Journal of Computer and Communications, 2015, 3, 68-82
Published Online September 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.39008

Approach Development Accelerate of Process Special Traffic Filtering

Karimov Madjit Malikovich, Gulomov Sherzod Rajaboevich*, Yusupov Bakhodir Karomatovich

Department of Information Security, Faculty of Computer Engineering, Tashkent University of Information Technologies, Tashkent, Uzbekistan
Email: *sherzod.gulomov@rambler.ru, sherhisor30@gmail.com

Received 13 August 2015; accepted 18 September 2015; published 21 September 2015

Abstract

This article is dedicated to the analysis list of a set of rules to traffic filtering, which is a multi-dimensional structure, where each dimension is a set of networking field or the field of action, measuring the cost of the rules to traffic filtering on computer networks, allowing to determine the difference between definition of the rules and the control of the packet fields. Furthermore, the article were considered a hierarchical model to optimize traffic filtering, which reduces the overhead traffic filtering rules and provides the semantic integrity of the original set of rules to traffic filtering. The hierarchical structure of the design and optimization of traffic filtering was researched. And also was developed the hierarchical approach to optimize traffic filtering for reducing set of rules traffic filtering. Analyzed the algorithm optimal solutions and algorithm of random search filters that, allowing you to find the shortest way to a set of rules to traffic filtering. Moreover, in this article was presented the effectiveness evaluation of the process accelerating traffic filtering proposed by HAOTF.

Keywords

Traffic Filtering, Hierarchical Structure, Model, HAOTF, Heuristic Method

1. Introduction

Integrated efficiency, reliability and availability of filtering software are crucial in ensuring the security and administration, especially when the network is under attack and threat. These problems require new designs, architecture and algorithms to accelerate traffic filtering. With the dynamic change of the load on the network to-

*Corresponding author.

How to cite this paper: Malikovich, K.M., Rajaboevich, G.S. and Karomatovich, Y.B. (2015) Approach Development Accelerate of Process Special Traffic Filtering. *Journal of Computer and Communications*, **3**, 68-82.
http://dx.doi.org/10.4236/jcc.2015.39008

pology and bandwidth demand, traffic filtering becomes a vulnerable point. All these factors create a demand for the most efficient, high-performance, affordable and reliable traffic filtering in Firewalls. Nowadays, Firewall enforces security policy with a set of multi-dimensional traffic filtering rules. Information security policy in networks is a very important task to speed up traffic filtering. In addition, with the increase in capacity of existing networks for processing and forwarding traffic at extremely high speed, Firewalls are very limited resources. Thus, the main objective is to overcome the drawbacks of the current traffic filtering and enhance their ability to dynamic changes in the load and network topology, especially in attack. The main objective of these rules to maintain the semantic integrity policy established at each level of hierarchical models.

2. List a Set of Traffic Filtering Rules

Security rule is a multidimensional structure, where each dimension is a set of network fields or field of action. A set of rules defines a security policy that traffic filtering. Rules define by a set of IP-source address and a set of destination IP-address a variety of service types and fields of action. The type of service usually includes a main type of protocol and port number. Field of action can either accept or refuse or direct. Take action allows access packet in a secure domain. Disclaimer action causes the packet, in violation of security policy. Formal rules R can be represented as: $R = \left[\Phi^1, \Phi^2, \cdots, \Phi^k; \Sigma\right]$, where Φ is the network field and the field of action. An example of a typical rule may be the following sequence:

$$\begin{aligned} &< src = \{s_1, s_2, \cdots, s_n\}; dst = \{d_1, d_2, \cdots, d_m\}; srv = \{\varsigma_1, \varsigma_2, \cdots, \varsigma_l\} \\ &\text{action} = \{\text{drop / accept / forward}\} > \end{aligned} \tag{1}$$

where s_n represents IP-address of the source, d_m destination IP-address and ς_l type of service. In the list of a set of rules to filter traffic, rules describing network security policies, formed as a “priority list”. Priority rules, also referred to as the rules of rank, based on its position within the list [1]. Previously derived rules are ranked higher than those which were later. The set of rules may be transformed by creating all possible permutations of the fields in a rule. Formal representation of a set:

$$\begin{aligned} &< src = s_n; dst = d_m; srv = \varsigma_l \\ &\text{action} = \{\text{drop / accept / forward}\} > \end{aligned} \tag{2}$$

List a set of rules to traffic filtering work by studying sets in sequential order. For each packet, the first matching set determines the action of traffic filtering. If the size of the list is small, then the list of a set of rules to traffic filtering is considered good. Since the size of the list are scaled up to one million sets, management and optimization of traffic filtering policy is a challenge.

2.1. Measuring the Cost of Traffic Filtering Rules

The main factor, which influences the performance of the filtering action of traffic are overhead data for packet inspection. Measurement computation is performed according to the rule, and can be readily applied to a set of rules within. To determine the overheads incurred by the traffic filtering treatment rules and application security policy, there are two specific measurements.

The first dimension, referred to as the size of the rules. The size of the rule in terms of the number of bits necessary to determine unambiguously as the difference between the definition of rules and control of the field of the packet. Assumptions underlying the measurement of the size of the rules related to the fact that the complexity of coordination of operations proportional to the size rules. Formally, this rule r, rules size (r) can be defined as:

$$\text{Rulesize}(r) = \left\{\sum S_p, D_p \left\{\alpha_1 \times \left\|s_p\right\| + \alpha_2 \times \left\|d_p\right\|\right\} + \beta \times N_s \times \left(\left\|Pr_r\right\| + \left\|Pr_o\right\|\right)\right\} \tag{3}$$

where α_1, α_2, и β are main parameters, S_p, и D_p respectively, a set of source and destination prefixes that occur in the rule definition, s_p, и d_p respectively are bit representation of the source and destination prefixes, N_s the number of services defined in rule, и Pr_r, и Po_r, are bit representation of the protocol and ports.

The second dimension used in the experiment of current costs at a given set of rules. These costs depend on the size and positioning of the rules on how often the rule is called traffic filtering. Formally, given a set of rules

$r_1, r_2, \cdots, r_k$, costs of this rule r_i, is defined as follows:

Expenditure(r_i) = number of retries$(r_i) \times \sum \forall r_k \in Pr_i \| r_k \|$, where Pr_i a set r_i -espredecessors on the basis of a list of a set of rules. Using the above measurements in order to optimize the reduction in size and therefore the rule set, the processing time rule set. This, in turn, reduces the overall operational costs of traffic filtering.

2.2. Criterion Conversion of Traffic Filtering

Traffic filtering defines its security policy with a set of policies or security rules. These security policies are defined formally as follows:

Let F is a list of the original set of rules to filter traffic.

Let $T(F)$ is converted traffic filtering, which retains the properties and a set of rules F. Is determined by the cost function, $C(f)$, which is the average overhead costs set of rules to filter traffic.

$T(F)$ is acceptable conversion F:

- $T(F)$ retains the properties and rules F (semantic integrity of the property);
- $C(T(F)) \le C(F)$ (property costs).

2.3. Transformation Approach of Traffic Filtering

Converting traffic filtering is achieved in the process of dividing the original set list of rules to filter traffic in the rule group subsets. A set of rules for filtering traffic sorted based on the traffic characteristics before conversion initiated. Getting the right subset retains the properties and rules of "semantic integrity" of the original list, a set of rules. In order to formulate a formal:

Let $S(F)$ is an approach conversion filter traffic based on a set of rule F.

