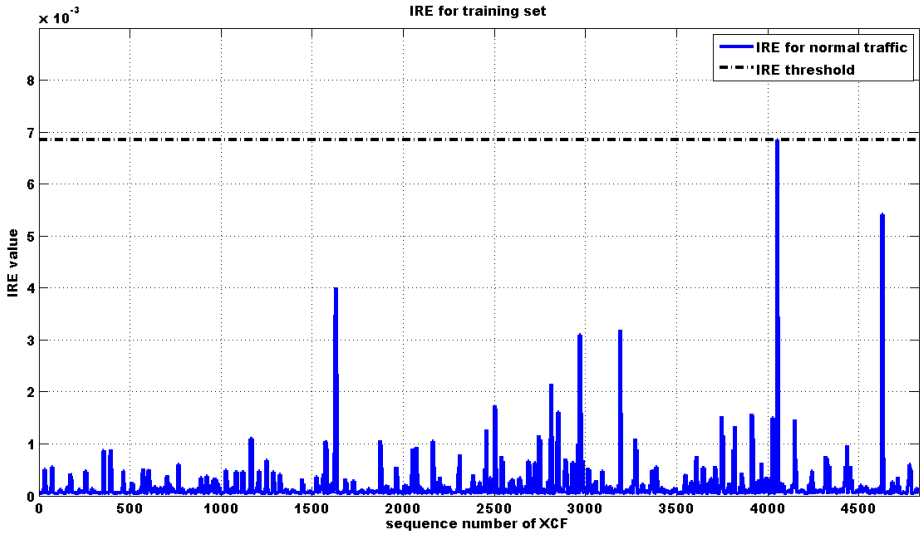

Figure 6. Instant reconstruction error (IRE) for each element of the training set.

To consider an exact value of IRE as an anomaly, it is necessary to select the threshold value. It is rational to choose the threshold value to be the largest value among the reconstruction errors calculated for the training set. In our case, this was 0.0068.

To test the trained autoencoder for the detection of anomalies, a test set with several SQL injection attacks and with normal operations should be prepared. We made three SQL injection attacks among normal operations with MyBB. The exploited SQL injection vulnerability of MyBB included two phases: query with SQL injection (POST SQLIAs) and further call of the embedded code (GET SQLIAs). According to the graphs of the intensity of the measured incoming and outgoing traffic, XCFs were calculated. Then, the vectors of the consequent intensities were fed to the input of the autoencoder to produce output vectors. The calculated IRE graph is shown in Figure 7 together with the graph of IRE for training data and the threshold line.

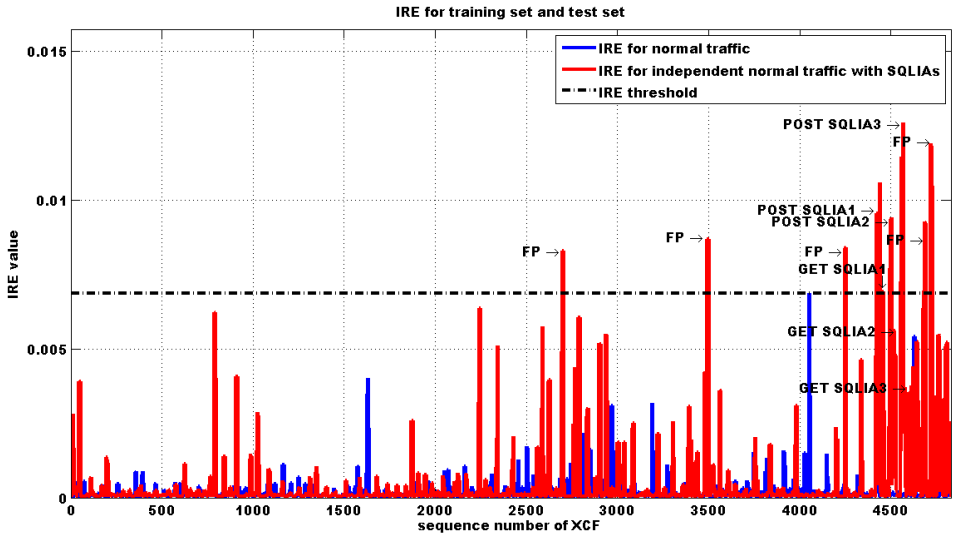

Figure 7. Instant reconstruction error for each element of the training set and test set containing samples corresponding to SQL injection attacks (SQLIAs).

Several anomalies were found in the test data. Three POST SQLIAs' attacking requests and one of GET SQLIA's request were detected. This means that all three SQL injection attacks were found by the

attack detection algorithm. Some of the detections were false positives (FP), which were not attacks. The presence of false positives can be explained both by shortcomings in the quality of the autoencoder training and by disturbances of the runtime environment of the MyBB, since other operating system processes may change the timing of requests' processing.

The delay to find the attack is equal to half of the window width to calculate XCF. The whole window width was $11 \times 0.1 = 1.1$ s. Therefore, the delay was 0.6 s. This is the minimum delay that can be delivered by the algorithm due to its parameters: sampling rate of 0.1 s and number of points in the XCF window $N = 11$.

Figure 8 shows the graphs of several XCFs, both normal (with small IRE) and anomalies, true and false positives.

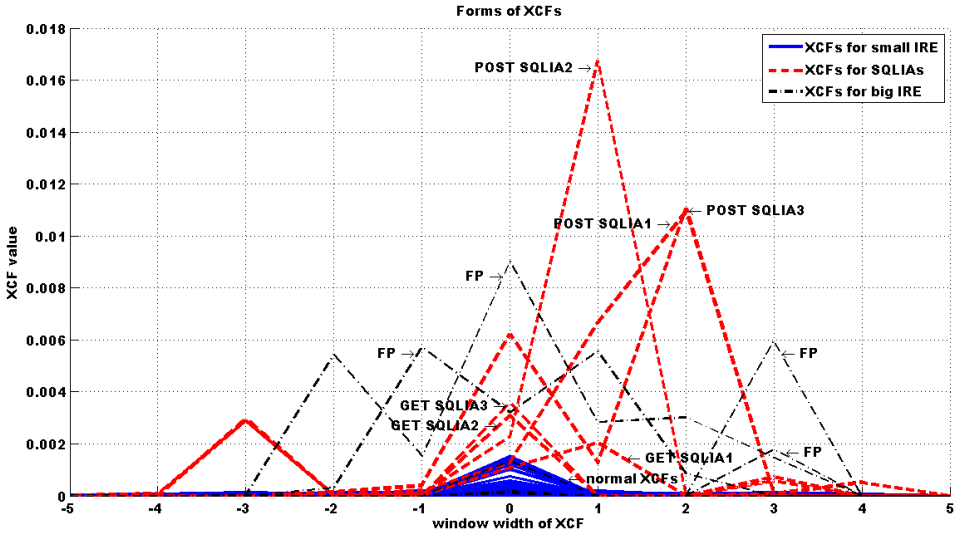

Figure 8. Forms of normal and abnormal XCFs.

