Firewall Packet Filtering Optimization Using Statistical Traffic Awareness Test

Zouheir Trabelsi, Liren Zhang, and Safaa Zeidan

Faculty of Information Technology
UAE University
Al-Ain, UAE
{trabelsi,lzhang,safaa.z}@uaeu.ac.ae

Abstract. In this paper, we present a mechanism that utilizes network traffic behavior and packet filtering statistics to improve firewall performance. The proposed mechanism allows optimizing the filtering rules order and their corresponding fields order upon certain threshold qualification following the divergence of the traffic behavior. The current and previous traffic windows statistics are used to check the system stability using Chi-Square Test. The achieved gain in processing time compared to related mechanisms is due to minimizing the overhead corresponding to the frequency of updating the security policy rule/field structures.

Keywords: Packet Classification, Rule Order, Rule-fields Order, System Stability, Chi-square Test.

1 Introduction

In this paper, we propose a mechanism to optimize firewall early acceptance path as well as early rejection path. Based on traffic statistics, a decision is made regarding whether or not there is a need to reorder the rules and/or rule-fields orders. The proposed mechanism is based on the following three optimization levels: 1) Filtering rules are reordered in a descending manner according to their packet matching histograms. This will yield to faster packet filtering time for the next similar traffic (optimization in the acceptance path). 2) Rule-fields are reordered in a descending manner according to their packet not matching histograms. This will reduce the time needed for tuple comparison (optimization in the rejection path). 3) The firewall will continue filtering packets using certain rule and rule-fields orders under a certain threshold qualification (the system stability decision). This will reduce the time needed for the reordering process and updating the firewall security policy structure. The three optimization levels will minimize the total packet filtering time and therefore the firewall performance will be improved significantly.

The paper is organized as follows: Section 2 discusses the related work. Section 3 presents the mathematical model of the proposed mechanism. Section 4 evaluates the firewall performance against the proposed mechanism. Finally, Section 5 concludes the paper.

2 Related Work

The most early research works on firewall focus on the improvement of packet searching times using various mechanisms including hardware-based solutions [6, 7], specialized data structures [8, 9, 5, 10, 11, 12], and heuristics [5]. Research works in [17, 23, 4, 13 and 14] focus on the statistical filtering schemes to improve the average packet filtering time. The structure of searching by taking into account the packet flow dynamics is introduced in [3, 14, 15, 16].

The idea of firewall optimization through early packet rejection was introduced in [1, 2, 23, 21and 18]. In [1] early packet rejection is done through rule- fields ordering. In [2] early packet rejection is done through multilevel filtering process including field and intersection filtering modules. In [23] a new approach named FVSC is proposed to optimize the rejection path. PBER technique in [18] is considered as a generalization of FVSC [23] it finds short cuts for both accepted and rejected packets. In [21] a binary search on prefix length algorithm is applied to every policy filtering field along with the property of splaying the search tree nodes while maintaining the min-node at high level for early packet rejection.

The most relative to this paper is research work dealing with rules reordering which falls into two categories: Rules reordering including dependency as in[16, 23,] these research work give an approximation of the optimal rules order. Disjoint Rules reordering as in [24, 25]. Up to our knowledge all research work done in the field of firewall optimization through rule reordering, emphasis on the importance of rule field reordering in early packet rejection. In [1] we were the first to propose the idea of rule-field reordering and focus on its major effect in reducing the overall packet processing time.

3 Proposed Work

In this paper, we use the Firewall Decision Tree Tool (FDT) described in [26]. FDT tool releases the dependency relation between rules. As a result, the filtering rules can be reordered according to their matching frequencies.

The mathematical model in this paper is based on rule and rule-fields histograms proposed in [1]. A mechanism named Dynamic Rule and Rule-Fields Ordering (DR-RFO) is proposed in [1]. In which the reordering process is carried out at the end of each network traffic window. Thus, in this paper we propose a mechanism named Dynamic Rule and Rule-Fields Ordering with Decision Test (DR-RFOD) to organize the reordering process according to the system stability test.

