Optimization of Firewall Rules

Tihomir Katić Predrag Pale
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, HR – 10000 Zagreb, Croatia
tihomir.katic@fer.hr predrag.pale@fer.hr

Abstract: *Network performance highly depends on efficiency of the firewall because for each network packet which enters or leaves the network a decision has to be made whether to accept it or reject it. This paper presents one approach to rule optimization solutions for improving firewall performance. The new software solution has been developed based on relations between rules. Its main purpose is to remove anomalies in ordering of Linux firewall rules and to merge similar rules.*

Developed rule optimization software (FIRO) is intended to be used with IP Tables Linux firewall command tool, but it can be easily adapted for other tool, as well. FIRO works in several passes through revised rule lists. In each step of optimization process FIRO generates a different rule list. Unlike existing solutions, FIRO also analyzes log rules and takes into account other rule parameters besides IP addresses, ports, protocols and action.

Keywords: firewall, rules, optimization, relations, anomalies, policy

1. Introduction

In order to ensure security of a private network the traffic needs to be controlled. Network firewall is a system which limits network access to and from a network. It is usually positioned between a trusted, protected private network and an untrusted, public network. Firewall allows only approved traffic in and out according to a thought-out plan (firewall policy). In its work firewall minimizes security threats which range from curious prowlers to well-organized, technically knowledgeable intruders that could gain access to private information or interfere with user's legitimate use of system.

Different organizations don't have equal needs from the point of allowed traffic. What is allowed in one organization doesn't have to be allowed in another. According to different needs, different firewall policies must be created. Firewall policies don't only differentiate between organizations – they also change over time. Furthermore, different administrators would most likely create different rule lists even for the same policy.

For each type of network traffic, there is one or more different rules. Every network packet, which arrives at firewall, must be checked against defined rules until first matching rule is found. The packet will be then allowed or banned access to the network, depending on the action specified in the matching rule.

If number of different rules is large, which is normal for large and complex organizations, administrators are not able to efficiently create new and remove existing rules. Even experienced administrators find it difficult to compare new rule against all existing rules and to detect all redundancies. Chances to find and remove redundant rules are reduced even further if firewall administrators change. Also, as the firewall administrator is less experienced, the larger number of redundant rules can be expected.

Redundant rules are called anomalies. This is the situation in which a redundant rule is never going to be matched because it is preceded by some other rule or rules, which are matching all of packets for which that redundant rule was designed. Packets which could be matched by redundant rule are not a problem for firewall performance because they are matched earlier. However, all other network packets which do reach redundant rule are being unnecessarily checked which reduces performance.

Degradation of firewall performance is also achieved if a rule is unnecessarily divided in two or more different rules. In those situations network packets which could be examined by

only one rule, will be checked against several rules in sequence.

This paper describes one aspect of firewall optimization based on removing unnecessary rules without changing firewall policy thus enhancing the performance of the firewall. Object of optimization are list-based firewalls where network packet is being checked against rules in a list until matching rule is found or end of list is reached. FIRO software, developed for rule optimization, is customized for Linux IP Tables firewall command tool, but it can easily be adapted for other list-based firewalls.

The main goal of this project was to develop software which would not require administrator to understand optimization procedures in detail. After each optimization step, software must create new firewall configuration. The list can then be checked by an administrator who can easily detect changes and reasons for those changes.

2. Firewall rules

Different firewalls usually provide different rule logic with different parameters. But some basic elements are common to all. They all allow an action to be defined allowing or banning specific network traffic. Also, all of them allow checking for most important elements in packets like IP addresses, ports and protocol.

Software for firewall rule optimization (FIRO) was originally developed for IP Tables firewall command tool. One of the most important functionalities of IP Tables firewall is stateful inspection. Stateful inspection automatically opens only the ports necessary for internal packets to access the Internet. It only allows transfer of packets which are defined in firewall rules and which are part of established connections.

2.1. Firewall chains

IP Tables group rules in chains. Different network packets are processed by different chains:

- incoming traffic – packets for firewall (INPUT chain),
- forwarding traffic – incoming packets for another machine (FORWARD chain),
- outgoing traffic – packets generated by firewall (OUTPUT chain).