The map of the "bottleneck" layer for the training and test sets is represented in Figure 9. Anomaly points are specifically marked. The area of normal points is encircled by a blue line.

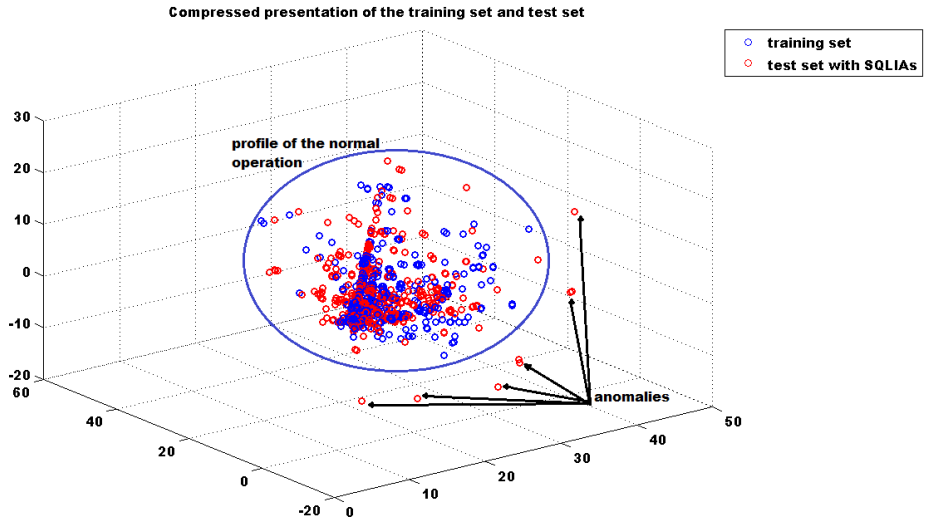

Figure 9. Cluster of normal XCFs and anomalies.

It can be seen that most of the XCFs from the test set were recognized by the autoencoder and referred to the field of normal XCFs that characterize the profile of normal operation. Some XCFs that were separated from the selected normal area were found by the autoencoder as abnormal; some of them, indeed, characterized the server's response to malicious SQL queries, and the other part the false positives.

Thus, the autoencoder-based anomaly detection method was successfully applied to detect anomalous server responses: three SQL injection attacks against the MyBB service.

The error matrix of the classifier for the test data is represented in Table 1. The false positive rate on the independent test set of 4830 XCFs was 0.104%. The accuracy was equal to 99.8%.

Table 1. Error matrix for the test set with SQL injection attacks.

	True	False
Positive	4	5
Negative	4819	2

4.2. Detection of Flood Attacks on the MyBB Web Server

Since the method of attack detection does not consider network traffic inspection and is independent of the nature of a network protocol, we considered applying the trained autoencoder classification algorithm to detection of flood attacks. Every type of flood attack was combined with normal operations with the MyBB web server.

4.2.1. SYN Flood Detection

For the case of the detection of SYN flood, the test sets of XCFs were calculated using the data of the intensity of incoming and outgoing traffic of the web server MyBB, which first processed only legitimate requests and then also SYN flood packets. After the test set autoencoder was fed to the input, the IRE values were calculated for each test set element. The IRE graphs both for training and test sets are shown in Figure 10.

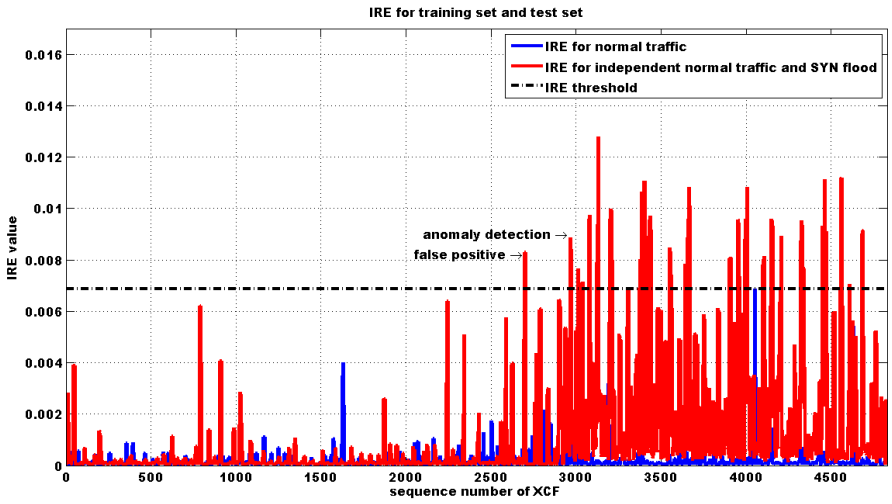

Figure 10. Instant reconstruction error for each element of the training set and test set containing samples corresponding to SYN flood.

Exceeding the IRE threshold means anomaly detection. One false positive detection is marked on the figure before the actual start of SYN flood attack. After analyzing the traffic, it was found that the autoencoder detected an SYN flood attack, with a delay of 19.2 s. The presence of delayed detection can be explained by the fact that TCP packets with the SYN flag set are not malicious and anomalous, and it is very difficult to distinguish legitimate SYN packets from SYN flood.

The comparison of XCF graphs for normal, false positive, and SYN flood traffic is represented on Figure 11.

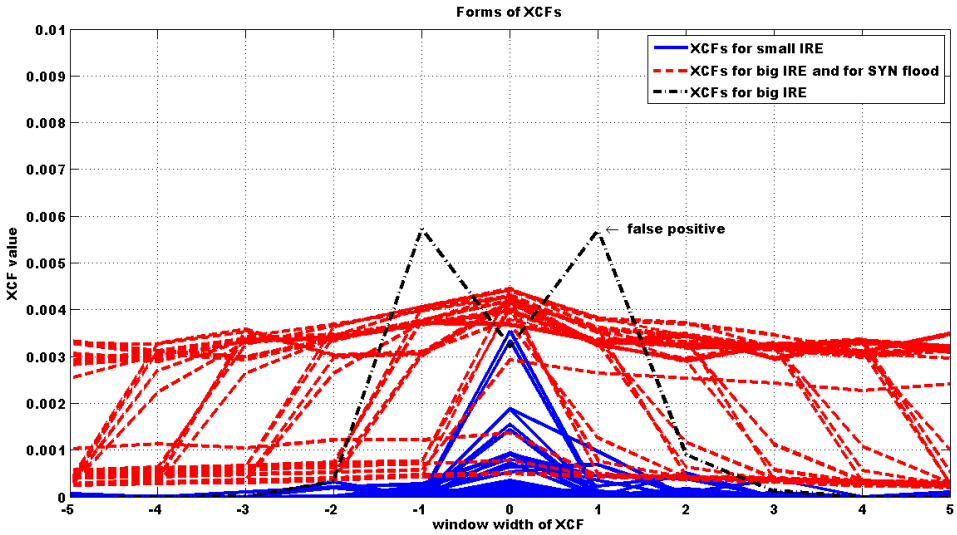

Figure 11. Normal XCFs from a training set and XCFs from a test set after starting an SYN flood.