Since the proposed work in this paper uses the rule and rule-fields histograms, we will describe them in more details in the following section. Then we will discuss the decision regarding the rule/rule-fields reordering processes.

3.1 Mathematical Model

Histogram of Rule Matching Probability and Field not Matching Probability
Considering that packet matching test in firewall is based on a security policy with N rules, excluding the default "Deny" rule. Each rule consists of a maximum number of M fields, excluding the action field. A N×M matrix vector F(i,j) represents the security policy, that is:

$$\mathrm{F(i,j)} = \begin{bmatrix} R(1) \\ \dots \\ R(i) \\ \dots \\ R(N) \end{bmatrix} = \begin{bmatrix} F(1,1) \dots F(1,j) \dots F(1,M_1) \\ \dots \\ F(i,1) \dots F(i,j) \dots F(i,M_i) \\ \dots \\ F(N,1) \dots F(N,j) \dots F(N,M_N) \end{bmatrix} \tag{1}$$

Where $i \in \{1,2,\dots,N\}$ and $j \in \{1,2,\dots,M_i\}$ are the indices for rule and field respectively.

Let $a_{w,s}(i,j)_l$ and $b_{w,s}(i,j)_l$ represent the status of the l^{th} packet matching and not matching an active field $F(i,j)$ in rule $R(i)$, respectively. Where w ($w \in \{1,2,\dots,W\}$), s ($s \in \{1,2,\dots,S\}$) and l($l \in \{1,2,\dots,L\}$) are the window, segment and packet indices, respectively. During the process, when the l^{th} packet matches the field $F(i,j)$ in the rule $R(i)$, the state value of $a_{w,s}(i,j)_l$ is incremented by "1", while $b_{w,s}(i,j)_l$ remains no change. That is:

$$\begin{cases} a_{w,s}(i,j)_l = a_{w,s}(i,j)_{l-1} + 1 \\ b_{w,s}(i,j)_l = b_{w,s}(i,j)_{l-1} \end{cases} \tag{2}$$

By contrast, when the l^{th} packet does not match the field $F(i,j)$in the rule $R(i)$, the state of $b_{w,s}(i,j)_l$is incremented by "1", while $a_{w,s}(i,j)_l$ remains no change. That is:

$$\begin{cases} a_{w,s}(i,j)_l = a_{w,s}(i,j)_{l-1} \\ b_{w,s}(i,j)_l = b_{w,s}(i,j)_{l-1} + 1 \end{cases} \tag{3}$$

Note that if the l^{th} packet is not tested for the field $F(i,j)$ in the rule $R(i)$, the state value of $a_{w,s}(i,j)_l$ and $b_{w,s}(i,j)_l$ remain no change. That is:

$$\begin{cases} a_{w,s}(i,j)_l = a_{w,s}(i,j)_{l-1} \\ b_{w,s}(i,j)_l = b_{w,s}(i,j)_{l-1} \end{cases} \tag{4}$$

Let $C_{w,s}(i)=a_{w,s}(i,M_i)$ and $D_{w,s}(i,j)=b_{w,s}(i,j)$ present the number of packets in the s^{th} segment matching rule $R(i)|_{i=1,2,\dots,N}$ and not matching field $F(i,j)|_{j=1,2,\dots,Mi}$ contained in $R(i)$ on segment basis, respectively.

Therefore, the probability of packet matching rule $R(i)$ on segment basis can be defined as:

$$P_r\left(C_{w,s}(i)\right) = \frac{C_{w,s}(i)}{L} \quad \text{for } 1 \le i \le N \tag{5}$$

Likewise, the probability of packet not matching field $F(i,j)|_{j=1,2,\ldots,Mi}$ in the rule $R(i)$ on segment basis can be defined as:

$$P_r\left(D_{w,s}(i,j)\right)=\begin{cases}\dfrac{D_{w,s}(1,1)}{L} & \text{for } i=1, j=1\\[2ex] \dfrac{D_{w,s}(1,j)}{L-\sum\limits_{k=1}^{j-1} D_{w,s}(1,k)} & \text{for } \mathrm{i}=1, 2\le j\le M_1\\[2ex] \dfrac{D_{w,s}(i,1)}{\sum\limits_{k=1}^{M_{i-1}} D_{w,s}(i-1,k)} & \text{for } 2\le i\le N, j=1\\[2ex] \dfrac{D_{w,s}(i,j)}{\sum\limits_{k=1}^{M_{i-1}} D_{w,s}(i-1,k)-\sum\limits_{k=1}^{j-1} D_{w,s}(i,k)} & \text{for } 2\le i\le N, 2\le j\le M_i\end{cases} \tag{6}$$

More explanation can be found in [1].