For each of these chains, FIRO creates an optimized set of rules.

2.2. Rule parameters

Each rule identifies specific type of network traffic. In order to enable this identification parameters for identification of specific network packets must be set for each rule.

FIRO provides optimizing procedure which is based on these parameters:

- IP addresses – it can be destination or source IP address; also, it can be written as a single IP, network IP or IP range,
- Ports - it can be destination or source port; also, it can be written as a single port, port range or port array,
- Protocol – it can be referred to TCP, UDP, ICMP or all together,
- Interface – it can be incoming or outgoing interface,
- TTL (Time To Live) field residing in the IP headers,
- ToS (Type of Service) field residing in the IP headers,
- Length of packet,
- MAC source address,
- Syn flag – identification of new connection,
- ICMP type,
- Limit – maximum number of packets in time interval.

Although FIRO allows use of all parameters, in real environment commonly used parameters are: source and destination IP addresses, destination port which defines service or application, and protocol.

2.3. Using relations between rules

Firewall administrators are trying to define rules which will be mutually independent but sometimes they fail. One rule can be in relation to another rule if they are matching the same network packets – matching isn't necessarily complete, but if some amount of overlapping between network packets checked by 2 rules exists then it can be said that those 2 rules are not independent. This precedence relationship between rules in a policy can be modeled as a Directed Acyclical Graph [2].

Two rules are in relation if intersection between every 2 of their parameters exists. If an

intersection doesn't exist, than rules are not in relation. Example of related rules is shown in Figure 1. Those parameters which are not specified in rules are considered to have value "ALL".

```
R1: iptables -A FORWARD -p TCP
      -s 10.0.0.0/24  -d 10.1.0.10
      --dport 80 -j DROP
R2: iptables -A FORWARD -p ALL
      -s 10.0.0.0/16  -d 10.1.0.10
      -j ACCEPT
R3: iptables -A FORWARD -p ALL
      -s 10.0.1.0/24 -j DROP
```

Figure 1. Related IP Tables rules

If two rules are in relation but have different actions (one rule allows network traffic, another denies it), it is important to preserve rule order between them in optimization procedure. Otherwise, resulting optimized firewall policy wouldn't behave as its original. For that reason it is important to have references between related rules. If a rule can be removed, another rule can take references of a deleted rule, if it doesn't already contain them. Relations between rules shown in Figure 1 are presented in Figure 2. Rules are presented as cycles while relations between them are shown as arrows connecting cycles. Order of related cycles must be equal to order in firewall policy list. Every rule must be presented with only one cycle. In search for related rules each rule must be compared with all the other rules and, as it is mentioned before, two rules are in relation if intersection between every 2 of their parameters exists. In Figure 2 all rules are in relation except rule R1 and R3 because intersection of their source addresses is zero.

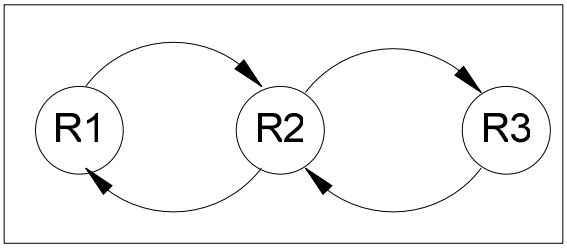

Figure 2. Relations between rules

3. Policy anomalies

Firewall policy anomaly occurs when rule list contains two or more rules which match the same packet. Also, it can occur when a rule never matches any packet that goes through the firewall. It is expected that the number of anomalies will increase as the number of rules in firewall policies grows. When number of rules is large, administrators are not able to effectively estimate effect of new rules on existing rules. This leads to anomalies.

Firewall policy anomalies can be classified in five groups [1]:

1) Shadowing anomaly – rule is shadowed by a preceding rule which matches all the packets that would match this shadowed rule. Thus the shadowed rule will never be activated.

2) Correlation Anomaly — two rules are correlated if they have different filtering actions, and the first rule matches some packets that match the second rule and the second rule matches some packets that match the first rule.