The 3D map of the "bottleneck" neurons displays a mixture of normal and SYN flood traffic (Figure 12). Probably, the clouds of red points at the top and at the bottom of the 3D map relate to SYN flood traffic. The center part of 3D map shares both training and test set points, which represent normal traffic.

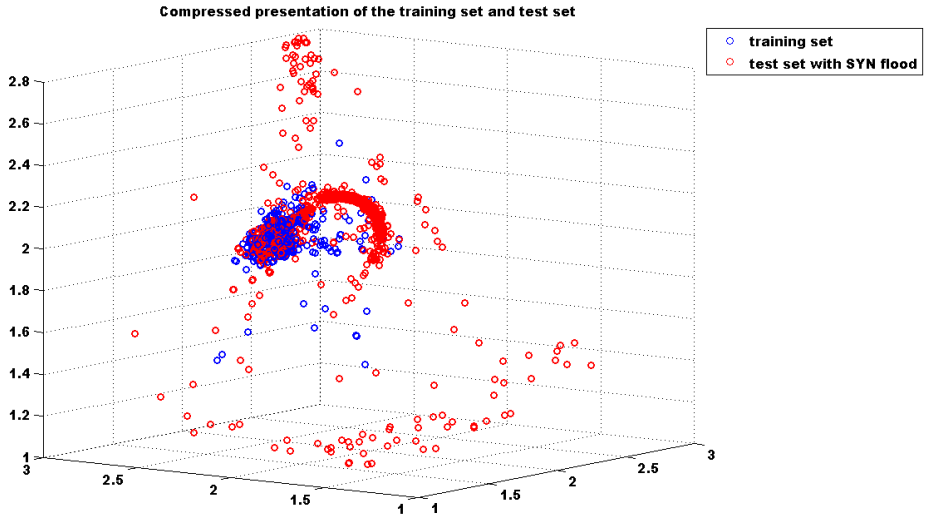

Figure 12. Compressed presentation of the training and test set, which includes XCFs corresponding to the SYN flood.

Error matrix for SYN flood detection is represented in Table 2. FPR on the test was equal to 0.036%. The accuracy was equal to 61.9%.

Table 2. Error matrix for the test set with SYN flood.

	True	False
Positive	208	1
Negative	2783	1838

4.2.2. TCP Flood Detection

Similar experiments were performed to detect the TCP flood. The IRE graphs both for the training and test sets are shown in Figure 13.

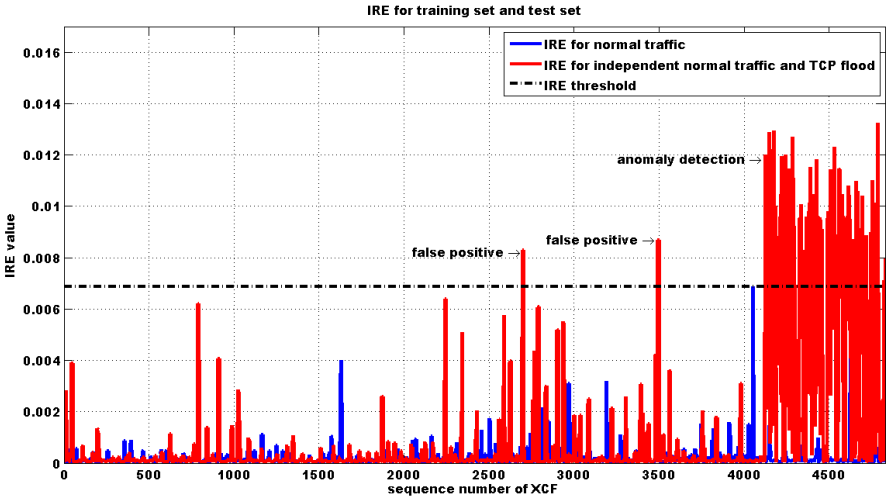

Figure 13. Instant reconstruction error for each element of the training set and test set containing XCFs corresponding to TCP flood.

The attack was detected with a delay of 28.3 s. Before the start of the attack, two false positive detections were found. The latency can be explained by the fact that TCP packets are not malicious and abnormal, and it is very difficult to distinguish legitimate users from TCP flood.

Consider the form of unidentified autoencoder XCFs, and compare them with those for which IRE is small. Figure 14 shows some of the XCFs from the training set and some XCFs from the test set, starting from the moment that the autoencoder detects anomalies.

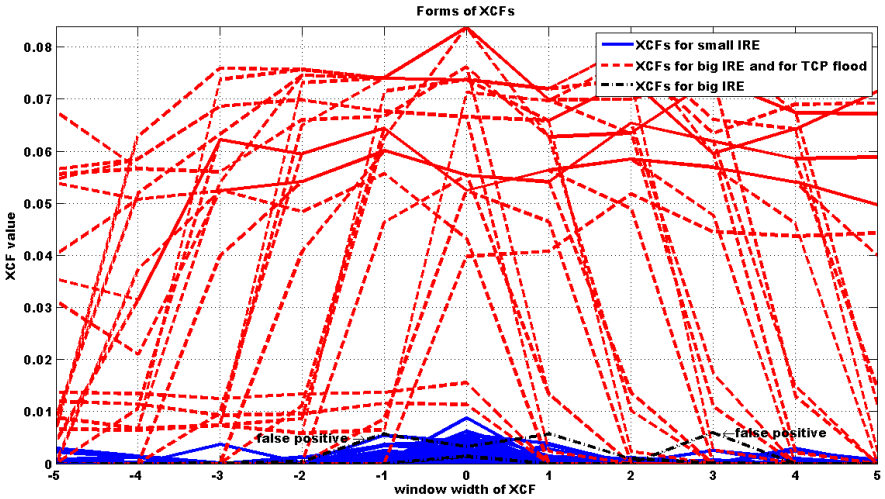

Figure 14. Normal XCFs from a training set and XCFs from a test set after starting a TCP flood.

The 3D map of the "bottleneck" neurons displays a mixture of normal and TCP flood traffic (Figure 15). The center part of the cloud contains both training and test set points. They obviously correspond to normal traffic. All points outside the center cloud probably be related to TCP flood attack.

