DETECTION METHOD FOR “DENIAL OF SERVICE” ATTACKS ON WEB APPLICATIONS

Azizbek Ravshonbek ugli Ibrohimov
State Unitary Enterprise "Centre of cyber security" Address: 10A, Kirk-Kiz street, Tashkent city, Republic of Uzbekistan E-mail: ibrohimov_az@mail.ru, Phone:+998-97-701-05-43; ibrohimov_az@mail.ru

Follow this and additional works at: https://uzjournals.edu.uz/ijctcm
Part of the Complex Fluids Commons, Controls and Control Theory Commons, Industrial Technology Commons, and the Process Control and Systems Commons

Recommended Citation
Ibrohimov, Azizbek Ravshonbek ugli (2020) "DETECTION METHOD FOR "DENIAL OF SERVICE" ATTACKS ON WEB APPLICATIONS," Chemical Technology, Control and Management: Vol. 2020 : Iss. 4 , Article 12. Available at: https://uzjournals.edu.uz/ijctcm/vol2020/iss4/12

This Article is brought to you for free and open access by 2030 Uzbekistan Research Online. It has been accepted for inclusion in Chemical Technology, Control and Management by an authorized editor of 2030 Uzbekistan Research Online. For more information, please contact sh.erkinov@edu.uz.
Detection Method for "Denial of Service" Attacks on Web Applications

Ibrohimov Azizbek Ravshonbek ugli¹

State Unitary Enterprise "Centre of cyber security"
Address: 10A, Kirk-Kiz street, Tashkent city, Republic of Uzbekistan
E-mail: ibrohimov_az@mail.ru, Phone:+998-97-701-05-43;

Abstract: A method for detecting denial-of-service attacks on web applications based on the use of a multi-layer perceptron is considered. The main issues related to the use of the neural network are described: the process of collecting and preparing statistical data, the features of forming indicators that characterize the state of the web application, as well as the issues of training and evaluating the quality of the neural network.

Keywords: attack detection, denial of service, DoS, web application.

Introduction

For a relatively short period on the Internet, a large number of web-applications have appeared that provide users with various information or provide services. Traditional applications are gradually being transferred to the Internet, which makes it possible to conveniently interact with them using various devices and almost anywhere. In this regard, there is a constant increase in the number of users of web applications and there is a problem of ensuring the reliable functioning of such systems.

Modern web applications are quite complex software that can process large amounts of data and generate responses to user requests. The complexity of the functioning of web applications leads to high loads on hardware resources, and a large number of users make such applications an interesting target for computer attacks. Attacks on web applications can be carried out for various purposes, for example, to steal personal information or information about users' credit cards [1].

The purpose of denial of service attacks is to completely or partially disable a web application, which deprives users of the opportunity to fully use it. A "denial of service” of a web application can be achieved using attacks aimed at its infrastructure, or it can be carried out at the application level, while interacting with the application via the HTTP protocol [2].
Attacks aimed at the web application infrastructure are well understood; there are a large number of methods for detecting such attacks:
- deviation of data parameters;
- change in the probabilistic data parameters;
- discovery using data mining;
- restoration of the path by signatures;
- packet marking method;
- generation of service packages, etc.

These methods are poorly applicable for the analysis of attacks on web applications conducted at the application level. This is because the various components of modern web applications are very different in frequency of use and consumption of hardware resources, as well as the fact that modern web applications are constantly increasing the number of pages available for viewing.

At the application level to the “denial of service” web-applications can cause operation of different vulnerabilities in web-application and exceeding the allowable number of requests that can be simultaneously continuously processed web-application.

An increase in the number of requests that are simultaneously being processed in a web application can occur:
- by increasing the intensity of the flow of requests;
- during attacks using vulnerabilities in the HTTP protocol (for example, slow HTTP POST).

The work deals with denial of service attacks that arise due to an increase in the flow of incoming requests caused by simulating user actions by automatic programs (bots). This type of attack in 2013 amounted to 86% of the total number of denial of service attacks on web applications conducted at the application level [3-4].

Currently, there are no generally accepted classifications and effective methods for detecting this type of attack. At the application level, a denial of service attack detection method based on an estimate of the probability of loss of applications is known, but it is applicable only in the case of low active attacks. In this regard, the task of creating I method to detect attacks such as “denial of service” on the web-application operating at the application level, is important.

**General scheme of a denial of service attack detection method for web applications**

There are two main approaches to detecting attacks: detecting anomalies and detecting abuses. When anomalies are detected, a profile of the normal state of the system is compiled, a deviation of the current state of the system from the profile is considered an anomaly and a conclusion is made about a possible attack. When abuses are detected, the task is to describe the profiles of the system in a state of various attacks in order to further recognize different types of attacking influences.

To construct the method, an approach is used associated with the detection of abuses using neural networks. This approach allows you to flexibly adapt the attack detection tool to the specific features of a specific web application (to automatically configure acceptable parameter values); it is successfully used to solve a number of related tasks and detect some classes of attacks.