3) Generalization Anomaly — rule is a generalization of a preceding rule if they have different actions, and if the first rule can match all the packets that match the second rule.

4) Redundancy Anomaly — rule is redundant if there is another rule that produces the same matching and action such that if the redundant rule is removed, the security policy will not be affected.

5) Irrelevance Anomaly — filtering rule in a firewall is irrelevant if this rule does not match any traffic that may flow through this firewall. This exists when both the source address and the destination address fields of the rule do not match any domain reachable through this firewall.

FIRO does not treat all anomalies in rule list. Reason for that is that not all of them are considered to be anomalies in real environments (correlation and generalization), and others are just not applicable for static optimization of rules (irrelevance).

Correlation anomaly wasn't considered to be anomaly because administrators often define rules which are correlated. Reason for that is that in order to avoid definition of two or more "smaller" rules that are not correlated, firewall administrators often define a single rule even though they are aware that this rule will correlate with some existing rule. If rule is divided, firewall performance will be decreased.

Generalization anomaly wasn't also considered to be anomaly because administrators often define some special cases which must be different than some global rule. That global rule could be divided in two or more "smaller" rules

but it would also cause decrease in firewall performance.

Irrelevance anomaly is considered to be anomaly, but detection of this anomaly type is not applicable to developed software which operates only with firewall rules as this software doesn't contain information about neighboring networks or services.

4. Removing policy anomalies

In the first pass through the rule list FIRO creates Directed Acyclical Graph with relations between rules. Also it removes redundant rules which are either shadowed or which are followed by a rule which matches all the packets that match the redundant rule and have the same action as the redundant rule.

```
R1: iptables -A FORWARD -p ALL
      -i eth0  -s 10.0.0.0/24  -o eth1
      -d 10.1.0.5  -j ACCEPT
R2: iptables -A FORWARD -p ALL
      -s 172.20.0.0/24  -j ACCEPT
R3: iptables -A FORWARD -p ALL
      -s 10.0.0.0/24 -j DROP
R4: iptables -A FORWARD -p ALL
      -d 10.1.0.5 -j ACCEPT
R5: iptables -A FORWARD -p ALL
      -s 172.20.0.0/28 -d 10.1.0.10
      -j DROP
```

Figure 3. IP Tables rules with shadowing anomaly

In Figure 3 rules R2 and R5 present shadowing anomaly. All the packets with source IP address from network 172.20.0.0/24 are matched by rule R2. Rule R5 can only match packets which are a part of packets from network 172.20.0.0/24. This example represents shadowing anomaly and rule R5 must be removed.

Relations between rules are not very important for shadowing anomaly. But they are important for redundant rules which are followed by some rule which matches all the packets that match the redundant rule. If rule R1 was preceeded by rule R4, that would present a shadowing anomaly and it would be removed. Because rule R1 preceeds rule R4, other check procedures must be performed. Rule R1 can be removed if other rules which have different action than rule R1 and R4 don't exist between rules R1 and R4. As it is presented in Figure 4, between rules R1 and rule R4 there is a rule R3 which has a different action. Because of that rule R1 can not be removed.

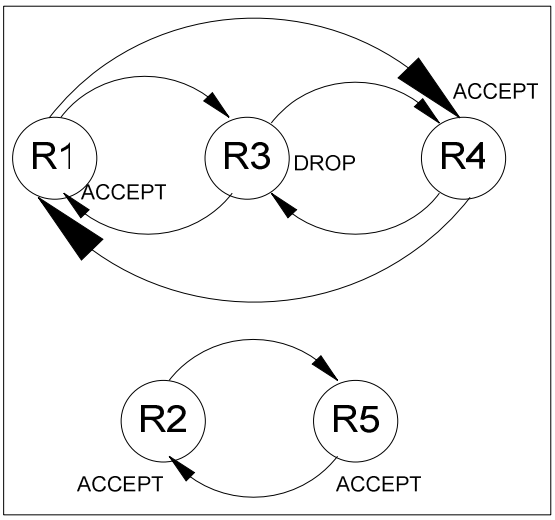

Figure 4. Relations between five rules

Algorithm for removing redundant rules defines that in search for redundant rules it is necessary to compare only those rules which are related. For faster anomaly discovery and removal it is necessary to maintain relational graph of rules. By using graph not all rules are compared but only related ones. It allows quicker finding of anomalies which can or can not be removed.