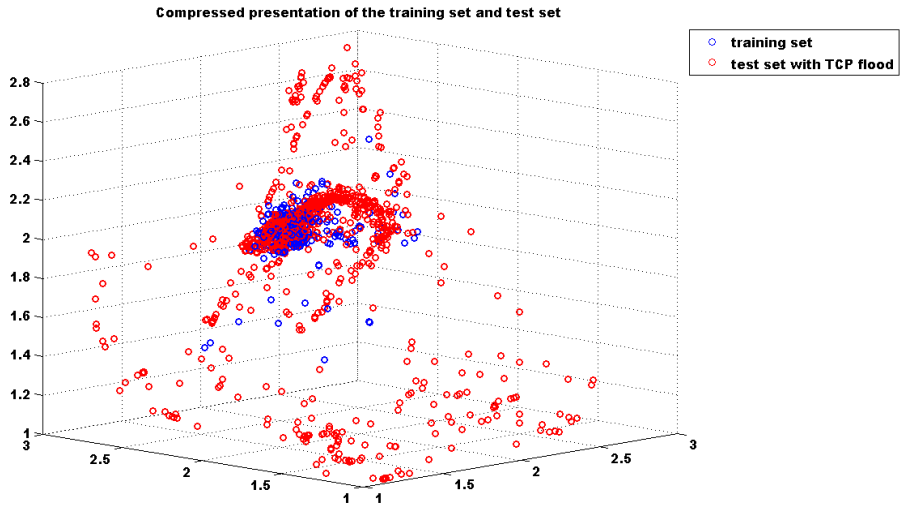

Figure 15. Compressed presentation of the training set and test set, which includes XCFs corresponding to the TCP flood.

The error matrix for TCP flood detection is represented in Table 3. FPR on the test was equal to 0.052%. The accuracy was equal to 87.1%.

Table 3. Error matrix for the test set with TCP flood.

	True	False
Positive	361	2
Negative	3848	619

4.2.3. UDP Flood Detection

Let us observe the result of the application of the developed algorithm on UDP flood attack. Since only the intensity of the traffic is used, the sum of UDP and TCP (web) traffic intensities may be evaluated by the classification algorithm. Therefore, the test set contains normal requests and responses together with UDP flood attack. The IRE graphs both for training and test sets are shown in Figure 16.

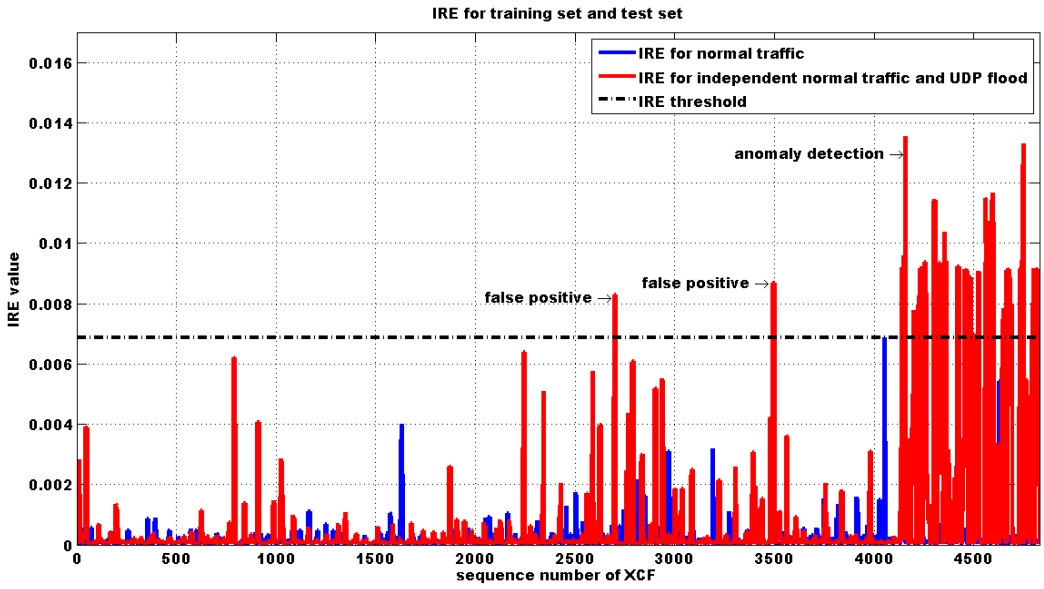

Figure 16. Instant reconstruction error for each element of the training set and test set containing XCFs corresponding to UDP flood.

The attack was detected with a delay of 4.7 s. Before the start of the attack, two false positive detections were present. Consider the form of unidentified autoencoder XCFs, and compare them with those for which IRE is small. Figure 17 represents some of the XCFs from the training set and some XCFs from the test set, starting from the moment that the autoencoder detects anomalies. The amplitude of anomalous XCFs differ significantly from normal traffic's XCF, but the form of normal and anomalous XCFs is the same.

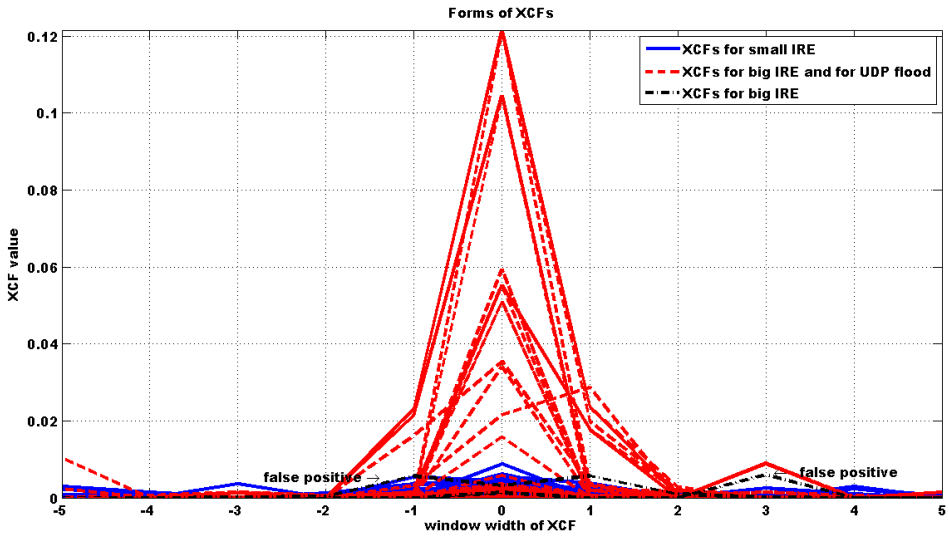

Figure 17. Normal XCFs from a training set and XCFs from a test set after starting a UDP flood.

On the 3D map of the "bottleneck" neurons, most of the anomalous points are located on the left margin of the points' cloud near the cluster of normal traffic points (Figure 18).