At the end of each window there will be an average probability for each rule and field which give us a further indication of the importance of that rule or field, that is:

$$\overline{P_r}\left(R(i)\right)_w=\frac{\sum\limits_{s=1}^{S} P_r\left(C_{w,s}(i)\right)}{S} \quad \text{for } 1\le i\le N \tag{7}$$

$$\overline{P_r}\left(F(i,j)_{R(i)}\right)_w=\frac{\sum\limits_{s=1}^{S} P_r\left(D_{w,s}(i,j)\right)}{S} \quad \text{for} 1\le i\le N, 1\le j\le M_i \tag{8}$$

Where $\overline{P_r}\left(R(i)\right)_w$ and $\overline{P_r}\left(F(i,j)_{R(i)}\right)_w$ are the average probabilities for $R(i)$ and $F(i,j)$ in $R(i)$ in the w^{th} window, respectively.

Reordering Decision

A-Statistical rules reordering decision

Assume that the firewall consists of N filtering Rules with certain order in the previous window $(w\text{-}1)^{th}$. We want to know if this order will be changed or not in the w^{th} window. First let us introduce some notations to be used in Table 1.

Table 1. Previous and current situations for policy filtering Rules

$State \leftharpoondown k \leftarrow$	**R$_1$**	**R$_2$**	…	**R$_N$**	**Total**
Previous(w-1)	$\mathrm{n}_{(w\text{-}1),1}$ $E_{(w\text{-}1),1}$	$\mathrm{n}_{(w\text{-}1),2}$ $E_{(w\text{-}1),2}$		$\mathrm{n}_{(w\text{-}1),N}$ $E_{(w\text{-}1),N}$	$T_{(w\text{-}1)}$
Current(w)	$n_{w,1}$ $E_{w,1}$	$n_{w,2}$ $E_{w,2}$		$n_{w,\mathrm{N}}$ $E_{\mathrm{w},N}$	T_w
Total	C_1	C_2		C_N	T

Let $\mathrm{n}_{(w\text{-}1),i}$ and $n_{w,i}$ (observed values) are the number of matched packets by Rule $R(i)$ in the $(w\text{-}1)^{th}$ and w^{th} windows, respectively. To know if there is a significant difference between the observed and expected values we use the Chi-square test to test the equality of two multinomial distributions. That is:

$$\chi^2(Rules(N)) = \sum_{k=(w-1)}^{w} \sum_{i=1}^{N} \frac{(n_{k,i} - E_{k,i})^2}{E_{k,i}} \tag{9}$$

where $E_{k,i}$ is the expected number of packets to be matched by $R(i)$ in the current or previous window. That is:

$$E_{k,i} = \frac{T_k C_i}{T} \quad \text{for } k = \{w, (w-1)\} \tag{10}$$

If *p_value* ($\chi^2(Rules(N))$,dF)<α → The system is not stable and there is a need to reorder the security policy rules according to histograms of packet matching $R(i)$ on window basis in descending order. The new rule distribution will be computed using the following equation, where $\delta = 1 - (P\_value)$:

$$\overline{P}_r(R(i))_w = \delta\, \overline{P}_r(R(i))_w + (1-\delta)\overline{P}_r(R(i))_{w-1} \tag{11}$$

Otherwise, if *p_value* ($\chi^2(Rules(N))$,dF)>α → The system is stable, no need to reorder the rules. The same previous rule order will be used for the next window and the rules histograms will be renewed using the above equation.

The probability of the current window is given more weight. By doing this the behavior of the traffic in the previous window will not be ignored and will have relatively less effect than the traffic behavior in the current window. As a result the new computed average probabilities would allow producing a better optimized rules ordering for the next window traffic. This procedure will be followed for each rule fields.