5. Merging firewall rules

Firewall rules can be even further optimized even if they don't contain anomalies. Those are the cases where two or more rules can be joined in one single rule. To join different rules, all of rule parameters must be equal except one parameter. After that parameter is found, optimization procedure is trying to join different values of those parameters in one acceptable parameter.

FIRO is trying to join different rules according to protocol, source and destination ports and IP addresses. Protocol optimization procedures are trying to join similar rules for TCP, UDP and ICMP protocols. Port optimization procedures are trying to join rules in one single rule which would have acceptable port array or port range. IP address optimization procedures are trying to create one rule from more rules with acceptable IP range or IP network.

```
R1: iptables -A FORWARD -p ALL
      -s 10.0.0.0/24  -j ACCEPT
R2: iptables -A FORWARD -p ALL
      -s 10.0.1.0/24  -j ACCEPT

R_join = R1 + R2:
    iptables -A FORWARD -p ALL
      -s 10.0.0.0/23 -j ACCEPT
```

Figure 5. Example of merging firewall rules

In Figure 5 rules R1 and R2 match different types of network traffic so they are not in relation. Rule R1 matches traffic with source IP address from network 10.0.0.0/24, and rule R2 traffic with source IP address from network 10.0.1.0/24. Using merging procedure those two rules are replaced with rule R_join which matches traffic with source IP address from network 10.0.0.0/23 which consists of networks 10.0.0.0/24 and 10.0.1.0/24.

When different rules are merged in one single rule, that new rule inherits references of merged rules to some other rules (Rx). Also on those other rules references which pointed to old rules must be changed to point to newly created rule.

After initial pass in which some rules were merged FIRO reactivates procedure for removing policy anomalies. That is necessary because after merging procedure new rules are created which match larger amount of traffic and it is possible that they are shadowing some other "smaller" rules.

6. Special cases

Main functions of developed software are removing anomalies and merging similar firewall rules. But after those optimizations are performed, FIRO also searches for some special cases. Those specific procedures are developed for log rules and for limit options.

6.1. Log rules

Using log rules in firewall policies which are being optimized is not an ideal situation. Log rules require creating log records which must be written on disc system or sent to log server. And those services are time and performance expensive.

When administrators create redundant rules which will accept or reject some network traffic, it is obvious that those rules are not necessary for firewall policy. But when administrators create redundant log rules, it is possible that they didn't make mistake, but that they have done it on purpose. One of the reasons why administrators would do that is because they want to have duplicate log records of some data. Some global log rule contains only IP options whereas some specific log rule contains IP and TCP options.

Even if log rules match some network traffic they don't finish processing of that traffic in firewall chain. Processing of matched traffic is finished only by rules which accept or deny traffic. For that reason log rules in IP Tables firewall are positioned before rules which will accept or deny traffic which is being logged. If administrator makes mistake and places them after the corresponding rule which will accept or deny traffic, that mistake will be corrected by optimization software.

FIRO discards log order from initial firewall policy. It tries to place log rules as far as possible in the rule list because log rules are degrading firewall performance and they must be processed as late as possible.

Procedure for optimizing log rules searches for first accepting or denying rule which matches the same traffic as processed log rule. When it finds it, it places log rule just before it so traffic can be logged before it is accepted or denied.

6.2. Limit options

One of possible rule parameters is limit option. Number of limited packets can be expressed in seconds, minutes, hours and day time units. This match can, for example, be used to limit the amount of logging of specific network traffic. It is possible to limit how many times a certain rule may be matched in a certain time frame, for example to lessen the effects of DoS (Denial of Service) flood attacks. This is its main usage, but there are more usages.

If two rules are equal in all parameters or if second rule is a subset of the first rule, those rules present shadowing anomaly. But if limit parameter is different for those two rules, procedure for removing anomalies won't detect that anomaly. And that behavior is not wanted.

Limit option isn't like other parameters. Its value isn't found in network packet. Its value is related to action parameter which can be accepting, denying or logging specific network packet. Due to that fact it has to be processed in different way.

If one rule is shadowed by another rule, and they have different limit options, shadowed rule will be removed. Also, only if two rules have

equal limit parameters, they can be merged if other rule parameters allow it.

7. Further work

Optimization functionalities which are described in this paper are completed and implemented. Next step in developing software is to include NAT (Network Address Translation) rules in optimization procedures. Subsequently log record analysis should be performed in order to determine which rules have the largest possibility to be realized. It is assumed that in real environment network traffic is changed over time so rule order would also have to be changed.

Finally, when FIRO will be changed to include traffic-rule probabilities, it will also perform log record analysis with data mining procedures which would search for specific network traffic for which it would be useful to create separate rules.

8. Conclusion

Managing firewall policy rules is becoming increasingly complex and it requires effective administration and control. Firewall administrators are not able to compare all of the defined rules, and in order to meet different organizational needs, they have to create new rules which can be redundant compared to the existing ones. As needs for high performance firewalls grow, optimization procedures must also be more innovative and effective.

This paper presents static optimization procedures for removing redundant rules and for merging similar rules. Optimization also includes log rules and other rule parameters apart from IP addresses, ports, protocols and action which is not common in other optimization solutions. The effect which these new optimization procedures can have in real environment depends on number of rules in firewall policy and on experience of firewall administrator.

FIRO, software which was developed in C++ is able to be activated both on Linux and on Windows operation systems. Optimization procedures, described in this paper, result with multiple different firewall policy files where each file represents one step in optimization procedure. In that way administrators can more easily understand why some rules disappeared. Compared to Firewall Policy Advisor [1], FIRO is not interactive program and it doesn't allow inserting of new rules and detection of every type of anomaly. Compared to some other solutions [8], it doesn't provide traffic analysis, but it provides optimization of log rules and rules based on larger number of parameters.

FIRO can be very useful for administrators which are not very experienced or which have to administer large number of firewall rules. But if administrator is able to create optimized set of rules, FIRO is useless. For those reasons FIRO must be upgraded to include traffic analysis [8]. Even most experienced administrators are not able to accurately predict different probabilities of traffic during different periods of time.

9. References

[1] Al-Shaer E. S, Hamed H. H. Modeling and Management of Firewall Policies. IEEE Transactions Network and Service Management, vol. 1. no 1; 2004.

[2] Fulp E. W. Optimization of Network Firewall Policies Using Ordered Sets and Directed Acyclical Graphs. Proceedings of the IEEE Internet Management Conference (IM'05), Nice, France; 2005.

[3] Golnabi K, Min R. K, Khan L, Al-Shaer E. S. Analysis of Firewall Policy Rules Using Data Mining Techniques. Network Operations and Management Symposium 2006: Vancouver, Canada; 2006 April 3-7.

[4] Chomsiri T, Pornavalai C. Firewall Rules Analysis. The 2006 World Congress in Computer Science Computer Engineering, and Applied Computing: Las Vegas, USA; 2006 June 26-29.

[5] Al-Shaer E. S, Hamed H. H. Dynamic Rule-ordering Optimization for High-speed Firewall Filtering. ACM Symposium on InformAtion, Computer and Communications Security: Taipei, Taiwan, 2006 March 21-24.

[6] Al-Shaer E. S, Hamed H. H. Discovery of Policy Anomalies in Distributed Firewalls. Proceedings of the IEEE INFOCOM 2004, vol 23, no 1. Hong Kong, China; 2004.

[7] Hari B, Suri S, P.arulkar G. Detecting and Resolving Packet Filter Conflicts. Processing of 5th International Network Conference. Samos Islands, Greece; 2005 July.

[8] Acharya S, Wang J, Ge Z. Traffic-Aware Firewall Optimization Strategies, University of Pittsbourgh, 2005.