Traffic-aware Dynamic Firewall Policy Management: Techniques and Applications
Qi Duan and Ehab Al-Shaer

Abstract—Firewalls are important network security devices that protect networks by blocking the unwanted traffic based on filtering policies. However, the structure of firewall policies has a major impact on the firewall security and performance. In this paper, we classify, describe and compare traffic-aware firewall policy management techniques based on their objectives, schemes, complexity, applicability and limitations. We classify traffic-aware firewall policy techniques into two categories based on their goals: matching optimization and early rejection optimization schemes. Matching optimization techniques try to minimize the matching time of normal network traffic. Early rejection techniques create a minimum set of a policy preamble rules (constraints) that can potentially filter out the maximum amount of denied traffic. Both categories are self-adaptive to ensure that the performance gain will always supersede the dynamic management maintenance overhead. We believe that our work provides important insights on the operation and use of traffic-aware filtering.

I. INTRODUCTION

Firewalls are the most popular network-based security devices that had been widely deployed since the early days of computer networks. They are designed to permit or deny network traffic based upon a firewall policy that specifies what types of packets should be allowed from/into the protected network. Firewalls are usually deployed at the boundary points between networks and subnets. With the growth of the network complexity, it is very common to find firewall policies with thousands of rules. A firewall policy contains a set of rules which are usually checked in a sequential order. This implies that the higher the order of the matching rule, the more costly the firewall filtering overhead will be. Thus, to reduce the filtering overhead, it is crucial to have the appropriate rule ordering in the firewall policy. Firewall policy rule management is also important for security. An attacker can launch Denial of Service (DoS) attacks by overwhelming firewalls with packets that match high ordered rules such as the default deny rule. This attack can drastically reduce the overall throughput of firewalls and increase network delays dramatically [4]. However, finding the optimal rule ordering in firewall policy is a computationally hard problem [7]. It is important to design practical heuristics for adapting firewall policies dynamically without losing the semantic integrity.

The study of many Internet and private traces shows that the major portion of the network traffic flows matches a small subset of the firewall rules, which means that the frequency distribution for some of the traffic properties appears to be highly skewed [9]. Therefore, when performing packet filtering, it is desirable to decrease the number of packet matches required for large flows in order to reduce the overall packet matching time. It is also observed that the skewness in traffic distribution is likely to last long enough to make firewall policies adaptively take advantage of the traffic skewness [9]. Based on these observations, many techniques have been proposed to exploit this traffic property to achieve two goals: increase the firewall performance and defend against DoS attacks.

We classify firewall policy management techniques into two categories (as shown in Fig. 1): matching optimization and early rejection optimization. Matching optimization techniques can be further classified into two types: static and traffic-aware adaptive. Early rejection optimization techniques can be further classified into two types: online and blacklist blocking. In this paper we present a comprehensive survey of dynamic firewall policy management techniques that utilize traffic characteristics and give a detailed comparison for these techniques. We believe that this study can help researchers to have a good understanding of existing works and enable future innovations. We also believe that this study can help engineers and system administrators to be aware of this important problem and adopt the appropriate traffic-aware firewall technique based on specific applications.

The paper is organized as follows. Section II introduces necessary background knowledge for the paper. In Section III we describe statistical matching optimization techniques. In Section IV we describe early rejection optimization techniques. We give a detailed comparison for the techniques in Section V. Section VI concludes the paper.

II. BACKGROUND

A firewall is a network element that controls the traversal of packets across the boundaries of a secured network based on a specific security policy. A firewall security policy is a list of ordered filtering rules that define the actions performed on matching packets. A rule is composed of filtering fields (also called header tuples) such as protocol type, source IP address, destination IP address, source port and destination port, and an action field. Each network field could be a single value or range of values. Filtering actions are either accept, which passes the packet into or from the secure network, or deny, which causes the packet to be discarded. The packet is accepted or denied by a specific rule if the packet header information matches all the network fields of this rule. Otherwise, the following rule is examined and the process is repeated until a matching rule is found or the default policy action is performed. The filtering rules may not be disjoint, thereby packets may match one or more rules in the firewall policy. In this case, these rules are said to be dependent or overlapping and their relative ordering must be preserved for the firewall policy to operate correctly.
For example, in Fig. 2(a), rules $R_1$ and $R_2$ are overlapping rules.