B) Statistical policy rule-fields reordering decision

Here, we discuss whether to decide to reorder the policy rules fields or not using the number of packets non-matching field $F(i,j)$ in rule $R(i)$ Where $i \in \{1,2,\ldots,N\}$, $j \in \{1,2,\ldots,M_i\}$. So the same concept of chi-square used in the previous section will be applied for fields of each rule in the security policy as shown in table 2. That is:

$$\chi^2(F(i,j))_{R(i)} = \sum_{k=(w-1)}^{w} \sum_{j=1}^{Mi} \frac{(m_{k,j} - E_{k,j})^2}{E_{k,j}} \tag{12}$$

Where $m_{k,j}$ (observed) is the number of non-matched packets by $F(i,j)$ in $R(i)$, k refers to the current or previous situation. $E_{k,j}$ is the expected number of packets non matching $F(i,j)$ in $R(i)$ in the current or previous window. That is:

$$E_{k,j} = \frac{T_k C_j}{T} \quad \text{for } k = \{w, (w-1)\} \tag{13}$$

Table 2. Previous and current situations for policy filtering Fields for Rule R_i

State(*k*)	$F_{i,1}$	$F_{i,2}$	**…**	$F_{i,Mi}$	**Total**
Previous(*w*-1)	$m_{(w-1),1}$ $E_{(w-1),1}$	$m_{(w-1),2}$ $E_{(w-1),2}$		$m_{(w-1),Mi}$ $E_{(w-1),Mi}$	$T_{(w-1)}$
Current(*w*)	$m_{w,1}$ $E_{w,1}$	$m_{w,2}$ $E_{w,2}$		$m_{w,Mi}$ $E_{w,Mi}$	T_w
Total	C_1	C_2		C_{Mi}	T

If *p_value* ($\chi^2(F(i,j))_{R(i)}$,dF)<α → There is a need to reorder the fields in $R(i)$ according to histograms of packet non matching *F*(*i*,*j*) in *R*(*i*) on window basis in descending order. The new *F*(*i*,*j*) distribution in *R*(*i*) will be computed using the following equation, where $\delta = 1 - (P\_value)$.

$$\bar{P}_r\left(F(i,j)_{R(i)}\right)_w = \delta \bar{P}_r\left(F(i,j)_{R(i)}\right)_w + (1-\delta)\bar{P}_r\left(F(i,j)_{R(i)}\right)_{w-1} \tag{14}$$

Otherwise, if *p_value* ($\chi^2(F(i,j))_{R(i)}$,dF)>α → no need to reorder the fields in *R*(*i*). Rules and rule-fields reordering processes are independent of each other. Depending on $\chi^2(Rules(N))$ and $\chi^2(F(i,j))_{R(i)}$ tests, the system may change the rules order without changing the fields order and vice versa or changing only some rule fields order.

The following algorithms show the main operation of the statistical module. In Algorithm 1 the buildup of the candidate rule list that are independent and equivalent to the original security policy using FDT tool takes place as well as getting the initial rule and rule-field probabilities to start with after training the system S_0 segments. Algorithm 2 is responsible for packet filtering process using function *tuple_comparasion(l).* Then it computes rule and rule-fields statistics (Lines 12-13)

Algorithm 1. Startup Phase

```
1:  <SP>← FDT(Policy Rules)
2:  $s_0$←1
3:  $l_0$←1
4:  **repeat**
5:      inisialize($a_0$,$b_0$)
6:      **while** $l_0$<=$L_0$ **do**
7:          $l_{s0}$←get_pak($f_{s0}$)
8:          tuple_comparasion($l_{ws}$)
9:          $l_0$←$l_0$+1
10:     **end while**
11:     no_mat_$R_s$←last_Field(a)
12:     no_nonmat_$F_{rs}$←r_Field(b)
13:     $pR_s$ ←prob_Rule$_s$(no_mat_$R_s$)
14:     pField$_{rs}$ ←prob_Field(no_nonmat_$F_{rs}$)
15:     $s_0$← $s_0$ +1
16: **until** $s_0$=$S_0$
17: avgp$R_0$ ←avg(sum(p$R_{s0}$)$_{s0=1:S0}$)
18: avgpField$_{r0}$ ←avg(sum(pField$_{rs0}$)$_{s0=1:S0}$)
19: previous_$R_0$ ←sum(no_mat_$R_{s0}$)$_{s0=1:S0}$
20: previous_RField$_{r0}$ ←sum(no_nonmat_$F_{rs0}$)$_{s0=1:S0}$
21: previous_avgp$R_0$← avgp$R_0$
22: previous_avgpField$_{r0}$ ← avgpField$_{r0}$
```

Algorithm 2. System Stability

```
1:w← 1
2:s←1
3:repeat
4:      l←1
5:      repeat
6:              inisialize(a,b)
7:              while l<=L do
8:                      l_ws←get_pak(f_ws)
9:                      tuple_comparasion(l_ws)
10:                     l←l+1
11:             end while
12:             no_mat_R_s←last_Field(a)
13:             no_nonmat_F_rs←r_Field(b)
14:             pR_s ←prob_Rule_s(no_mat_R_s)
15:             pField_rs ←prob_Field(no_nonmat_F_rs)
16:             s← s +1
17:     until s=S
18:     avgpR ←avg(sum(pR_s)_s=1:S )
19:     avgpField_r ←avg(sum(pField_rs)_s=1:S )
20:     //*current state*//
21:     current_R ←sum(no_mat_R_s)_s=1:S
22:     current_RField_r ←sum(no_nonmat_F_rs) s=1:S
23:     current_avgpR← avgpR
24:     current_avgpField_r ← avgpField_r
25://*Rule-Fields stability test*//
26:     foreach r ϵ R
27:             p_value_r←chi_square(previous_RField_r    ,
        current_RField_r)
28:             δ_r ←1- p_value_r
29:             current_avgpField_r←(1- δ_r)*
                current_avgpField_r + δ_r *
                previous_avgpField_r
30:             if p_value_r<α
31:                     Rule_r ←reorder(rule_r,
        current_avgpField_r)
32:             end if
33:     end for
34://*Rules stability test*//
35:     p_value_R←chi_square(previous_R, current_R)
36:     current_avgpR←(1- δ_r)* current_avgpR +
δ_r * previous_    avgpR
37:     if p_value_R< α
                reorder(R, current_avgpR)
38:     end if
39: /*update the previous state*/
40:     previous_R ← current_R
41:     previous_RField_r ← current_RField_r
42:     previous_avgpR← current_avgpR
43:     previous_avgpField_r ← current_avgpField_r
44:     w← w +1
45:until w=W
```

and calculate the corresponding segment probabilities (Lines 14-15). Lines (18-19) compute the average rule and rule-fields probability on window basis. $\chi^2(F(i,j))_{R(i)}$ and $\chi^2(Rules(N))$ are computed in (Lines 25-38). Also, the re-ordering process for rule-fields and rules is done in a descending manner according to current average probability based on if statement in lines 30 and 37. The current rule-fields and rule average probability are computed in (Lines 29 and 36). In (Lines 38-42) the previous state variables are updated to be used in the next traffic window.

4 Evaluation

4.1 DR-RFOD vs DR-RFO

In order to evaluate the performance of the proposed mechanism, an algorithm that dynamically changes the order of the rules and rule-fields according to system stability has been implemented using MATLAB programming environment. This experiment is done using 20 filtering rules and 200 traffic windows each of 1000 packets. These numbers are used just to make it easy to trace the rules and fields ordering position process using both DR-RFO and DR-RFOD mechanisms.

DR-RFOD vs DR-RFO According to Rules Reordering Process

The algorithms in DR-RFO and DR-RFOD mechanisms start optimizing the rule positions after treating the second window. In DR-RFO mechanism positions of the rules are updated dynamically after treating each window. On the other hand, in DR-RFOD mechanism positions of the rules are updated dynamically according to eq. (9) and eq. (11) after system stability test. Fig.1 compares the evolution in R1 as an example using DR-RFO and DR-RFOD mechanisms. The horizontal constant lines in the figure shows the corresponding windows for DR-RFOD mechanism where the system was stable according to eq. (9) and no rule reordering process is done.