$S(F)$ is acceptable conversion F, if:

- $S(F)$ retains the properties and rules *F* (semantic integrity properties);
- costs (*S*(*F*)) ≤ performance F (property costs).

Semantic integrity of the property:

Let there be at least one rule r in $S(F)$ such that the action r network p packet is different than F on p. This means that the rule of traffic filtering installed F on this network traffic is different from the actions taken by the new transformed representation $S(F)$.

Thus, the semantics of the original set of rules to traffic filtering is not equivalent to transformation a set of rules to traffic filtering.

Rules based on a list of a set of rules to traffic filtering is scanned in a sequential order and divided into subsets of rules based on traffic characteristics. The order, in which the rules are divided into subsets of rules, does not add a new rule, or cause the removal of any rules. This means that no new rules are created or removed from the conversion process. This proves that the rule r in $S(F)$ must belong to some level of priority in the set of rule F, which implies a contradiction.

Consequently, there are no rules to convert a set of rules to traffic filtering $S(F)$, which have different activities, unlike those that exist in the original set of rules F. This proves that the semantic properties and rules in the original set of rules stored after conversion.

Property costs:

Let the overhead costs of the original set of rules to filter traffic and a set of transformation rules will be $C(F)$ and $C(S(F))$, respectively.

It is suggested that $C(F) > C(S(F))$.

For this assumption to be true, there exists at least one packet p, that it complies with the rule r a set of rules to traffic filtering, where compliance costs in the transformed set of rules to traffic filtering higher than the corresponding cost in the original set of rules.

Let overheads processing network packet p, that matches the rule, r a set of rules F traffic filtering is represented as x and the relevant rules r transformed traffic filtering $S(F)$ there will be y, where $y > x$.

Thanks to the list of rules to traffic filtering capacity y would be more than x, if the series r in $S(F)$ lower than the series of r in F.

Since both sets of rules to traffic filtering characteristics are sorted traffic (inbound), there is a rule r' in the rule subset $S(F)$ traffic filtering, which has a higher number than the r of the low number of the original set

of rules F.

So, all the rules in F sorted in accordance with the calculation information and no new rules created or deleted as a result of the conversion process.

Thus, it have proved that the costs set of rules $S(F)$ traffic filtering is ≤ on the cost of the original set of rule F. In the worst case, the length $S(F)$ will be equal to the length F, meaning $C(F)=S(F)$.

3. The Hierarchical Structure of the Optimization of Traffic Filtering

Data structure. Multilevel data structure consists of subsets of rules and their respective filters. To process typically uses a hierarchical data structure, in which a deep level of the hierarchy contains a subset of the rule and intermediate levels contain filters that cover the rules that are included in these subsets. Balances the hierarchical structure in order to reduce the length of the deepest subset rules, therefore, it is important to have the desire to achieve the maximum reduction of the costs of processing rules [2]. The data structure must ensure a reduction of overhead costs, semantic integrity and safety of the initial set of rules. It should be noted that overhead costs are determined by the rule subset. Furthermore, the data structure must be designed so that the process of re-balancing, in response to changes in traffic can be achieved at minimal cost.

Semantic integrity of the original set of rules can be achieved, and set during the reign of the separation process by calculating the filter that is clearly and fully complies with the rule subsets.

Moreover, after the packet processing must follow the same semantic instructions filters produced in the separation process. If the rules are divided and re-loaded, then to optimize the overhead costs, the process of repeating the original semantics of the rules must be achieved with lower costs.

3.1. The Hierarchical Development Model

The hierarchical model defines approach to designing networks and includes three logical layers (see **Figure 1**):

- corelayer;
- distribution layer;
- access layer.

Each layer has its own function. Three layers do not necessarily imply the presence of three different devices. To draw an analogy with the hierarchical model of OSI, it is a separate protocol does not always correspond to one of seven levels. Sometimes protocol corresponds to more than one level of OSI model, and sometimes several protocols implemented within the same layer. And in constructing hierarchical networks, at one layer can be multiple devices or a single device that performs all functions defined on two adjacent layers.

The core layer is at the top of the hierarchy and is responsible for the safe and rapid transfer of large amounts

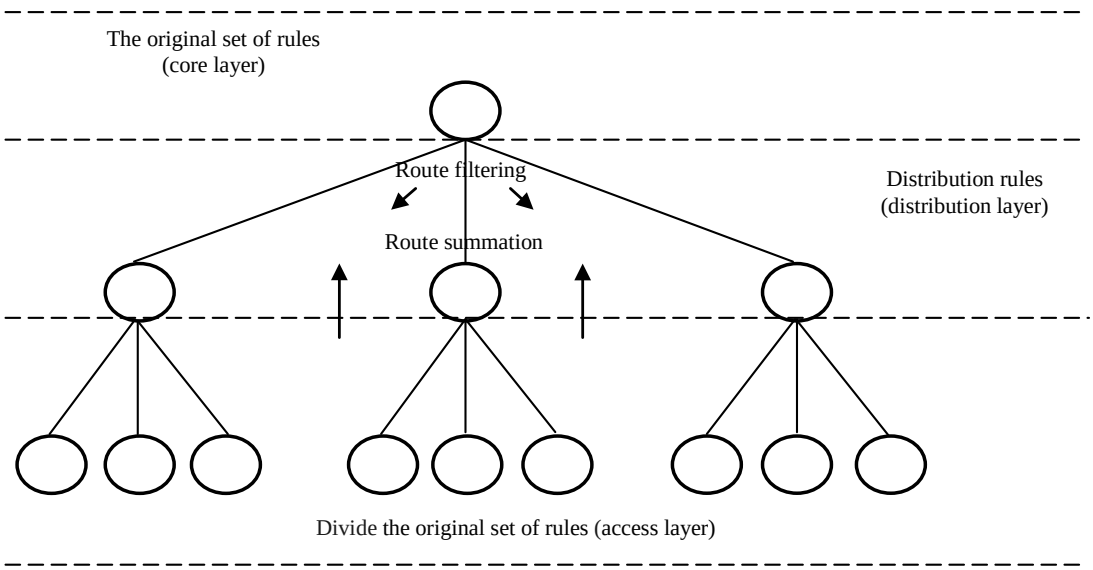

Figure 1. The layers of hierarchical structure.

of data. Traffic transmitted through the core, is common to most users. Most user data is handled at the distribution layer, which, if necessary, forwards the request to the core. For core layer its importance has resiliency as a failure at this layer could result in loss of connectivity between the levels of the distribution network.

The core layer for the following functions:

- shared bandwidth;
- switching bandwidth;
- MAC-level filtering;
- microsegmentation.

The distribution layer, which is sometimes called the working group layer, is the link between the layers of access and core. Depending on the implementation method, the distribution layer can perform the following functions:

- verification of source and destination addresses;
- check the input and output ports;
- filter routes to hide the internal network;
- providing routing, quality of service and network security;
- canal aggregation;
- transition from one technology to another (e.g. 100Base-TX to 1000Base-T);
- aggregated bandwidth low-bandwidth access to the high-speed backbone links.

The access layer controls user access and working groups to resources united network. The main objective access layer is to create points of exit/enter users to the network. This layer performs the following functions:

- continued (from the level of the distribution), access control and network policy;
- create separate collision domains (segmentation);
- connecting workgroups to the level of distribution;
- access layer uses switched LAN.