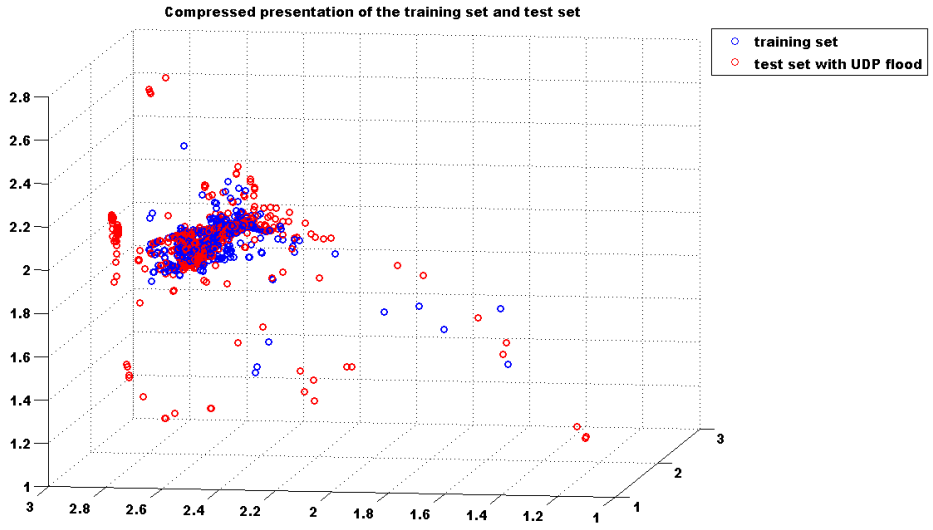

Figure 18. Compressed presentation of the training set and test set, which includes XCFs corresponding to the UDP flood.

The error matrix for UDP flood detection is represented in Table 4. FPR on the test was equal to 0.049%. The accuracy was equal to 89.8%.

Table 4. Error matrix for the test set with UDP flood.

	True	False
Positive	232	2
Negative	4091	505

4.2.4. ICMP Flood Detection

Further, an experiment with ICMP flood was performed the same way as with UDP flood. The IRE graph is shown in Figure 19.

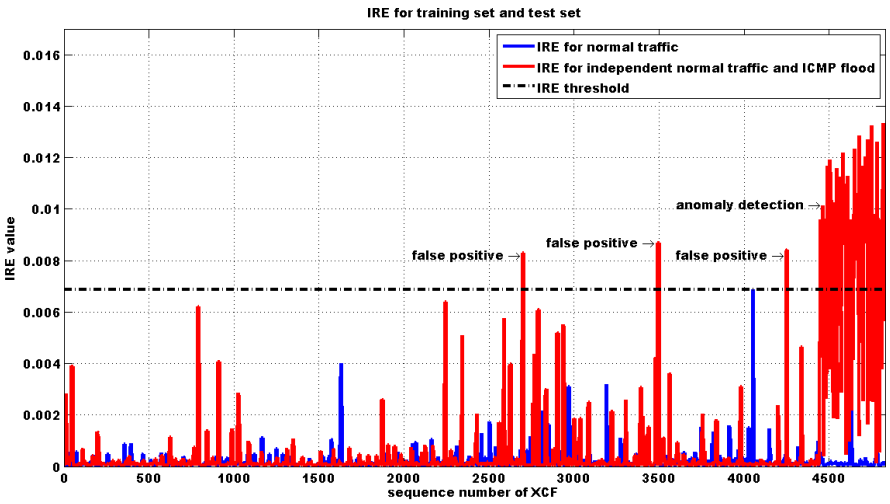

Figure 19. Instant reconstruction error for each element of the training set and test set containing XCFs corresponding to ICMP flood.

The attack was detected with a delay of 1.3 s. Before the start of the attack, three false positive detections were found. The form and the amplitude of anomalous XCFs related to ICMP flood differed greatly from XCFs of the normal traffic (Figure 20). Thus, the autoencoder has to detect ICMP flood attack without any problem.

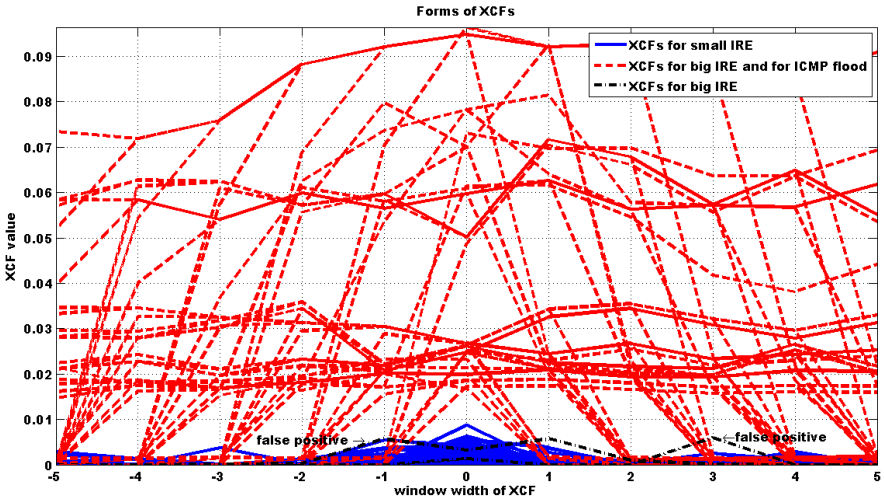

Figure 20. Normal XCFs from a training set and XCFs from a test set after starting an ICMP flood.

The 3D map of "bottleneck" neurons proves our suggestion: normal traffic is marked by points in the center of the map, and many points of anomalies are distributed on the margins of the 3D area (Figure 21).

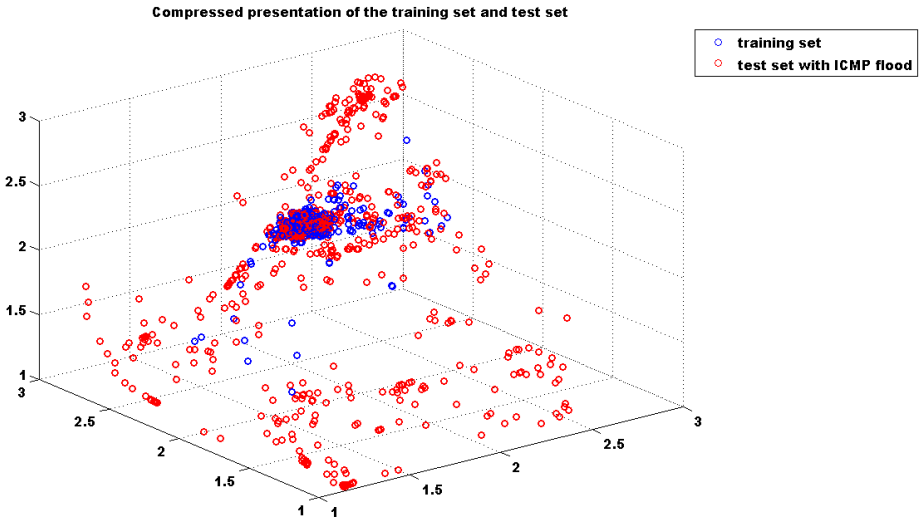

Figure 21. Compressed presentation of the training set and test set, which includes XCFs corresponding to the ICMP flood.