As a neural network used, it is advisable to use a network of a multilayer perceptron due to the following features:
- the effectiveness of solving approximation problems;
- the ability to work with nonlinearly separable input indicators;
- ease of implementation;
- the high speed of the network, this makes it possible to use this type of neural network to detect denial of service attacks in real time;

The general scheme of the denial of service attack detection method is shown in Fig. 1.

The method is as follows:
1. prepare statistical information about user behavior;
2. generate statistical information about various types of attacking effects leading to a “denial of service”;
3. in the considered time period (not less than a week) calculate the values of indicators (and corresponding indicators of statistical structures);
4. assess the applicability of indicators and form a set of indicators that are most suitable for determining various types of attacks;
5. determine the best parameters of the architecture of neural networks for various types of attacks and the quality of detection of attacks;
6. group various types of attacks into classes of attacking forces in order to reduce the number of neural networks used;
7. to train neural networks;
8. use neural networks to detect denial of service attacks
9. if necessary, retrain the neural networks based on new statistics.

Statistics collection
To describe the user's request for a web application, the following data vector format has been developed:

\[(Time_i; SessionID_i; IsSessionStart_i; URL_i; Parameters_i; IP_i; Referer_i; UserAgent_i; Latency_i; DocSize_i; Memory_i; CpuTime_i)\]

- \(i\) - serial number of the request to the web application;
- \(Time_i\) - request time, expressed in timestamp format;
- \(SessionID_i\) – session identifier;
- \(IsSessionStart_i\) - indicator of the beginning of the session;
- \(URL_i\) – URL address of the request;
- \(Parameters_i\) - GET- request parameters;
- \(IP_i\) - IP address from which the request was made;
- \(Referer_i\) - address of the previous visited page;
- \(UserAgent_i\) - identification line of the client program;
- \(Latency_i\) - server response time when processing this request;
- \(DocSize_i\) – the amount of data that was downloaded by the user in response to a request;
- \(Memory_i\) - the amount of server RAM used to process the request;
- \(CpuTime_i\) - CPU time spent on the request[4].

The procedure for preparing statistics on user requests is presented in Fig. 2.

After collecting statistics in the proposed format, it is necessary to convert URL requests to a canonical form using the rules for converting addresses, deleting and setting the order of query parameters.

We introduce the following notation:
- URL string URL;
- Script URL string URL without query string (the initial fragment of the URL to the character “?”);
- \(n\) - the maximum possible number of parameters in the request;
- \(s\) - the number of all possible parameters in the web-system;
- \{\(parameter_1\), ..., \(parameter_s\)\} - the set of all possible parameters in the web-system;
- \(parameter_i^j\) - some of the first \(j\)-parameter in the request \(j \in \{1, ..., n\}, i \in \{1, ..., s\}\);
- \(parameter = parameter_i^j \& \& \& \& \& parameter_i^k\) - parameter string containing parameters;
- \(URLFilter (ScriptURL)\) - a function that describes the algorithm for generating the path to the resource on the server that does not contain special constructions;
- \(ParametersSequence (parameter_i^1, ..., parameter_i^k)\) - a function describing the application of the rules to the sequence parameter forming a sequence parameter.
parameter_{i_1}, \ldots, parameter_{i_k}, i_j \in \{1, \ldots, s\}, j \in \{1, \ldots, k\}, the function result is independent of the order of the arguments;

- ParameterPresence (parameter_{i}) - a function describing the application of the rules to delete some parameter settings parameter_{i}, i \in \{1, \ldots, s\};
- operation of “gluing” the lines.

The algorithm for generating the URLFilter (ScriptURL) path can be described as follows: remove from the portion of the ScriptURL line next to the domain name and port in the following sequence:

1. constructions of the form “//” from the string ScriptURL;
2. constructions of the form “/.” From the string ScriptURL;
3. constructions of the form “/text/ ../”, where text is an arbitrary string that does not contain the characters “/”.

Figure 1. The general scheme of the operation of the denial of service attack detection method for web applications.

Collecting and preparing static data about user requests

Generation of attack streams

Calculation of indicators

Has neural network training been performed before

Determine the number of the neural network, vectors of indicators and parameters of the neural network architecture

Create vectors of the highest quality indicators for each attack

Determine the parameters of the neural network architecture

Evaluate the quality of attack detection

Determine the number of neural networks used and their corresponding vectors of indicators

Detecting denial of service attacks

Neural network training

Collect primary statistics in vectors of the developed format

Convert a URL request to a canonical form

The definition of a canonical representation of the ID string

Associate a query with some logical query class

Correlate a query with a certain load class of queries and define the load parameters of the query

To form the auxiliary statistical structure

Figure 2. The procedure for preparing statistics.
In the notation introduced, URL can be represented as
\[ URL = \text{ScriptURL}:?;\text{parameters}; \]

The canonical request form for some URL request is a request of the following form:
\[ \text{CanonicalURL}(URL) = \text{URLFilter}(\text{ScriptURL}):?;\]
\[ \text{ParametersSequence}\left(\text{ParameterPresence}(\text{parameter}^1),\text{ParameterPresence}(\text{parameter}^k)\right). \]

A canonical representation of the identity string is necessary to parse the User-Agent string. Analysis of the source data of the User-Agent is difficult because there are a large number of browsers, and these lines can vary significantly even with different versions of the same browser. Therefore, it is necessary to select the relevant parameters from the User-Agent strings and bring their string record to a single form. This process is similar to obtaining the canonical form of URL query strings: it is necessary to create rules for deleting and ordering the parameters of the User-Agent string [5-6].