And so, the core layer is required in order to produce a separation of the original set of rules to a set of disjoint subsets of rules. This process involves taking a set of access rules and rules on the division of its subsets, each of which is determined by the filter. Each filter consists of a series of disjoint sets that fully cover its corresponding rule subset (see **Figure 1**).

At distribution layer, rule is distributed in such a way that high standards of cost have been moved to the top of a set of rules. As previously stated, the rules are based on cost rules size. The amount of traffic passing through this rule indicates the number of his matches. At this stage, costs are reduced by the general rules of traffic processing.

The access layer is taken as the input of the original list a set of rules based on the list, and produces an optimized set of rules. This is the best set of rules consists of fully overlapping and concise rules, where all exceptions to the rules and dependencies are removed. The fact that the rules in the rule set intersect to change the order of the rules and dividing them into subsets of rules without breaking the semantics of the original set of rules at the kernel level.

3.2. Development of a Hierarchical Approach to Optimize Traffic Filtering

To develop a hierarchical approach to optimize traffic filtering (HAOTF) uses a hierarchical structure to the division of the initial set of rules and regulations mutually exclusive subsets to reduce the overhead costs of traffic filtering.

In the original set of rules, HAOTF uses a multistep process in which it originally shared the original rule set into two subsets. Then recursively passes the access layer to subsets produced in the previous step generate the next layer of the hierarchy. Formation to access layer continues as long as the total cost of processing does not eclipse the benefit gained by continuing the current access layer subsets.

Then, in this case, the access layer is terminated and the previous layer is selected as optimum as possible the depth of hierarchical structure. Efficiency distribution layer depends strongly of a subset of the rules used at different layers of the hierarchy.

In solution the complex markup, HAOTF uses an iterative approach for the separation of the original set of rules and make a multilevel hierarchy, economically balanced subsets of rules.

Initially, the set of rules is divided into two subsets and the filters are designed and cover rules contained in

each subset.

In HAOTF the time of packet processing begins at the root of the hierarchical structure. Packets subsequently are sent to the other layers of the hierarchy for further processing. Packet processing is terminated, if the difference between the attributes of the packet, as defined in the traffic filtering, the security policy is triggered. In this case, the action is determined by compliance with the relevant filter rules. The default action may be taken in either case the packet is sent to the destination, or rejected, and in this case, the packet is dropped. Given a large set of rules, the goal HAOTF is to partition the set of mutually exclusive subsets *K*. Each subset is associated with a unique filter which is an extension of associated policies subsets. HAOTF driven by three major design goals:

- to reduce the cost of processing a set of rules determined by the average processing time packet, which he takes on;
- to save the semantics of the original set of rules;
- to maintain an optimal set of rules as templates for traffic and changes the set of rules.

It should be noted that in the general form of *K* may be reduced to the "clustering". **Figure 2** describes action rules *N* in subset *K*.

The resulting subsets, along with according filters, form a first layer of the hierarchy. This iterative process continues until, while further division of the subsets at the current level of the hierarchy will not be effective. It should be noted that this cost also includes certain expenses filters. The structure of the basic steps described on HAOTF in **Figure 3**.

3.3. Service Hierarchical Structure

The hierarchical structure is built based on the current traffic patterns and a set of rules. As traffic patterns and a

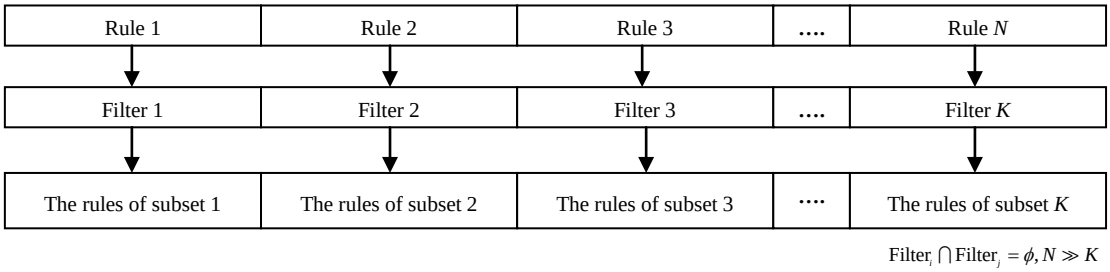

Figure 2. Action rules *N* in subset *K*.

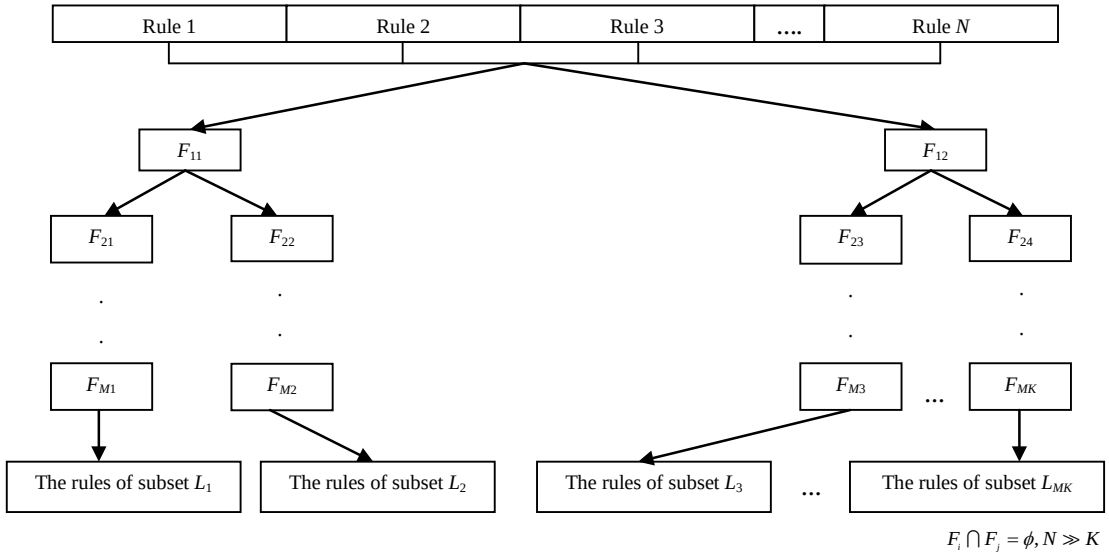

Figure 3. The basic action structure of HAOTF.

set of rules changes, the hierarchical structure should be updated to maintain its balance.

To identify changes, HAOTF monitors traffic logs in real time and adjusts counters. HAOTF notifies that changes have occurred, if the difference between the old and the updated graphs of any rules exceeds a predetermined threshold.

This threshold is configurable parameter that is determined based on the traffic characteristics. If necessary to balance the hierarchical structure, HAOTF uses existing cost upgrade path rules in the rule of subsets, including the rules that have been added to reflect the new security policy. Then HAOTF uses reverse distribution layer, reverse access layer and also assistance layer in restoring balance hierarchical structure. The flowchart of HAOTF between layer hierarchical structures is shown in **Figure 4**.