The traffic address space is the space whose elements contains all possible tuples that identify traffic flows in a network environment. This is usually composed of the transport protocol, source address/port, and destination address/port. A segment is a subset of the total traffic address space that is covered by a unique set of rules. In other words, segments are equivalence classes over packets. The equivalence of packets is defined by the same policy rules that match these packets. Therefore, segments exhibit the following two properties: (1) all segments are pairwise disjoint, and (2) any packet must fall in exactly one segment. Fig. 2(a) shows a simple firewall policy composed of three rules: $R_1$, $R_2$, and $R_3$. Fig. 2(b) shows the segmentation of the firewall policy in Fig. 2(a). As a result of intersecting (or overlapping) the three rules, four address segments are produced: $S_1$, $S_2$, $S_3$, and $S_4$.

III. MATCHING OPTIMIZATION TECHNIQUES

A. An Overview

The objective of matching optimization is to reduce the number of rules to be inspected in average case [9]. As shown in Fig. 1, there are two types of matching optimization techniques: static and adaptive. Many algorithmic-based techniques have been proposed for static filtering optimization. These techniques try to improve the search time using various algorithmic techniques such as hardware-based solutions, specialized data structures and heuristics. However, these static techniques are used to improve the worst-case scenario and they do not consider the properties of network traffic. The earliest work to exploit traffic properties for packet classification is by Gupta et al. in [6]. In this work, the authors used depth-constrained single-field alphabetic trees to reduce lookup time of destination IP addresses of packets against entries in the routing table. However one cannot directly use it for firewalls which have multiple matching fields. In this paper, we focus on adaptive optimization techniques that improve the efficiency of filtering in average case. These techniques can adjust the filtering policies to fit the matching frequency of firewall rules or filtering field values. Rule based techniques include common branch decision tree, offline statistical-based rule generation and dynamic rule ordering. Field value based techniques include multi-field alphabetic tree, Huffman tree based filtering and segment list based filtering. In the following subsections, we will describe these traffic-aware dynamic firewall policy techniques because they represent the state-of-the-art of traffic-aware firewall filtering for firewalls with large number of overlapping rules.

B. Rule Based Optimization

1) Common Branch Tree: In [3], the authors present common branching tree based packet classification algorithms in an operational network. Decision trees can be categorized under three optimization criteria: worst-case, average case and mixed. Based on the skewness in Internet traffic, they suggest that average-case time is an important metric in the packet classification settings and propose algorithms to construct common branches decision trees to achieve good average-case performance. It needs time $\Theta(k\log(l/k))$ and space $\Theta(l)$ to build common branches decision tree, where $l$ is the number of rules and $k$ is the number of fields.

The experimental evaluation on real-world filters showed that common branches trees use much less memory than binary decision trees and have comparable worst-case and average-case search times. The good performance of common branching tree can be attributed to the presence of extensive wildcarding with certain structure in the rule sets. The limitation of the technique is that the entire decision tree needs to be rebuilt every time when the traffic pattern changes, and it is not appropriate for heavily overlapping rules.