The error matrix for ICMP flood detection is represented in Table 5. FPR on the test was equal to 0.067%. The accuracy was equal to 98.5%.

Table 5. Error matrix for test set with ICMP flood.

	True	False
Positive	313	3
Negative	4444	70

4.2.5. HTTP Flood Detection

The latest experimental data confirm HTTP flood detection. The IRE graph is represented in Figure 22.

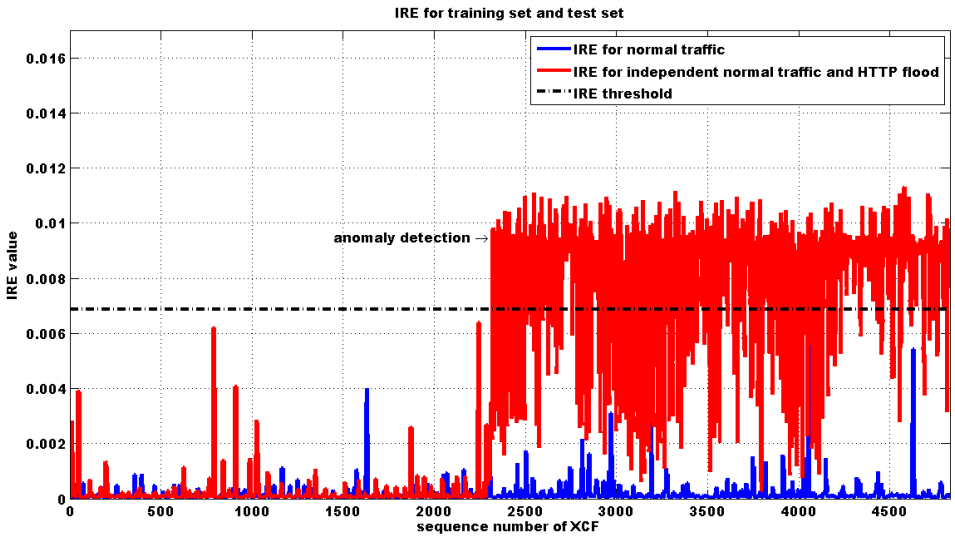

Figure 22. Instant reconstruction error for each element of the training set and test set containing XCFs corresponding to HTTP flood.

We can see no false positive detections before the start of the attack. The delay of the attack detection was equal to 8.3 s. The form and the amplitude of the anomalous XCFs greatly differed from normal traffic ones (Figure 23). Such visual differences usually lead to easy classification.

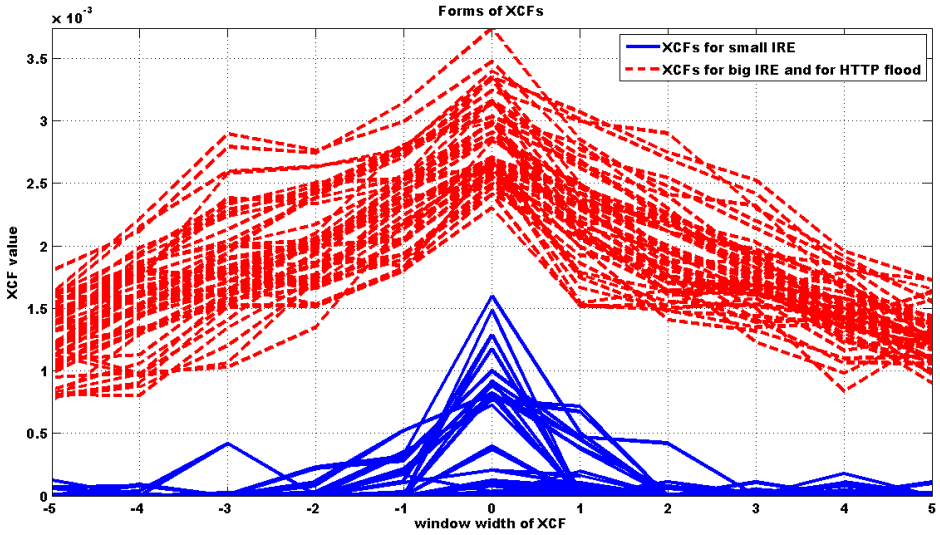

Figure 23. Normal XCFs from a training set and XCFs from a test set after starting an HTTP flood.

The 3D map of "bottleneck" neurons displays a dense cluster of points obviously related to the attack apart from normal traffic points (Figure 24). However, a part of the normal traffic area intersects with the part of the HTTP flood traffic.

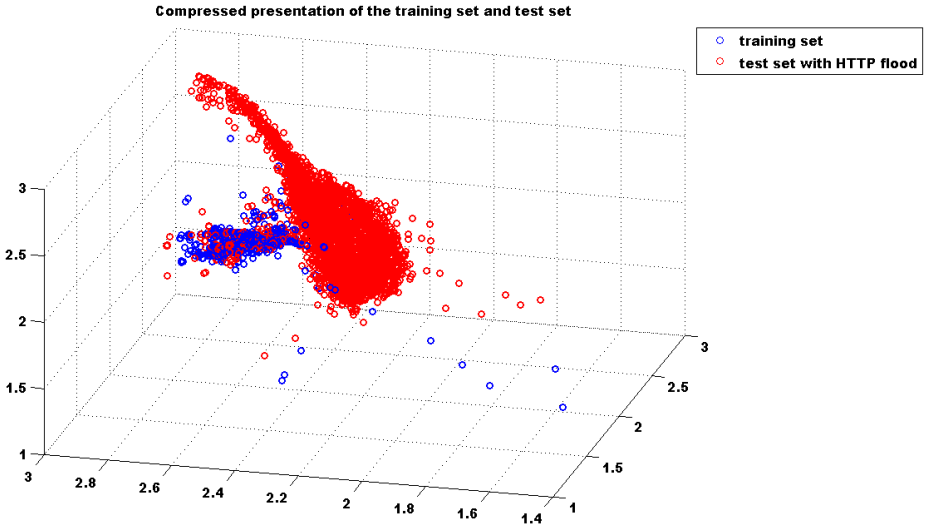

Figure 24. Compressed presentation of the training set and test set, which includes XCFs corresponding to the HTTP flood.

The error matrix for HTTP flood detection is represented in Table 6. FPR on the test was equal to 0%. The accuracy was equal to 91.4%.

Table 6. Error matrix for test set with HTTP flood.

	True	False
Positive	1833	0
Negative	2583	750

4.3. Summary of the Results

The presented results demonstrated the applicability of the proposed approach for the detection of several types of attacks without traffic content inspection. XCFs' difference for several types of normal and attacking traffic was illustrated both in original representation and in 3D mapping after compression by the autoencoder neural network. The key results of classification of all experiments are gathered together in the summary Table 7.

Table 7. Summary table of the results of the network attack detection algorithm.