The reverse distribution layer consists of redistributing priorities in the rule subset deeper layer hierarchical structure. This layer takes effect variations in traffic to the number of rules in the rule set. The reverse distribution layer is triggered when between current and previous number of rules exceeded the threshold value. The reverse access layer caused, when the hierarchical structure becomes out of balance due to changes in traffic [3]. The hierarchical structure is considered to be out of balance, if the average cost of processing packets exceeds a predetermined threshold. This layer may be at any layer, including at the root of the hierarchical structure. When the balance is a hierarchical structure, the access layer is applied to the initial subset rule that is generated from the hierarchical structure. In some cases, impossible to obtain a more balanced hierarchical structure, in this case indicated as optimal layer increases and the threshold for the intermediate layers. The assistance layer aimed at reducing overhead costs packet processing at different layers of the hierarchy. The need to promote the rules

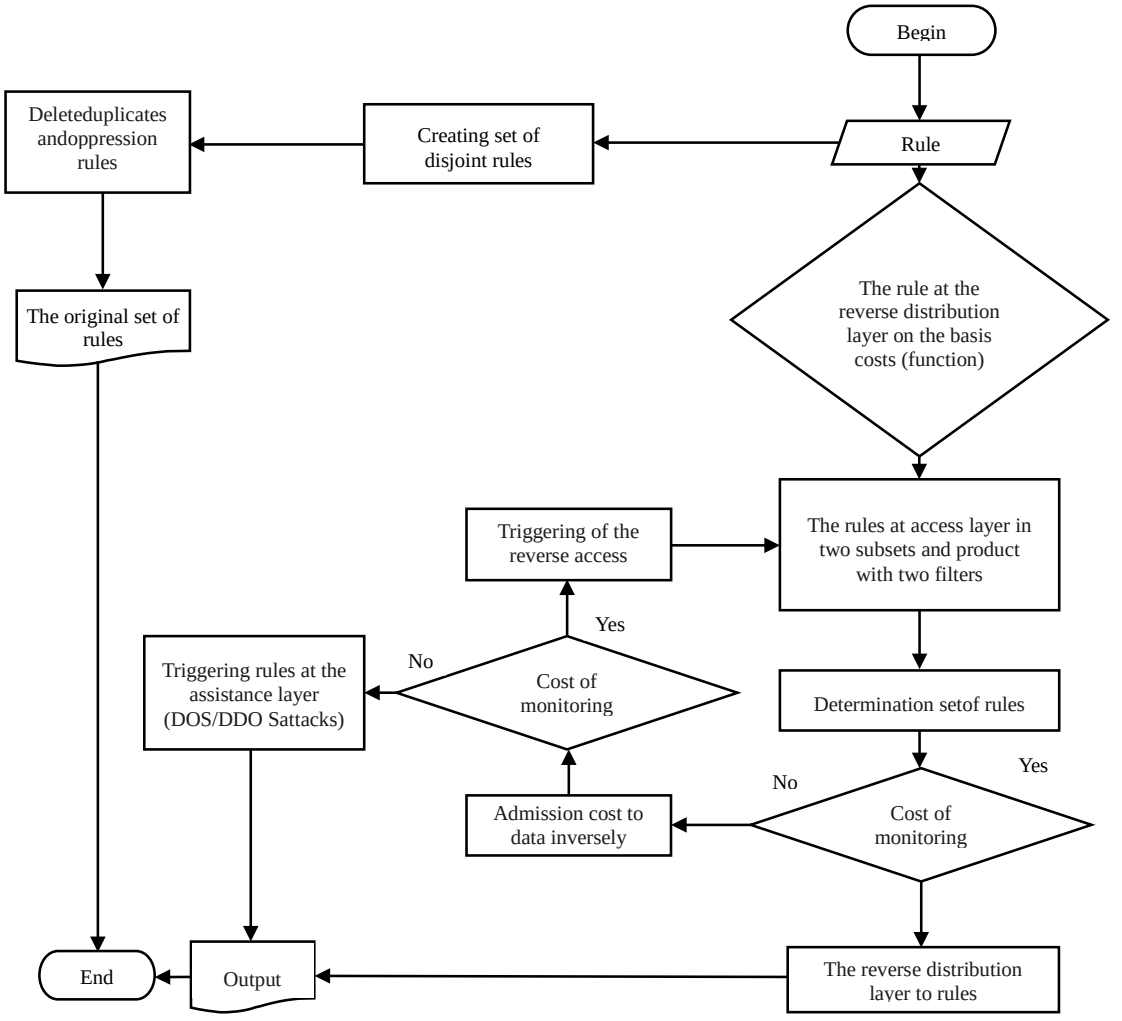

Figure 4. Flowchart of HAOTF between layer hierarchical structures.

occurs when one number rules sharply increases and exceeds a predetermined threshold value. This scenario may occur during abnormal traffic behavior, often observed in a denial of service and distributed denial of service (DoS/DDoS) attacks. To mitigate the impact of DoS/DDoS attacks to significantly reduce the cost of processing the traffic generated by these attacks, usually rises to a layer above the filter [4]. Depending on the rules, the assistance layer may continue recursively until it reaches an appropriate layer of priority. In extreme cases, the rules can be moved all the way up to the root of the hierarchical structure.

This assistance is temporary and usually it is never removed from the rules of subsets. The reason is due to the temporary facilitate temporary state of DoS/DDoS attacks. After the traffic returns to its normal layers, may typically be removed from the higher layers.

The "match" function—checks that the set is covered by the filter. Source and destination IP-addresses are compared with the range specified in the filter. The same port number is compared with a range of ports in the filter. Protocol type is checked against a list of protocol types. This function returns if they meet the set, otherwise they are false.

The "distance" function calculates the distance, given a set of filter. If the filter corresponds to the set, the value returned by this function 0. Otherwise, this function returns 1, positive number between 0 and 1, not inclusive. If the IP-address ranges are then calculated based on a distance function between the two most distant points within. A similar procedure is used to calculate the distance between ports or port ranges. Distance protocol 0, if the protocol already exists in the list of protocols for the filter. Otherwise, sets the distance 1. All distances are normalized to the maximum value corresponds to their fields. Summation of these normalized values are compared again and normalized to produce a value between 0 and 1.

The "extension" function is used to extend the filter so that it corresponds to a given set. This is achieved by expanding the range of IP, port range and protocols. The function calculates costs based on a set of traffic characteristics and other property sets (see **Figure 5**).

4. The Algorithm of Optimal Search Solutions

The algorithm of optimal search solutions uses a priori estimates of the cost of the path to the target state, which ensures high efficiency of the search. The algorithm assumes that there are two lists of vertices of open and closed. In the first peaks are not yet proven algorithm and the second those vertices that have already met in the search for solutions. At each new step from the list of open vertices selected vertex with the smallest weight.

The basic idea of the algorithm is used for each node *n* on the graph of the state space of the evaluation function of the form $f(n)=g(n)+h(n)$. There $g(n)$ on the graph represents the distance from the node *n* to the

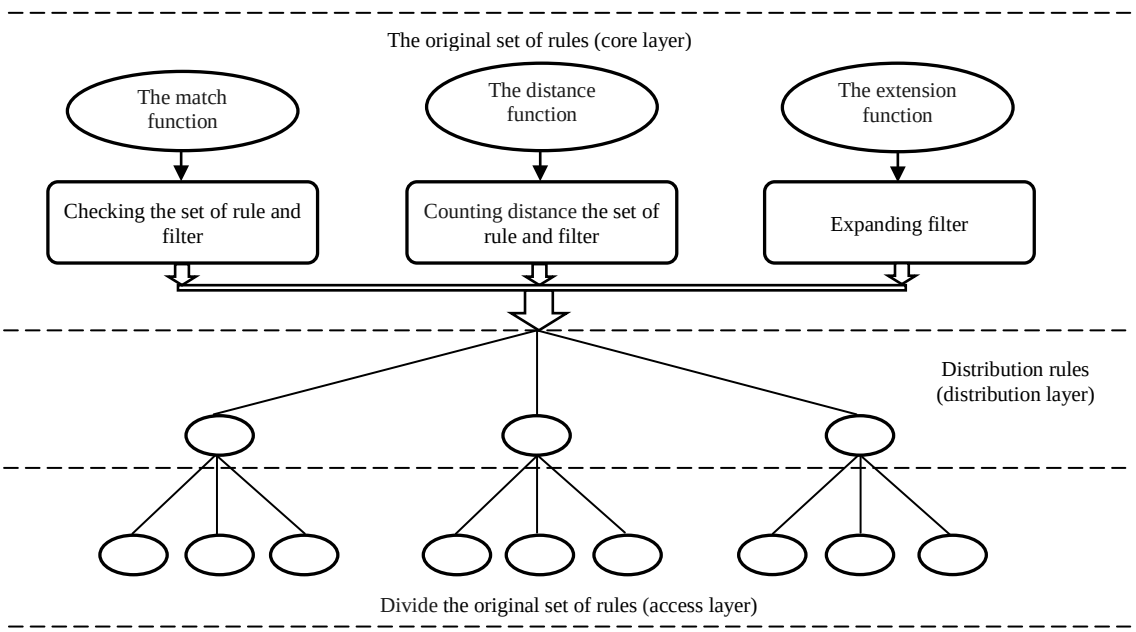

Figure 5. The distributionfunction ofthe core.

initial state, and $h(n)$ estimate of the distance from the node *n* to the node representing the final (target) condition. The lower value of the evaluation function $f(n)$ better, *i.e.* node *n* lies at a short distance from the initial state to the target. The idea of the algorithm is to via $f(n)$ find the shortest path on a graph from the initial state to the target (**Table 1**).

Achieving the algorithm of optimal search solutions in HAOTF possible, since the costs calculated in the aggregate for each division as fixed and does not vary with the priority set of rules.

There function $g(n)$ determines the cost of a set of filter rules from node *n* to the initial state.

Function $h(n)$ on the other hand, calculates the optimal cost remaining sets assigned if placed in any of the subsets, then the value is an approximation of the path to the target node *n*.

Function $h_{\max}(n)$ calculates the maximum number of remaining sets.

It can be used as a guide to complete the calculation of the filter, if the value of the benefits arising from the new filters do not increase the benefits of the previous configuration. Another mechanism that is used to reduce the overhead costs incurred in the search for a possible optimal solutions, limiting the search space.

It is triggered when the difference between $h_{\max}(n)$ and $h_{\min}(n)$ lower than the specified error rate. It is convenient to search for filters and is an optimal solution with a much faster pace.

$$f(n) = g(n) + h_{\max}(n) / h_{\min}(n) \tag{4}$$

Figure 6 shows a topological model search filters in action, where the nodes—is a list of a set of rules and $h(n)$ is the shortest distance to the initial state. Red color—the initial state, blue color—the goal, orange color—visited nodes (see **Figure 6**).

Basic steps algorithm of search filters was shown in **Listing 1**.

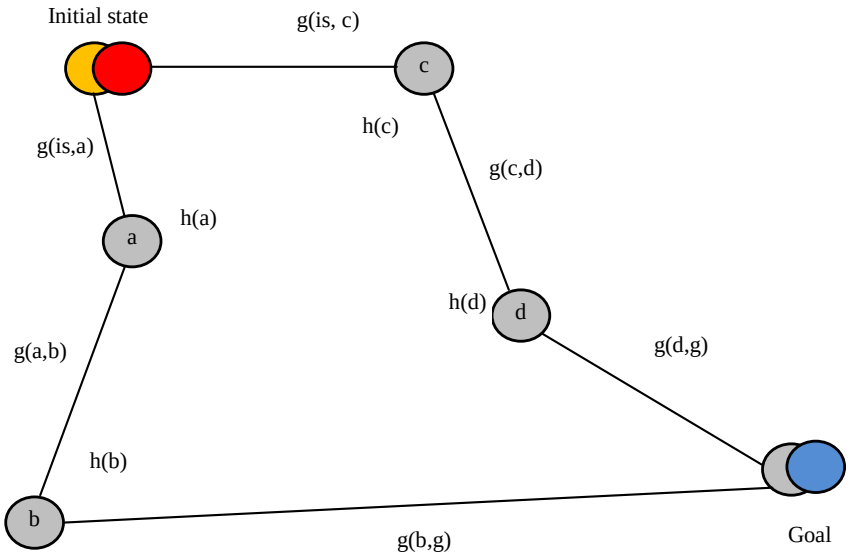

Figure 6. Topological model of searchfilters.

Table 1. Callating algorithm search of path filters.

1.	$f(a) = g(is,a) + h(a)$
2.	$f(c) = g(is,c) + h(c)$
3.	$f(b) = g(is,a,b) + h(b)$
4.	$f(d) = g(is,c,d) + h(d)$
5.	$f(g1) = g(is,a,b,g) + h(g)$
6.	$f(g2) = g(is,c,d) + h(g)$

Listing 1.
$g(i,n)$ = *costs list*$_a$ *ulist*$_b$ *after addition of n sets of the list*
$h(n)$ = *current cost optimal placement of the number of sets* $(i,n) = g(i,n) + h(n)$
filetr$_a$, *list*$_a$ = *filter and set for list A*
filter$_b$, *list*$_b$ = *filter and set for list B*
steck = the required stack and the smallest upper value.
Input:
set[] - *List sorted sets cost*
Algorithm
Counter = 0; *list*$_a$= ϴ, *list*$_b$=ϴ
the current set = set[*counter*]
until the counter < set size ()*perform*
if costs (*A, current set*) < *of expenses* (*B, current set*)
then
iffilter$_a$∩*filter*$_b$*expansion* (*the current set*) ≠
<*any, any, any, any* >
then
createsteck(<*list*$_a$, *list*$_b$*currentset, filter*$_a$, *filter*$_b$. *expansion* (*current set*),*counter*>)
End If
filter$_a$*expansion* (*current set*), *filter*$_a$.*create*(*currentset*)
iffilter$_a$*and filter*$_b$ = <*any, any, any, any*>
then
<*list*$_a$, *list*$_b$, *filter*$_a$, *filter*$_b$, *counter*> = *steck none*()
End If
End If
counter
current set = set[*counter*]
end as long as
Out:<*filter*$_a$, *list*$_a$, *filter*$_b$, *list*$_b$>

Despite the improved algorithms optimal search solutions defined drawback of this algorithm (see **Figure 7**). The disadvantage is that the memory requirements may also be unacceptable, as the number of sets becomes large. To solve this drawback is proposed heuristic method of random search.

5. Heuristic Method of Random Search

Heuristic method of random search based on the sequence of random generation of a large number of values in between finding the roots followed by narrowing these gaps.

The main steps of the method of random search filters:

- in each series with a random number generator is formed from an array of N points of the function $F(x_i), x_i \in [x_{iM}, x_{iH}]$ $(i = 1, 2, \cdots, n)$; n-number of filters $(N > 100)$.
- among the elements of the array of the function is the optimal value $(V_{\min} | V_{\max})$, as well as the corresponding value of the variable $(X_{\min} | X_{\max})$.

And so forth:

Typically, the choice of solutions can be represented sequence elections. If you make this election with the help of a random mechanism, the solution is very fast, so that we can find a solution several times and memorize the "record", the best of solutions encountered. The proposed method of random search filter is a coarse search for solutions aimed at defining a set of filters and separation set list set into two subsets. Each set of list of bases independently of the other set.

- is calculated for each new coordinate interval, within which is produced after selection of the N filter.

And so, in order to reduce costs on a set of rules 10%, $M = 0.9 * (b - a)$

$$\begin{aligned} a1 &= x - M/2; \text{ if } a1 < a \text{ then } a1 = a \\ b1 &= x + M/2; \text{ if } b1 < b \text{ then } b1 = b \end{aligned} \tag{5}$$

Figure 8 shows a topological model of random search filters in action, here:

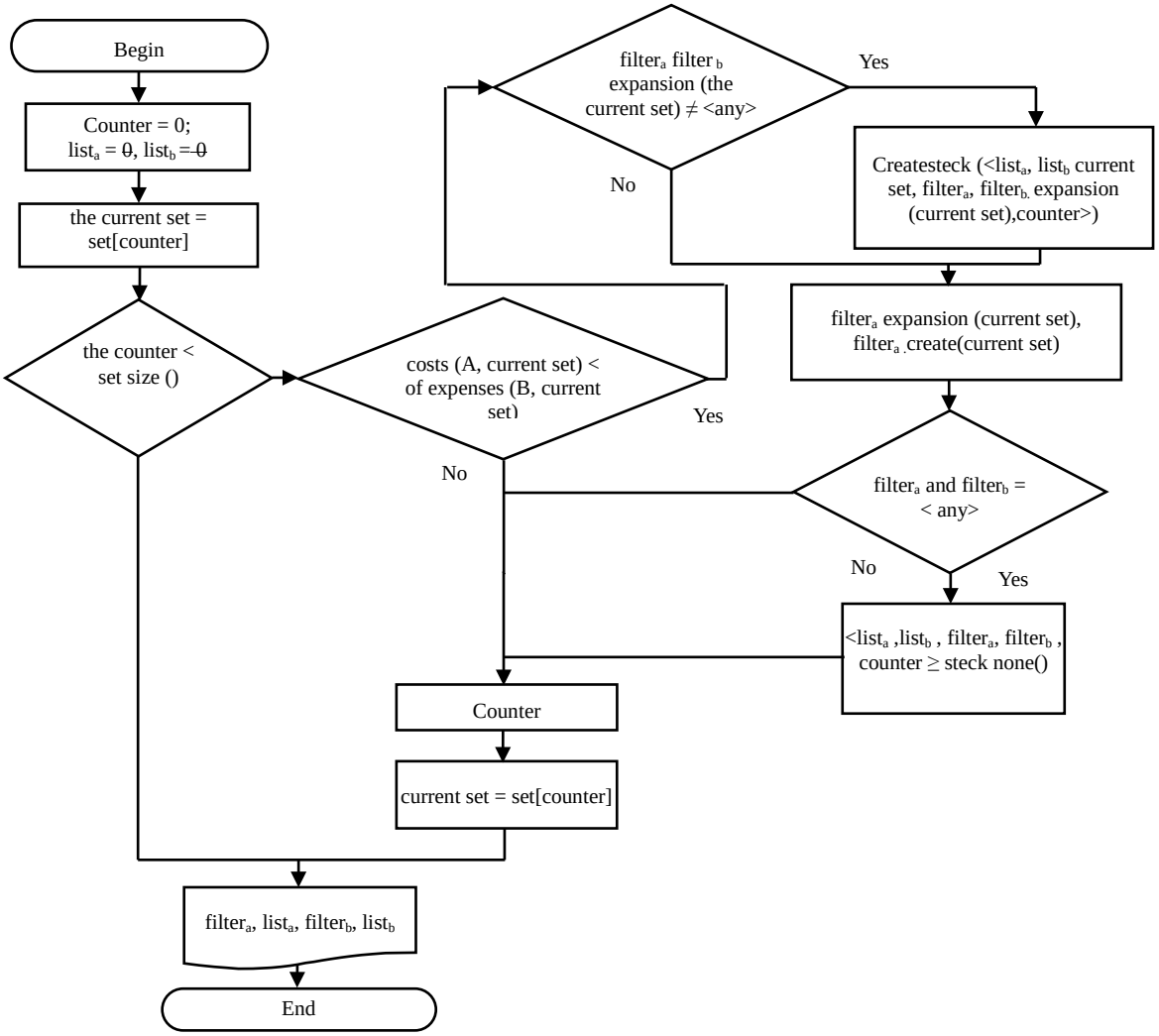

Figure 7. Flowchart of algorithmoptimal search solutions.

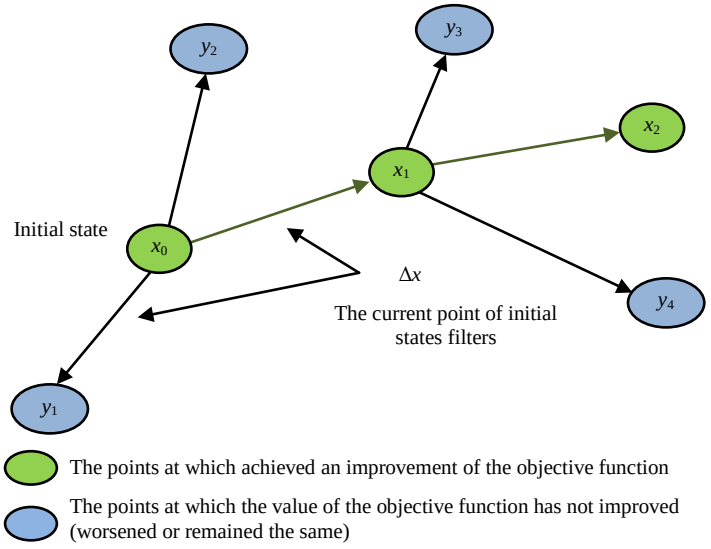

Figure 8. Topological modelof random searchfilters.

The starting point of the state of the filter represented by the vector x_0, declared its current and calculated values of the objective function it.

- The current state of the filter is attached to the point increment in the form of a random vector delta Δx and calculated value of the objective function.
- If the value of the objective function improved, then this point is the current.
- Check the stop condition. If it holds, then go to step 4, Otherwise, in a step 2.
- if $(x_0 = x_1)$ и $(x_1 = x_2)$, while improving the objective function;
- if ($x_0 = y_1$ или $x_0 = y_2$) и ($x_1 = y_3$ или $x_1 = y_4$), then the objective function is deteriorating or remains the same.

This division will perform sets into smaller sets of subsets. This method uses a randomly chosen initial condition of the filter and uses a random algorithm (see **Listing 2**) and search strategy for determining an initial filter from a set of possible filters (see **Figure 9**).

In applying the method sets do not intersect with each other. In other words, the creation of a set of non-overlapping with to each other [5], allow complete flexibility of access layers and reverse distribution based on traffic characteristics.

Basic steps algorithmof random search filters was shownin **Listing 2** (see **Figure 10**).

Listing 2.

Input:
set[] - *lists set sorted by expenditure*
$filter_a$ = *set*[0]
$filter_b$ = *set*[1]
for i = 2 *length set*() *do*
if$filter_a$*matches* (*set*[*i*])
then
create set[*i*] *for* $list_a$
else
if$filter_b$*matches* (*set*[*i*])
then
create set[*i*] *for* $list_b$
else
$distance_a$ =$filter_a$*distance*(*set*[*i*])
$distance_b$ =$filter_b$*distance*(*set*[*i*])
if$distance_a$<$distance_b$
then
$filter_a$*expansion*(*set*[*i*])
create set[*i*] *for* $list_a$
else
$filter_b$*expansion*(*set*[*i*])
create set[*i*] *for* $list_b$

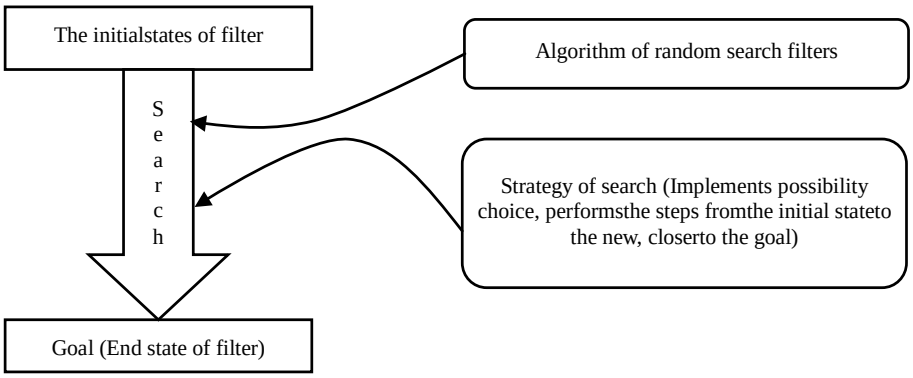

Figure 9. Structure procedure of random search filters.

end if
end if
end if
Out:
filter$_a$ - set filter for list A
filter$_b$ - set filter for list B
list$_a$- set list for filter A
list$_b$- set list for filter B

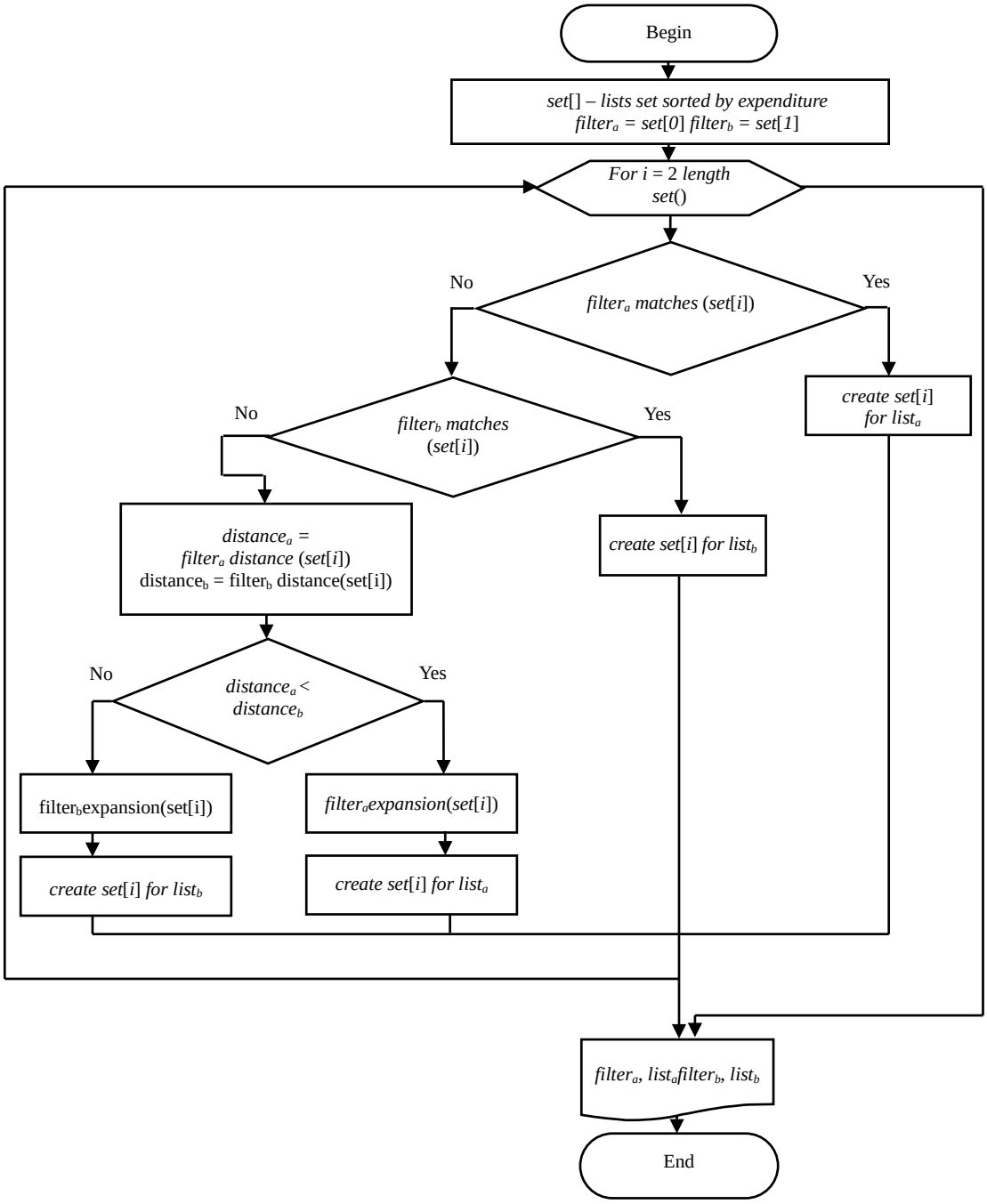

Figure 10. Flowchart algorithm ofa random searchfilters.

6. Evaluate the Efficiency of Accelerating Process Traffic Filtering Proposed by HAOTF

For evaluate the efficiency of accelerating process traffic filtering proposed by HAOTF is proposed scheme, which is shown in **Figure 11**.

There are two types of traffic and it is in the worst case condition is simulated traffic and traffic consists of the same packet that does not match any of the sets. This ensures that the package will be caught only the default action. Emulation traffic generated by creating packages that coincide with each set and the direction of their movement proportional to filter traffic. Traffic filtering action in the worst case is determined by measuring the size and permanent traffic CPU. Size is determined by the traffic load of traffic filtering use from 35% to 100%, the established list of set of rules.

6.1. Evaluate of Processing Attacks

The aim of this research is to test the strength of HAOTF the presence of attacks and traffic fluctuations. Since the number of hits set the default action is great and unpredictable, it can cause a great degree of vulnerability of all the work to filter traffic. To test the performance of HAOTF in handling such attacks are simulated attacks and increases algorithm of random search filters, is determined action set of rules of default from 0 to 200,000 (see **Figure 12**).

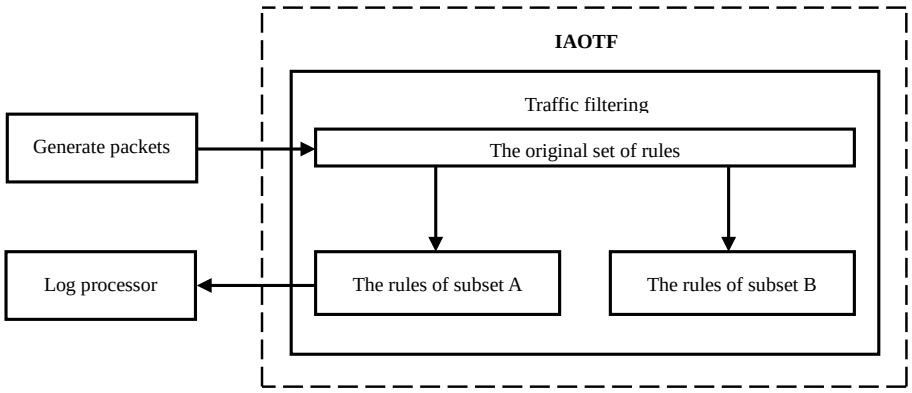

Figure 11. Scheme evaluate the efficiency of accelerating process traffic filtering proposed by HAOTF.

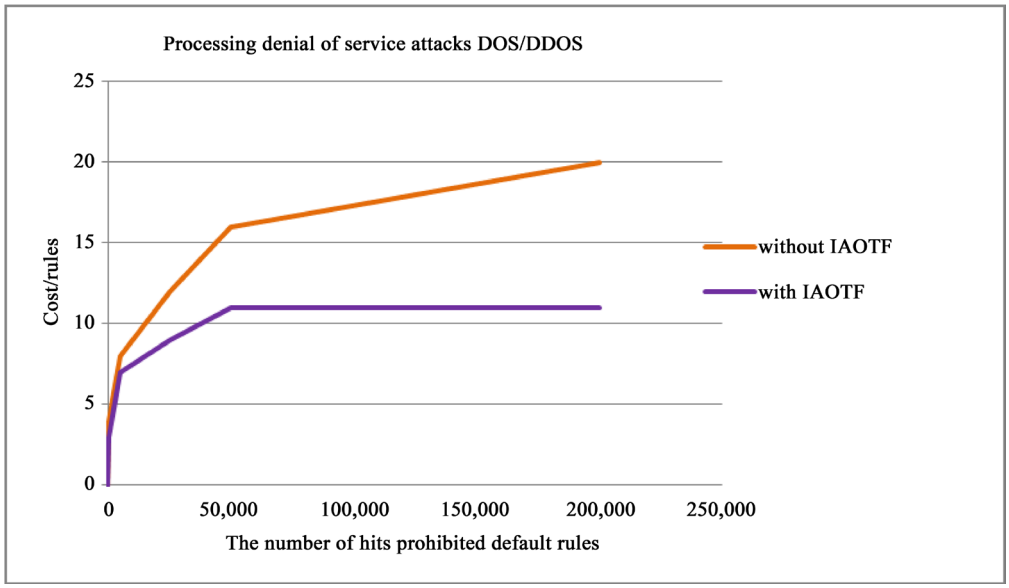

Figure 12. Confrontation DOS / DDOS attacks.

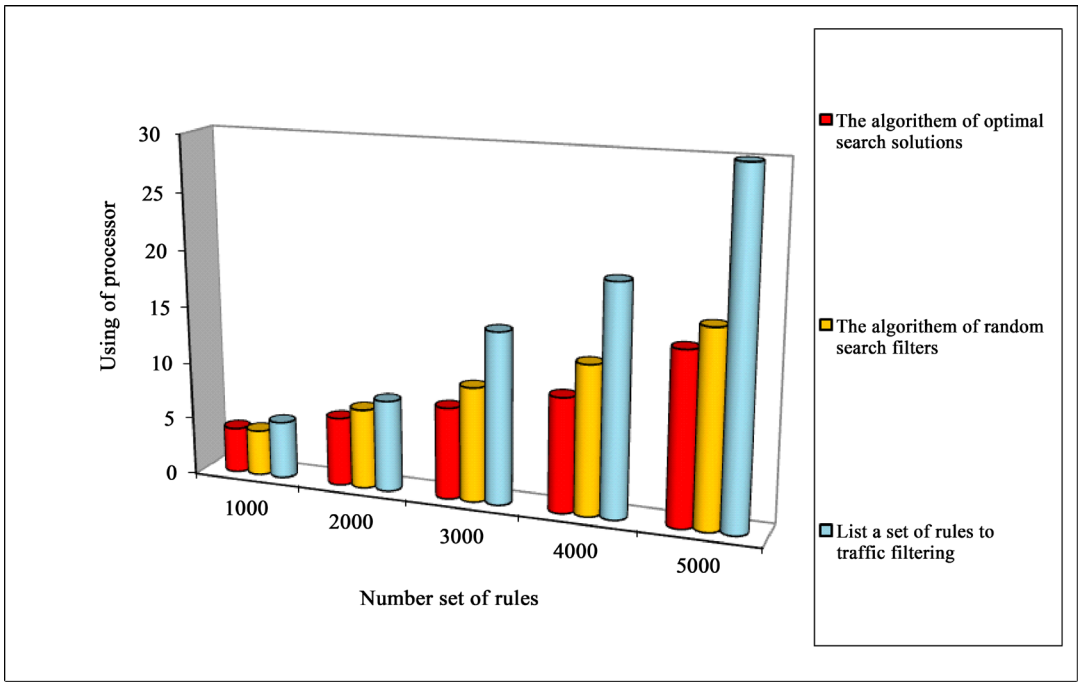

Figure 13. Sensitivity analysis.

6.2. Evaluate of the Sensitivity Analysis

The final research was conducted for the sensitivity analysis of the proposed HAOTF. Analysis was performed for the set rules to different sizes from 0 to 5000 (see **Figure 13**).

Figure 13 shows a comparative research between the list of set of rules traffic filtering, optimal, the algorithm of optimal search solutions and the algorithm of random search filters. The evaluation was conducted for heavily loaded traffic filtering process.

7. Conclusion

This article devoted to the problem of optimization traffic filtering. For this, the article proposes a hierarchical structure for optimization traffic filtering. The basic principle of this structure is to provide lower overhead costs, semantic integrity and security of the initial set of rules to traffic filtering based on two algorithms, such as the algorithm of optimal search solutions and the algorithm of random search filters. In the processing of analysis of the results, it can be concluded that the proposed algorithm of random search filters is suited for hierarchical optimization of traffic filtering.

References

[1] Schneider, F., Wallerich, J. and Feldmann, A. (2007) Packet Capture in 10-Gigabit Ethernet Environments Using Contemporary Commodity Hardware. Springer, Berlin Heidelberg, 207-217.

[2] Wu, W. (2011) G-NetMon: A GPU-Accelerated Network Performance Monitoring System for Large Scale Scientific Collaborations. In *IEEE LCN*'11, 2001.

[3] Lu, G., Guo, C., Li, Y., Zhou, Z., Yuan, T., Wu, H., Xiong, Y., Gao, R. and Zhang, Y. (2011) Server Switch: A Programmable and High Performance Platform for Data Center Networks. *Proceedings of the* 8*th USENIX Conference on Networked Systems Design and Implementation*, Boston, 1 April 2011.

[4] Gulomov, S., Abdurakhmanov, A. and Nasrullaev, N. (2015) Design Method and Monitoring Special Traffic Filtering under Developing «Electronic Government». *International Journal of Emerging Technology & Advanced Engineering* (ISSN 2250-2459, ISO 9001:2008 Certified Journal), Volume 5, Issue 1, January, 2015.

[5] Karimov, M.M., Ganiev, A.A. and Gulomov, S.R. (2014) Models of Network Processes for Describing Operation of Network Protection Tools. 4*th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education*, Sofia, 24-25 October 2014.