2) Offline Statistical-based Rule Generation: In [1], the authors presented an offline statistical-based rule generation technique called Traffic-aware Firewall Optimizer (TFO). The first step in TFO is called Pre-Optimization. This step will remove all redundancies in the rule set. The second step of TFO uses two optimizers: a rule set based optimizer and a traffic based optimizer. The rule set based optimizer contains the Disjoint Set Creator (DSC) algorithm and the Disjoint Set Merger (DSM) algorithm. The DSC algorithm converts the original rule set to a semantically equivalent disjoint rule set, which can provide the traffic based optimizer with full flexibility to re-order rules based on traffic characteristics. The DSM algorithm merges the rules of the disjoint rule set produced by DSC to reduce rule set size. The traffic based optimizer has four components, namely hot caching, total re-ordering, default proxy, and online adaptation. The hot caching scheme tries to put those rules that are frequently hit to the top of the rule set. The total re-ordering scheme combines the measure of hit frequency and rule size to optimize rule ordering. The default proxy scheme can optimize the firewall performance when the default deny action is heavily invoked. The online adaptation scheme builds long-term rule hit profile to optimize the rule set.

TFO is based on the assumption that only a small portion of rules are dependent on other rules, so it cannot handle policies with heavily overlapped rules. Also, TFO has only limited adaptivity because the rule hit profile is built offline. The authors conducted experiments in only one enterprise firewall, which may not be able to represent other real world firewalls.

3) Dynamic Rule Ordering: This technique uses a heuristic approximation algorithm for optimal dynamic firewall rule ordering based on real-time traffic characteristics [7]. The objective of optimizing the firewall rules is to create a semantically equivalent rule order that minimizes the packet matching time in firewalls. The Optimal Rule Ordering (ORO) problem is mapped to the job scheduling problem for a single machine with precedence constraints. Since the scheduling problem is NP-hard, the general ORO problem is also NP-hard. Thus a heuristic approximation algorithm that runs in polynomial time and achieves near-optimal results for the most common firewall policies is presented.

The implementation of the technique also includes a method to compute filtering rule weights in order to capture the matching importance of every rule relative to others. Each rule in the filtering policy is given a weight that reflects the dominance of this rule in matching the traffic processed by the firewall.
The rule weight is calculated based on matching frequency, which determines how frequent the rule has been triggered, and matching recency, which determines how recent the rule has been triggered during packet matching. The optimized rule list is constructed based on the computed rule weights and is used for matching incoming packets to the firewall. Since the traffic distribution of Internet flows over filtering rules is constantly changing, the rule weights should be dynamically adjusted to reflect the most recent distribution. Two types of rule list updates are used: performance-based triggered updates and time-based periodic updates. In this way, an ordering that is as close as possible to the optimal can be computed, while the overhead to compute these updates can be minimized. The limitation of the technique is that it is not good for policies with a large number of overlapping rules.

C. Field Based Optimization

1) Multi-field Alphabetic Tree: In [8] and [9], a new multi-field alphabetic tree based technique for dynamic firewall policy management was presented. The technique calculates the field value frequency (entropy) and uses this entropy information to build the alphabetic search tree for adaptive packet filtering. The alphabetic search tree stores field values in the leaves based on given weights such that the inherent order of the stored values is preserved. At each internal node, the left subtree contains nodes that have values less than those at the right-hand-side. This added constraint of enforcing an order on the placement of values in the tree enables the matching algorithm to branch left or right. The constraint can also eliminate the need for preprocessing of the packet field values. The alphabetic search tree inserts values of higher occurrence probability (matching frequency) at higher tree levels than the values with less probability. Field values that commonly exist in the traffic will have less number of packet matches compared to uncommon values.

The alphabetic search tree can improve the overall average filtering by significantly reducing the number of matches for most popular packets, though it may not be in favor of less-frequently matched traffic. The gain in the filtering performance is proportional to the degree of skewness in the traffic distribution over field values. Even in the worst case scenario when the traffic distribution is uniform, this technique cannot do worse than the binary search as a lower bound.

The alphabetic search tree can be build with time $O(n\log n)$ and space $O(n)$ where $n$ is the number of rules. The limitation of this technique is that the overhead of updating the tree can be significant.

2) Huffman Tree Based Filtering: This technique in [5] uses Huffman tree to represent the segmentation of traffic address space in the firewall policy. The Huffman tree can minimize the average number of comparisons applied on packets arriving at the firewall ports.

To build the Huffman tree, the traffic statistics can be kept as hit-count measurements for every segments. The internal nodes of the Huffman tree contain a Boolean expression that should be satisfied by each packet passing through this node towards its descendants. The expression is built by taking logic OR of the two corresponding expressions from the children nodes. When a new node is created in a merging step of two nodes with the lowest weight, it will have the expression that takes the logic OR of these two nodes.

The operation in the Huffman tree building algorithm differs slightly with the original Huffman’s algorithm. The changes occur in swapping the left and right children based on whichever is easier to evaluate. Another difference is the added expression at each node.

The time and space complexity of building the Huffman tree are $O(ns)$ and $s$ respectively, where $n$ is the number of rules and $s$ is the number of segments in the policy. Using the skewness of the segments to build a Huffman tree over these segments can enhance the performance of searching. The technique is good for policies with a large number of rules. The limitation is that the Huffman tree needs to be rebuilt periodically to reflect the changes in the network flows.

3) Segment List Based Filtering: In order to get rid of the maintenance cost of the Huffman tree, one can use policy segments-based search list [5] to utilize the very high imbalance in the frequency distribution of packets over the policy segments. One can obtain a very simple yet extremely effective structure by building a simple list of segments that is updated after each packet match. Theoretically, the optimal order is to have the segments sorted in reverse order of their popularity. But it is impossible to guarantee this without prior knowledge of the exact distribution. A heuristic algorithm can be used to minimize the average search time. The main idea in the algorithm is to match an incoming packet against the segment list one-by-one, and once a match is found, the list ordering will be adjusted.

The time and space complexity of to build the segment list are $O(ns)$ and $s$ respectively, where $n$ is the number of rules and $s$ is the number of segments in the policy. This technique only contains very simple operations. It is good for extremely biased traffic. The only limitation is that it has a transient behavior until a good order of segments is obtained.

IV. Early Rejection Optimization Techniques

A. An Overview

The problem of early packet filters received much attention for two main reasons. The first is to protect firewalls from DoS attacks that target the default deny rule. The second is to minimize the filtering overhead due to discarding unwanted traffic by introducing approximate policies that can easily filter out discarded traffic.

Early rejection optimization can be classified as online and blacklist blocking. Online early rejection techniques include field value cover early rejection [8] and BDD based relaxed policy [4]. Blacklist blocking techniques include LCP based blacklist blocking [10].

B. Online Early Rejection

1) Field Value Cover Based Early Rejection: The first paper to present an early rejection optimization technique is [8]. In this technique, an early filtering module is deployed as another layer before the actual filtering that takes place in a firewall. The goal of this early filtering module is to filter out as many discarded packets as possible with the lowest overhead. This is
because discarded packets might traverse a long decision path of rule matching before they are finally rejected by the default deny rule, which causes significant overhead proportional to the number of rules in the firewall policy. For those packets that the early filtering module has made a decision (deny or accept), there is no need to pass the packet to the original filtering module. If the early filtering module cannot reach a decision based on its approximation of the policy, then the filtering process will be delegated to the original filtering algorithm.

The early rejection rules (RR) can be formed as a combination of the common field values that cover all rules in the policy. It can be shown that these rules are more feasible to find because the number of distinct field values is usually small relative to the policy size. It is desirable to search in the firewall policy for a combination of common field values such that every rule uses at least one of these values. This problem can be modeled as a set cover problem. Two approximation algorithms can be used to solve this problem. The first one runs in time \( O(n^2) \) and has an approximation ratio \( \log(n) \) where \( n \) is the number of rules. The second one uses integer programming to achieve an \( f \) approximation ratio where \( f \) is the maximum number of subsets that any element can belong to. The set cover approximation algorithm generates a combination of common field values to be used as early rejection rules. But it is not known in advance how many and which ones that we should use to achieve an optimal rejection solution. A suit of three algorithms were used to address this problem. These three algorithms are the startup phase algorithm, dynamic rule selection algorithm and early rejection algorithm. In the startup phase algorithm, the candidate rejection rule list is built from different solutions that belong to the set cover problem. The dynamic rule selection algorithm is responsible for the periodic addition/removal of rules according to the performance gain/loss of each rule. The early rejection algorithm calculates the location of early rejection relative to normal packet filtering and defines the per-packet operation of filtering. The limitation of the technique is that it is not suitable for large policies (policies with a large number of rules).

2) BDD Based Relaxed Policy: Another technique is called BDD (Binary Decision Diagram) based relaxed policy. The basic idea of BDD based relaxed policy is to approximate the current policy with another new policy. The technique evaluates every packet against the new policy, and decide to accept, reject, or forward it to the original policy. The original policy is deployed as normal, but it is not executed unless the early filtering module fails to reach a decision.

Efficient Boolean expression can be used to represent and approximate the policy. The actual implementation is based on Boolean expressions, which are constructed using BDD. Each Boolean expression represents the different packets that match a specific rule, and the variables used for this expression correspond to the bits of individual packet header fields (e.g., source IP, destination port, protocol, etc). BDDs can facilitate the matching by representing the expression in the form of a tree, where each variable is checked only once. An important advantage of the BDD tree is that the decision can be quickly made for a large portion of the packet space. When a packet arrives, the fields in the packet header are extracted and sorted according to their order in the expression tree, so they can be used one-by-one in navigating the tree. Tree navigation is itself a very simple set of instructions. One needs only to check the variable at the current node, load the left child node entry in the BDD table if true and the right child entry if false. This is repeated until a node is reached with a final value or reaching the maximum depth allowed in the tree. Note that this BDD construction technique can also be used for matching optimization.

The whole system is dynamic to traffic properties and can be tuned by the policy structure and previous performance measures. The limitation of the technique is that the overhead to build the BDD is usually significant.

C. LCP Based Blacklist Blocking

This technique is the general framework for modeling the filter selection as resource allocation problems in [10]. A filter is a set of simple access control rules to specify that the addresses with certain prefixes should be blocked. The goal of filter selection is to build filtering rules that can minimize the impact of malicious sources in the network using the available network resources. The technique considers different filtering problems based on different attack scenarios, operators’ policies and constraints. Each filtering problem can be modeled as an optimization problem. The data structure to represent the problems is the Longest Common Prefix (LCP) tree. The LCP tree is a kind of binary tree such that the leaves of the tree are the malicious IP addresses and all the other nodes represent the longest common prefixes between any pair of IPs in the tree. The LCP tree can be built in time \( O(mN) \) and space \( O(N) \) where \( m = 32 \) is the number of bits in the IP address and \( N \) is the number of malicious IP addresses. Polynomial or pseudo-polynomial algorithms were designed to solve the filtering problems. The limitation of this technique is that all the malicious IP addresses must be known before the computation of the optimal solution. To defend the attackers who can move quickly among multiple source IPs, one need to re-compute the optimal solutions frequently. This technique may not be efficient enough in this case.

V. A Comparative Study

In this section we present a detailed comparison for the traffic-aware dynamic firewall policy management techniques discussed in the paper.

Table I shows the comparison for the data structure, algorithm, adaptivity, measurement and complexity (time and space) of the techniques. Here the “adaptivity” column denotes whether the technique is online or offline, and the “measurement” column denotes the key quantitative values used in the technique. We compare the nine techniques discussed in the paper for traffic-aware packet filtering. The algorithmic-based static techniques in Fig. 1 are not included in our study since they are not traffic-aware filtering techniques. Note that in the table, \( n, d, s, \) and \( N \) are the number of rules, the number of fields, the number of segments, and the number of malicious source addresses in the policy, respectively. For BDD based techniques, in the worst case the complexity can be exponential. However, in most cases, the complexity of BDD construction
for firewall policies does not exceed the quadratic of the number of rules due to rule correlation and overlapping [2]. For **offline statistical-based rule generation**, the complexity is not known because the authors did not give a complexity analysis. We can note that **dynamic rule ordering** and **multi-field alphabetic tree** have similar complexity, while **Huffman tree based filtering** and **segment list based filtering** have complexity related to the number of rules and segments in the policy. Table II compares the functionality, policy structure, traffic characterization and maintenance cost of each technique. In this table, plus (+) sign means the corresponding technique is suitable for the property, and minus (−) sign means the corresponding technique may not be suitable for the property. N/A means the technique is not applicable for the property. Skewness is the measure of the asymmetry in the probability distribution of the traffic. Dynamic means that the traffic pattern has frequent changes.

**LCP based blacklist blocking** is not applicable for general policies of multi-field rules. Although **dynamic rule ordering**, **common branch tree**, and **offline statistical-based rule generation** techniques are good for rules with multiple fields, they are not suitable for heavily overlapping policies. Although **dynamic rule ordering**, **segment list based filtering**, **multi-field alphabetic tree** and **Huffman tree based filtering** techniques exhibit high dynamism and low maintenance overhead, they expect high-to-moderate skewness in the traffic distribution (i.e., there is a small number of popular servers) except **dynamic rule ordering**, which can tolerate low skewness in traffic.

**BDD based relaxed policy**, **field value cover based early rejection**, and **offline statistical-based rule generation** are not suitable for large policies or policies with large number of field values. **Huffman tree based filtering**, **common branch tree**, and **offline statistical-based rule generation** could exhibit high maintenance overhead. Online early rejection techniques (**BDD based relaxed policy**, **field value cover based early rejection**) look robust with traffic distribution and rules overlapping, but they suffer from the high number of rules. It is worth noticing that **Offline statistical-based rule generation** is not fit for overlapping rules and traffic dynamics, and **LCP based blacklist blocking** can only be applied efficiently to a pre-defined blacklist. To achieve the desired goal one should choose the appropriate technique according to the properties of the firewall policy.

**VI. Summary**

In this paper we present a taxonomy of traffic-aware firewall management techniques. We believe that our analysis of the research works in traffic-aware dynamic firewall policy management can help researchers and engineers to understand the importance of the problem and provide useful guidance for adopting a specific technique based on application requirements. This study will provide ingredients for more innovative solution in this field.

**REFERENCES**


Classification of firewall policy management techniques

Matching optimization
  - Static
    - Algorithm based static techniques
      - Rule based optimization
        - Common branch tree
        - Offline statistical-based rule generation
      - Adaptive traffic-aware
        - Dynamic rule ordering
          - Multi-field alphabetic tree
          - Huffman tree based filtering
      - Field based optimization
        - Segment list based filtering
  - Field value cover based early rejection
  - BDD based relaxed policy
  - Offline statistical−
  - Matching optimization
    - Early rejection optimization
      - Online early rejection
      - Static
        - Adaptive traffic-aware
          - Rule based optimization
            - Common branch tree
            - Offline statistical-based rule generation
          - Dynamic rule ordering
            - Multi-field alphabetic tree
            - Huffman tree based filtering
          - Field based optimization
            - Segment list based filtering
            - Strict
          - LCP based blacklist blocking
            - Relaxed
              - BDD based relaxed policy

Fig. 1. Classification of Traffic Aware Firewall Policy Techniques
Fig. 2. An example of firewall rules and segmentation

<table>
<thead>
<tr>
<th>Rule</th>
<th>Protocol</th>
<th>Source Address:Port</th>
<th>Destination Address:Port</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>tcp</td>
<td>140.192.37.30:100</td>
<td><em>.</em>.<em>.</em>:any</td>
<td>accept</td>
</tr>
<tr>
<td>R_2</td>
<td>tcp</td>
<td><em>.</em>.<em>.</em>:20</td>
<td>140.192.37.30:100</td>
<td>deny</td>
</tr>
<tr>
<td>R_3</td>
<td>tcp</td>
<td><em>.</em>.<em>.</em>:any</td>
<td><em>.</em>.<em>.</em>:any</td>
<td>deny</td>
</tr>
</tbody>
</table>

(a) Rule set (b) Segmentation
<table>
<thead>
<tr>
<th>Technique</th>
<th>Data Structure</th>
<th>Algorithm</th>
<th>Adaptivity</th>
<th>Measurement</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common branch tree [3]</td>
<td>tree</td>
<td>decision tree optimization</td>
<td>offline</td>
<td>N/A</td>
<td>$\Theta(d\log(n/d))$ or $n^{0.63}$</td>
<td>$n^{287^{(2)}}$</td>
</tr>
<tr>
<td>Offline statistical-based rule generation [1]</td>
<td>list</td>
<td>optimal splitting</td>
<td>offline</td>
<td>N/A</td>
<td>$2^n$</td>
<td>$n$</td>
</tr>
<tr>
<td>Dynamic rule ordering [7]</td>
<td>rule</td>
<td>job scheduling with precedence</td>
<td>online</td>
<td>rule frequency and recency</td>
<td>$O(n \log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Multi-field alphabetic tree [8]</td>
<td>tree</td>
<td>alphabetic tree</td>
<td>online</td>
<td>field entropy</td>
<td>$O(n \log n)$</td>
<td>$O(dn)$</td>
</tr>
<tr>
<td>Huffman tree filtering [5]</td>
<td>tree</td>
<td>Huffman tree</td>
<td>online</td>
<td>segment entropy</td>
<td>$O(ns)$</td>
<td>$O(s)$</td>
</tr>
<tr>
<td>Segment list based filtering [5]</td>
<td>list</td>
<td>linear search</td>
<td>online</td>
<td>segment counting</td>
<td>$O(ns)$</td>
<td>$O(s)$</td>
</tr>
<tr>
<td>Field value cover based early rejection [8]</td>
<td>Boolean expression</td>
<td>set cover approximation</td>
<td>online</td>
<td>statistical thresholding</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>BDD based relaxed policy [4]</td>
<td>BDD</td>
<td>BDD optimization</td>
<td>online</td>
<td>rule probability</td>
<td>$O(2^n)$</td>
<td>$O(2^n)$</td>
</tr>
<tr>
<td>LCP based blacklist blocking [10]</td>
<td>tree</td>
<td>dynamic programming</td>
<td>offline</td>
<td>N/A</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
</tbody>
</table>

**TABLE I**

Comparison for the algorithm, data structure and complexity of traffic-aware dynamic firewall policy techniques
<table>
<thead>
<tr>
<th>Technique</th>
<th>Functionality</th>
<th>Policy structure</th>
<th>Traffic characterization</th>
<th>maintenance cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>matching</td>
<td>multi-</td>
<td>large</td>
<td>skewness</td>
</tr>
<tr>
<td></td>
<td>optimization</td>
<td>field</td>
<td>No. of field value</td>
<td>dynamic</td>
</tr>
<tr>
<td></td>
<td>early</td>
<td>high overlapping</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rejection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common branch tree</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td>Offline statistical-based rule</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>N/A</td>
</tr>
<tr>
<td>generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic rule ordering</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi-field alphabetic tree</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td>Huffman tree filtering</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td>Segment list based filtering</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td>Field value cover based early</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td>rejection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BDD based relaxed policy</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>LCP based blacklist blocking</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**TABLE II**

Comparison for Limitation and Application Suitability of the Traffic-aware Dynamic Firewall Policy Techniques