Name of Attack	Detection Delay, s	FPR, %	Accuracy, %
SQL injection	0.6	0.104	99.8
SYN flood	19.2	0.036	61.9
TCP flood	28.3	0.052	87.1
UDP flood	4.7	0.049	89.8
ICMP flood	1.3	0.067	98.5
HTTP flood	8.3	0.000	91.4

We see that the developed method allows for detecting all considered types of attacks. In the course of the experiments, relatively few false positives were detected. In this paper, using a method based on an autoencoder, we managed to detect SQL injection attacks with high accuracy and without delays. The delay in detecting flood attacks can be explained by the fact that packets that are sent in large numbers to the server do not themselves carry malicious content and are initially regarded by the server and anomaly detector as normal. However, after some time, the flood attacks had an impact on the network stack, and as the bandwidth was filled and system resources exhausted, it became obvious that the traffic was not a legitimate request, but a DDoS attack. The proposed anomaly detection method detects a long-term anomalous server behavior caused by a flood attack, with an average delay of 12.3 s.

Table 8 shows the details of the experiment: server and operating system parameters.

Table 8. Server and operating system parameters.

Processor	Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz
CPUop-mode(s)	32-bit
System	Ubuntu 14.04.5 LTS i686
Kernel	4.4.0-31-generic
Memory	7.5 Gb
Disk	32 Gb

5. Discussion

The developed method did not engage any human expert knowledge to label training data or to construct heuristic rules, which can be used for labeling. On the contrary, it was absolutely unsupervised, and in the case study, expert knowledge was applied only for parameters' selection: neural network structure, width of XCF, and time sampling rate. The criterion to distinguish normal functioning from anomalies was inspired by the control system theory and implied the most basic

assumptions about the protected system: it should be a web server that processes the client's requests, and the dynamic character of this processing should change in case of a break in normal functioning.

The anomaly-based approach implemented in the method promises the capability to detect zero-day attacks. Really, there were no predefined attacks in the training data, so all anomalies were considered by the classification algorithm as unknown ones. An interesting side effect of the method is that it will detect all events of malfunctioning of the protected web server; it does not matter whether they were malicious events or not.

A significant disadvantage of the method is that all anomalies were of the same type, and to reveal the root cause of the anomaly event, a special effort should be made. Nevertheless, this is the usual behavior of the anomaly-based approach.

We suggest that the method has little computational complexity on the target system because all operations and data volumes involved in them are relatively modest for the modern desktop and server CPUs and even for multicore mobile and embedded CPUs. Our certainty is based on not using deep packet inspection techniques or semantic analysis of the traffic content at all. The series of the traffic intensity needed for training can be gathered on the fly and consume a small amount of memory in storage. There is no need to analyze the content of the network traffic, allowing use of the proposed method to protect network servers that use cryptographic protocols such as TLS.

Since the method calculated XCF between incoming and outgoing traffic intensity, as well as in the training phase and in the evaluation phase, it is hard to obtain appropriate data for comparative evaluation with other methods. For example, well-known datasets such as KDD'99 do not fit the purpose of comparative testing because they do not provide data for training of the developed method. That is why there is no comparison with other methods and approaches.

Moreover, the method requires training for a specific web server, and the trained classifier possibly will not work properly with other web servers or with the same web server with different hardware and software components. Such an unexpected and somewhat strange behavior specializes the application of the method to the systems with fixed functionality and the absence of changes in the configuration for long periods. A good subject for potential applications of the developed method is a hardware appliance such as a dedicated server with rare maintenance or an IoT device with an appropriate use case scenario.

A summary of the imaginable disadvantages of the developed method is listed below:

- multiple simultaneous requests can be classified as false positive if such a case were not included in the training dataset;
- complex dynamic processing of requests in distributed and multi-tier network services makes the server's response to the same request unpredictable and therefore may lead to false positives;
- all changes in server's hardware and software (including configuration settings) that may modify the performance will change the dynamics of the request-response and inevitably require training the classifier from scratch to prevent numerous false positives;
- simultaneous work of extraneous tasks with noticeable performance impact that do not correlate with processing requests (system administration activity, OS logging, antivirus, DB backup, scheduled tasks, etc.) make the classification quality worse;
- an attack giving a dynamic response of the server, similar to one of the normal ones, will be missed;
- the detected anomaly cannot be automatically identified in opposition to signature methods for attack detection.

The significant level of false negative errors for flood attacks clearly means that some events will be missed by the detection system. Let us discuss whether this is important or not. It should be noted that new events after the first anomaly detection do not seem very informative, because they characterize the same attack, and the first anomaly detection should attract the attention of the system administrator well enough. The following events are just duplicates of the first one. If some of the following events were missed, this should not change the resulting efficiency of the detection system.

In real-world IDSand threat intelligence systems duplicated detection events of the same attack are packed together, otherwise they would fill the dashboard with useless information. That is why we think that the delay of the flood attack detection is more important than the level of false negative errors after the first detection of the anomaly.

Our investigations allows us to highlight several proven and potential advantages:

- high quality of attack detection for a relatively simple web server;
- independence from expert labeling to train the classifier;
- no need for the classifier update until there are significant changes in the web server;
- capable of detecting several types of attacks and sensitive to zero-day attacks;
- simple algorithms and low resource consumption for lightweight implementation;
- applicable for all network protocols and capable of working with encrypted traffic.

6. Conclusions

In the paper, a new method of computer attack detection was established. The case study with vulnerable web server under several types of attacks, including SQL injection, was investigated and proved the efficiency of the method. Perspective features and visible limitations of the method suggest its main application for dedicated web servers with simple fixed functions and for IoT devices with server-like use cases.

Author Contributions: Conceptualization, V.E.; methodology, V.E.; software, A.G.; validation, A.G.; formal analysis, V.E. and A.G.; investigation, V.E. and A.G.; resources, V.E. and A.G.; data curation, V.E. and A.G.; writing, original draft preparation, V.E. and A.G.; writing, review and editing, V.E. and A.G.; visualization, A.G.; supervision, V.E.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

References

1. DDoS Attacks in Q3 2018. Available online: https://securelist.com/ddos-report-in-q3-2018/88617/ (accessed on 26 February 2019).
2. DDoS Attacks in Q1 2018. Available online: https://securelist.com/ddos-report-in-q1-2018/85373/ (accessed on 26 February 2019).
3. DDoS Attacks in Q4 2017. Available online: https://securelist.com/ddos-attacks-in-q4-2017/83729/ (accessed on 26 February 2019).
4. DDoS Attacks Increased 91% in 2017 Thanks to IoT. Available online: https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/ (accessed on 26 February 2019).
5. Necurs: World's Largest Botnet. Available online: https://resources.infosecinstitute.com/necurs-worlds-largest-botnet/ (accessed on 26 February 2019).
6. Predictions for 2018: Zero-Day Exploits Leaked from Security Agencies, Next-Level Ransomware. Available online: https://businessinsights.bitdefender.com/predictions-zero-day-agencies-ransomware (accessed on 26 February 2019).
7. Chithra, S.; George Dharma, E.; Raj, P. Overview of DDoS algorithms: A survey. *Int. J. Comput. Sci. Mobile Comput.* **2013**, *2*, 207–213.
8. Ternovoy, O.S.; Shatohin, A.S. Early detection of DDOS attacks by statistical methods, taking into seasonal variation. *Proc. TUSUR* **2012**, *1–2*, 104–107.
9. Haris, S.H.C.; Ahmad, R.B.; Ghani, M.A.H.A. Detecting TCP SYN flood attack based on anomaly detection. In Proceedings of the 2010 Second International Conference on Network Applications, Protocols and Services, Kedah, Malaysia, 22–23 September 2010; pp. 240–244.
10. Exploit Database. ImageMagick 6.9.3-9 / 7.0.1-0//Multiple Vulnerabilities (ImageTragick). Available online: https://www.exploit-db.com/exploits/39767/ (accessed on 26 February 2019).

11. The 10 Most Common Application Attacks in Action. Available online: https://securityintelligence.com/the-10-most-common-application-attacks-in-action/ (accessed on 26 February 2019).
12. Ray, D.; Ligatti, J. Defining code-injection attacks. *SIGPLAN Not.* **2012**, *47*, 179–190. [CrossRef]
13. Seacord, R.; Householder, A. A Structured Approach to Classifying Security Vulnerabilities. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Note CMU/SEI-2005-TN-003. 2005. Available online: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7377 (accessed on 26 February 2019).
14. Lowis, L.; Accorsi, R. On a Classification Approach for SOA Vulnerabilities. In Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference, Seattle, WA, USA, 20–24 July 2009; Volume 2, pp. 439–444.
15. Open Web Application Security Project. OWASP Top 10 Application Security Risks—2017. Available online: https://www.owasp.org/index.php/Top_10-2017_Top_10 (accessed on 26 February 2019).
16. Website Security Statistics Report. WhiteHat Security. Available online: https://info.whitehatsec.com/rs/whitehatsecurity/images/statsreport2014-20140410.pdf (accessed on 26 February 2019).
17. Web Applications Security Statistics Report. WhiteHat Security. Available online: https://info.whitehatsec.com/Website-Stats-Report-2016-LP.html (accessed on 26 February 2019).
18. Hodo, E.; Bellekens, X.; Hamilton, A.; Tachtatzis, C.; Shallow, R.A. Deep Networks Intrusion Detection System: A Taxonomy and Survey. 2017. Available online: https://info.whitehatsec.com/Website-Stats-Report-2016-LP.html (accessed on 26 February 2019).
19. Lee, I.; Jeong, S.; Yeo, S.; Moon, J. A novel method for SQL injection attack detection based on removing SQL query attribute values. *Math. Comput. Model.* **2012**, *55*, 58–68. [CrossRef]
20. AlNabulsi, H.; Islam, M.R.; Mamun, Q. Detecting SQL injection attacks using SNORT IDS. In Proceedings of the Asia-Pacific World Congress on Computer Science and Engineering, Nadi, Fiji, 4–5 November 2014.[CrossRef]
21. Nithya, V.; Regan, R.; Vijayaraghavan, J. A Survey on SQL Injection attacks, their Detection and Prevention Techniques. *Int. J. Eng. Comput. Sci.* **2013**, *2*, 886–905.
22. Kindy, D.A.; Pathan, A.-S.K. A Detailed Survey on various aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks and Remedies. *Int. J. Commun. Netw. Inf. Secur.* **2013**, *5*, 80–92.
23. Elshazly, K.; Fouad, Y.; Saleh, M.; Sewisy, A. A Survey of SQL Injection Attack Detection and Prevention. *J. Comput. Commun.* **2014**, *2*, 1–9. [CrossRef]
24. Moosa, A. Artificial Neural Network based Web Application Firewall for SQL Injection. *Int. J. Comput. Electr. Autom. Control Inf. Eng.* **2014**, *4*, 510–619.
25. Valeur, F.; Mutz, D.; Vigna, G. A learning-based approach to the detection of SQL attacks. In Proceedings of the Second International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'05), Vienna, Austria, 7–8 July 2005; Volume 4, pp. 123–140.
26. Torrano-Gimenez, C.; Perez-Villegas, A.; Alvarez, G. An Anomaly-Based Approach for Intrusion Detection in Web Traffic. *J. Inf. Assur. Secur.* **2010**, *5*, 446–454.
27. Elisa, B.; Ashish, K.; James, E. Profiling Database Application to Detect SQL Injection Attacks. In Proceedings of the IEEE International Performance, Computing, and Communications Conference, New Orleans, LA, USA, 11–13 April 2007; Volume 5, pp. 449–458. [CrossRef]
28. Choras, M.; Kozik, R.; Puchalski, D.; Holubowicz, W. Correlation Approach for SQL Injection Attacks Detection. In *CISIS/ICEUTE/SOCO Special Sessions*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 189, pp. 177–185.
29. Skaruz, J.; Seredyński, F. Recurrent neural networks towards detection of SQL attacks. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium, Rome, Italy, 26–30 March 2007, pp. 1–8. [CrossRef]
30. Sheykhkanloo, N.M. SQL-IDS: Evaluation of SQLi attack detection and classification based on machine learning techniques. In *SIN '15 Proceedings of the 8th International Conference on Security of Information and Networks*; ACM: New York, NY, USA, 2015; pp. 258–266. [CrossRef]
31. Dong, Y.; Zhang, Y. Adaptively Detecting Malicious Queries in Web Attacks. 2017. Available online: https://arxiv.org/pdf/1701.07774.pdf (accessed on 26 February 2019).

32. Eliseev, V.; Shabalin, Y. Dynamic response recognition by neural network to detect network host anomaly activity. In *SIN '15 Proceedings of the 8th International Conference on Security of Information and Networks*; ACM: New York, NY, USA, 2015; pp. 246–249.
33. Eliseev, V.; Gurina, A. Algorithms for network server anomaly behavior detection without traffic content inspection. In *SIN '16 Proceedings of the 9th International Conference on Security of Information and Networks*; ACM: New York, NY, USA, 2016; pp. 67–71.
34. Jain, D.; Choudhary, N. An Automatic Detection System for SQL Injection. *Int. J. Comput. Appl.* **2015**, *126*, 16–21. [CrossRef]
35. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 1–58. [CrossRef]
36. Hodge, V.J.; Austin, J. A survey of outlier detection methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
37. Packet Storm. MyBB Version 1.8.6 Suffers from a Remote SQL Injection Vulnerability. Available online: https://packetstormsecurity.com/files/138744/MyBB-1.8.6-SQL-Injection.html (accessed on 26 February 2019).