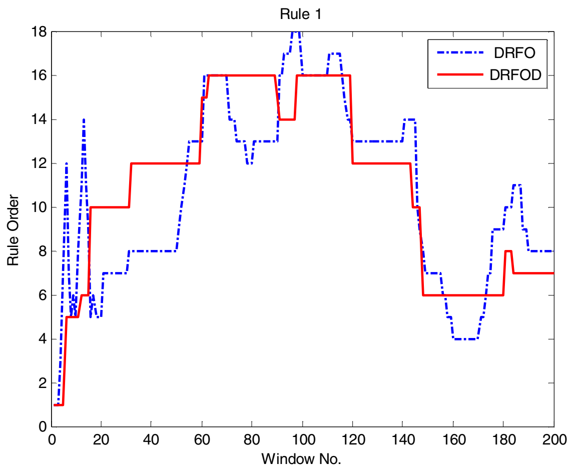

Fig. 1. Evolution in R1 using DR-RFO and DR-RFOD

DR-RFOD vs DR-RFO According to Rule-Fields Reordering Process
The same concept used in rules reordering process will be used in rule-fields reordering process. In DR-RFO mechanism positions of the rule-fields are updated dynamically after treating each window. On the other hand, in DR-RFOD mechanism positions of the rule-fields are updated dynamically according to eq. (12) and eq. (14) after system stability test. Fig.2 compares the evolution for field Source-IP in R1 as an example using DR-RFO and DR-RFOD mechanisms.

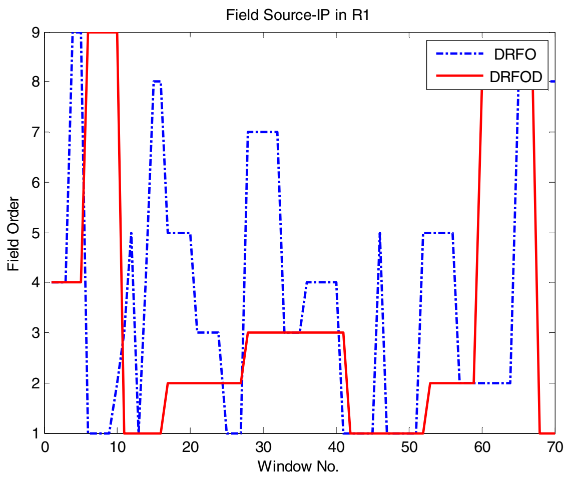

Fig. 2. Evolution in Field Source-Port in R1 using DR-RFO and DR-RFOD

Fig. 3 shows the cumulative processing time for DR-RFO and DR-RFOD for different values of α. For α=0.005, the gain for using DR-RFOD for 200 traffic windows is 9.1119(s), while for α=0.05 the gain is18.2855(s).

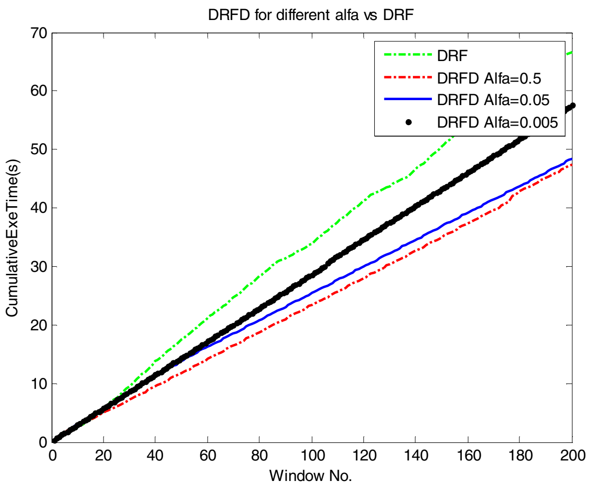

Fig. 3. Cumulative processing time for DR-RFOD vs DR-RFO for different α values

4.2 The Effect of Error Precision (α) on DR-RFOD Mechanism

This experiment studies the effect of different α values in the cumulative execution time and the number of rule/rule-fields reordering process. Fig.4 gives an idea about the execution time needed for each of the 200's windows for different α values.

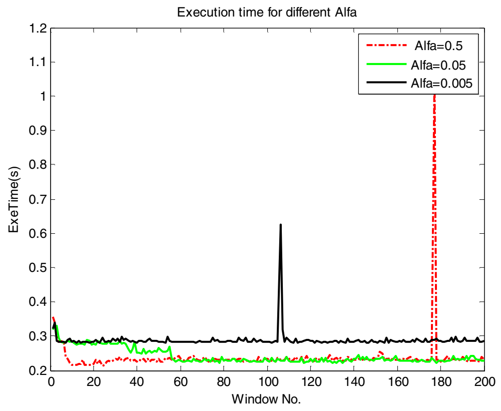

Fig. 4. Execution time for DR-RFOD for different α values

Table 3 compares different values of α and their corresponding number of rule/rule-field reordering process. When α decreases: 1) The frequency of the reordering process is also decreased this is because for a given computed χ^2, decreasing the value of α will increase χ^2_α ending with χ^2_α >computed χ^2 and therefore no need for reordering. 2) The cumulative execution time increases. This is because in fact when we decide not to reorder we might keep the system running with a non-efficient configuration (order) for longer time and therefore it might take longer execution time than when we reorder often especially if the cost of re-ordering is small and this depends on the number of rules and rule-fields in the security policy.

Table 3. The effect of different alfa in reordering frequency and cumulative processing time

α	**No. Reordering Rules/Fields**			**Cumulative Processing Time (s)DR-RFOD**	**≈ (s)**
	R	**RF**	**Total**		
o▷5	102	198	300	47.4047	
o▷4	80	151	231	47.8677	
o▷3	63	124	187	47.9697	47-48
o▷2	47	80	127	48.0900	
o▷1	26	36	62	48.1036	
o▷05	14	28	42	48.3715	
o▷04	11	29	40	49.3784	
o▷03	10	27	37	49.7656	48-57
o▷02	9	18	27	49.9930	
o▷01	6	3	9	57.3541	
o▷005	4	0	4	57.5448	
o▷004	4	0	4	57.5537	
o▷003	3	0	3	57.7527	57-60
o▷002	1	0	1	58.1344	
o▷001	1	0	1	60.5990	

5 Conclusion

In this paper, we have proposed a mechanism to improve firewall packet filtering time through optimizing the order of security policy filtering rules and rule-fields. The proposed mechanism is based on reordering rules and rule-fields according to packet matching and non-matching histograms, respectively. The current and previous traffic windows statistics are used to check the system stability using Chi-Square Test. If the system stability test indicates that the firewall is stable the same current rule and/or rule-fields orders are used for filtering the next traffic window. Otherwise, an update of the rule and/or rule-fields order structures is required for filtering the next traffic window. The proposed mechanism gives better cumulative execution time compared to DR-RFO mechanism. Also, the effect of α on the cumulative processing time and on the frequency of the reordering process has been discussed. In future work, we intend to investigate the effect of dynamically changing the traffic window size, and improve the proposed mathematical model to take into consideration security policy with dependent rules.

Acknowledgment. The authors acknowledge the support of NRF Foundation and Emirates Foundation through researches grants 21T023 and 2009/161, respectively.

References

1. Trabelsi, Z., Zhang, L., Zeidan, S.: Packet Flow Histograms to Improve Firewall Efficiency. In: ICICS (December 2011)
2. Trabelsi, Z., Zeidan, S.: Multilevel Early Packet Filtering Technique based on Traffic Statistics and Splay Trees for Firewall Performance Improvement. In: ICC (June 2012)
3. Lan, K., Heidemann, J.: On the correlation of internet flow characteristics. Technical Report ISI-TR-574, USC/ISI (2003)
4. El-Atawy, A., Samak, T., Al-Shaer, E., Li, H.: Using online traffic statistical matching for optimizing packet filtering performance. In: IEEE INFOCOM 2007, pp. 866–874 (2007)
5. Gupta, P., McKeown, N.: Algorithms for packet classification. IEEE Network 15(2), 24–32 (2001)
6. Baboescu, F., Varghese, G.: Scalable packet classification. In: ACM SIGCOMM 2001 (2001)
7. McAulay, A.J., Francis, P.: Fast routing table lookup using CAMs. In: IEEE INFOCOM 1993 (March 1993)
8. Srinivasan, V., Suri, S., Varghese, G.: Packet classification using tuple space search. In: Computer ACM SIGCOMM Communication Review, pp. 135–146 (October 1999)
9. Feldmann, A., Muthukrishnan, S.: Tradeoffs for packet classification. In: IEEE INFOCOM 2000 (March 2000)
10. Gupta, P., McKeown, N.: Packet classification using hierarchical intelligent cuttings. In: Interconnects VII (August 1999)
11. Cohen, E., Lund, C.: Packet classification in large isps: design and evaluation of decision tree classifiers. In: SIGMETRICS 2005: Proceedings of the 2005 ACM SIGMETRIC International Conference on Measurement and Modeling of Computer Systems, pp. 73–84. ACM Press, New York (2005)

12. Woo, T.Y.C.: A modular approach to packet classification: Algorithms and results. In: IEEE INFOCOM 2000, pp. 1213–1222 (March 2000)
13. Gupta, P., Prabhakar, B., Boyd, S.: Near optimal routing lookups with bounded worst case performance. In: IEEE INFOCOM 2000 (2000)
14. Kencl, L., Schwarzer, C.: Traffic-adaptive packet filtering of denial of service attacks. In: WOWMOM 2006: The 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks, Washington, DC, USA, pp. 485–489 (2006)
15. Acharya, S., Abliz, M., Mills, B., Znati, T.F.: Optwall: a hierarchical traffic-aware firewall. In: Proceedings of 14th Annual Network & Distributed System Security Symposium (NDSS), San Diego, US (February 2007)
16. Hamed, H., Al-shear, E.: Dynamic Rule-ordering optimization for High-speed Firewall Filtering. In: ASIACCs 2006, Tuipei, Taiwam, March 21-24 (2006)
17. Hamed, H., El-Atawy, A., Al-Shaer, E.: On Dynamic Optimization of Packet Matching in High-Speed Firewalls. IEEE Journal on Selected Areas in Communications 24(10) (October 2006)
18. Al-Shear, E., El-Atawy, A., Tran, T.: Adaptive Early Packet filtering for Defending firewalls against DoS Attack. In: Proceeding of IEEE INFOCOM, pp. 1–9 (2009)
19. Waldvogel, M., Varghese, G., Turner, J., Plattner, B.: Scalable High Speed IP Routing Lookups. In: Proceedings of the ACM SIGCOMM (SIGCOMM 1997), pp. 25–36 (1997)
20. Sleator, D., Tarjan, R.: Self Adjusting Binary Search Trees. Journal of the ACM 32(3), 652–686 (1985)
21. Neji, N., Bouhououla, A.: Dynamic Scheme for Packet Classification Using Splay trees. Information Assurance and Security, 1–9 (2009)
22. Hamed, H., El-Atawy, A., Al-Shaer, E.: Adaptive statistical optimization techniques for firewall packet filtering. In: IEEE INFOCOM 2006 (April 2006)
23. Mothersole, I., Reed, M.: Optimizing Rule Order for a Packet Filtering Firewall. In: SAR-SSI (2011)
24. Wang, W., Chen, H., Chen, J., Liu, B.: Firewall rule Ordering based on statistical Model. In: International Conference on Computer Enginnering and Technology (2009)
25. Wang, W., Ji, R., Chen, W., Chen, B., Li, Z.: Firewall Rules Sorting Baseb on Markov Model. In: Procedings of the International Symposium on Data Privacy and E-Comerce (2007)
26. Liu, A., Gouda, M.: Complete Redundancy Detection in Firewalls. In: Jajodia, S., Wijesekera, D. (eds.) Data and Applications Security 2005. LNCS, vol. 3654, pp. 193–206. Springer, Heidelberg (2005)