In the future, it is necessary to break down all the queries according to the logical and load type of pages. The logical assumptions are based on the following assumptions:
- users tend to perform certain sequences of actions that have a similar semantic load: for example, viewing news usually involves viewing a news feed, alternating with viewing pages of specific news;
- similar actions of users (for example, viewing news or a profile) lead to the appearance of similar loads on the server.

The breakdown by the logical type of pages is carried out in two stages:
- the allocation of large logical sections on many pages of a web application (for example, forum, chat, news feed, etc.);
- allocation of various typical user actions within logical sections (for example, for a forum: reading a list of forums, reading a list of topics, reading a topic, etc.).

When clustering by load indicators, requests are placed in one class, creating similar loads on different resources on the server. Clustering by load characteristics is not informative for describing user behavior, since load-similar requests can be completely different, not related to the actions of the user of the web application. Therefore, it is advisable to use a combined approach, in which the queries are first divided by logical type, and then queries from one logical class are once again divided into classes according to load indicators[7].

To divide by load characteristics, you can use the algorithm for constructing a minimum spanning tree.

To form indicators taking into account the dependencies between user requests, it is necessary to form additional statistical structures that describe various properties of user behavior. As part of the study, the author identified the following areas of dependency analysis:
- the dependence of the appearance of the request on the previous request;
- the dependence of the appearance of the request on its location in user sessions;
- the dependence of the appearance of the request on the sequence of previous requests.

To account for each of these dependencies, it is proposed to use the following statistical structures:
- matrix of dependencies from previous queries;
- matrix of dependencies on the request number in the session;
- forest of session trees.

In the process of research, a model for the conduct of web application users and a classification of the main types of automatic programs, the actions of which can lead to a “denial of service,” are considered. The following main classes of automatic programs are distinguished:
- search;
- unloading the contents of the pages of a web application;
- web applications interacting with search results;
- associated with specific HTML forms;
- collecting information about a web application; attackers.

Based on this classification, query statistics can be generated for various types of attacking activities. The classification and imitation of various classes of automatic programs will be considered in more detail in other works of the author [8].

**Training, determining the parameters of architecture and assessing the quality of the neural network**

The multilayer perceptron is trained using the back propagation algorithm of the error. For training, you can use the following stopping criterion:
- the learning error stops decreasing (or the intensity of decreasing the error reaches a threshold value);
- the error reaches a certain threshold value;
- the number of training iterations reaches a threshold value (in this case, the correct training of a multilayer perceptron is not guaranteed).

To determine the parameters of architecture, it is necessary to conduct a sequence of tests with an increase in the number of layers of neurons and neurons in one layer. The test results determine the parameters when the minimum [6]:
- the average error of learning at the time of stopping learning;
- the average number of educational eras.

To assess the quality of the neural network, the following indicators are used:
- share of correct answers;
- the proportion of errors of the first kind;
- the proportion of errors of the second kind;
- share of uncertain answers.

**The formation of indicator vectors for training a neural network**

At the first stage, for the formation of indicator vectors, it is necessary to determine the most suitable indicators for determining various types of attacking effects. For this, a methodology for assessing the applicability of indicators is used, based on the calculation of the following characteristics:
- amplitudes (variation of indicator values);
- cyclicality (the degree of difference between the values of the indicator on different days of the week);
- differentiation (the degree of difference in the average values of the indicator on the statistics of normal user behavior and attack statistics).

Based on the developed methodology, the most suitable indicators for the detection of various types of attacking effects are selected [7].

With this approach, different neural networks are used to detect different types of attacking influences. To speed up the work of the method, a technique was proposed for combining various types of attacking actions into attack classes, which made it possible to use a common vector of indicators and one neural network for attacking actions that are in the same attack class.

Details of the methodology for assessing the quality of an indicator and forming vectors of indicators will be considered in other works of the author.

**Conclusion**

A method is considered that allows the detection of denial of service attacks on web applications at the application level. This method has been successfully tested and is planned to be used in the construction and implementation of systems for detecting attacks on web applications.
The described approach to constructing a variety of indicators allows us to ensure the efficiency of the method, taking into account changes in web application attendance over time, on different days of the week and taking into account natural changes in the popularity of web applications.

Methods for evaluating indicators and forming vectors of indicators and attack classes can improve the performance of the method by reducing the number of neural networks and indicator vectors used. This allows you to use the method in real time.

References: