Analysis of Firewall Policy Rules Using Data Mining Techniques

Korosh Golnabi, Richard K. Min, Latifur Khan
Department of Computer Science
The University of Texas at Dallas
Richardson, USA
(koroshg, rkm010300, lkhan)@utdallas.edu

Ehab Al-Shaer
School of Computer Science, Telecommunications and Information Systems
DePaul University
Chicago, USA
ehab@cs.depaul.edu

***Abstract*— Firewall is the *de facto* core technology of today's network security and defense. However, the management of firewall rules has been proven to be complex, error-prone, costly and inefficient for many large-networked organizations. These firewall rules are mostly custom-designed and hand-written thus in constant need for tuning and validation, due to the dynamic nature of the traffic characteristics, ever-changing network environment and its market demands. One of the main problems that we address in this paper is that how much the firewall rules are useful, up-to-dated, well-organized or efficient to reflect the current characteristics of network traffics. In this paper, we present a set of techniques and algorithms to analysis and manage firewall policy rules: (1) Data Mining technique to deduce efficient firewall policy rules by mining its network traffic log based on its frequency, (2) Filtering-Rule Generalization (FRG) to reduce the number of policy rules by generalization, and (3) a technique to identify any decaying rule and a set of few dominant rules, to generate a new set of efficient firewall policy rules. The anomaly detection based on the mining exposes many hidden but not detectable by analyzing only the firewall policy rules, resulting in two new types of the anomalies. As a result of these mechanisms, network security administrators can automatically review and update the rules. We have developed a prototype system and demonstrated usefulness of our approaches.**

Keywords- firewall; data mining; network security; policy

I. INTRODUCTION

In the global world of Internet, firewall is the *de facto* core technology of today's network security and first line of defense against external network attacks and threats. Firewall controls or governs network access by allowing or denying the incoming or outgoing network traffic according to firewall policy rules. These rules are explicitly written and managed to filter out any unwanted traffic coming into or going from the secure network. However, the management of firewall rules has been proven to be complex, error-prone, costly and inefficient for many large-networked organizations. These firewall rules are often custom-designed and hand-written by and for the human policy writer of an organization and tailored to accommodate ever-changing business and market demands of the global Internet. Therefore, these rules are in a constant need of updating, tuning and validating to optimize firewall security.

Although the deployment of firewall technology is the first important milestone toward securing networks, the effectiveness of firewall security may be limited or compromised by a poor management of firewall policy rules.

One of the interesting problems is on how much the rules are useful, up-to-dated, well-organized or efficient to reflect current characteristics and volume of network packets. For example, the network traffic trend may show that some rules are out-dated or not used recently. This may further lead one to consider removing, aggregating or reordering of the rules to optimize the firewall policy and efficiency. Also, server and network logs may validate or confirm that firewall policy rules are updated and consistent with the current network services.

More importantly, data mining over network traffic log may further reveal any violation against the current firewall polices such as transferring plain text over a secure IPSec/VPN link or allowing traffic which may not be permitted by the other downstream devices behind firewalls. Furthermore, the task of manually-managing firewall policy rules becomes very difficult and time-consuming, if not impossible, as the number of filtering rules increases drastically beyond the reasonable scope and scale of a manual process. This enormous task addresses the need for the effective management of firewall security with policy management techniques and tools that enable network administrators with ease to optimize and validate firewall rules automatically.

The first step to bridge the gap between what is written in the firewall policy rules and what is being observed in the network is to analyze traffic and log of the packets using data-mining techniques. Current research is extending the traditional and classical research in the analysis and management of the these policy rules, not only to minimize the number of rules but to generate an efficient rule-set, to reflect the up-to-dated traffic trend, and further to provide a real-time policy update capability from the application (for example, mail server detecting a massive incoming spam-mails from a host).

In this paper, we present a set of techniques and algorithms to analysis and manage firewall policy rules: (1) a data mining technique, Mining firewall Log using Frequency (MLF) to deduce efficient firewall policy rules by mining its network traffic log based on its frequency, (2) Filtering-Rule Generalization (FRG) to reduce the number of policy rules by

1-4244-0143-7/06/$20.00 ©2006 IEEE.

generalization, and (3) a technique to identify *decaying rule* and *dominant rule*, to generate a new set of efficient firewall policy rules. The technique proposed in this paper provides an automated tool to discover frequent traffic patterns and filtering rules, using Log Mining Frequency and Filtering-Rule Generalization, to discover and generate an effective and anomaly-free firewall policy rules.

Also there has been a significant research in policy anomaly detection area [1, 2, 10]. Unlike our approach, most of the current work is based on static analysis of the existing policy configuration. Considering traffic and log mining enables discovering other important policy properties which are unrevealed by other approaches. In addition, the anomaly detection based on the mining exposes many hidden that are not detectable by analyzing only the firewall policy rules.

This paper is organized into the following sections. First, a summary of related work is presented in Section II. In Section III, the process of to analysis and manage firewall policy rules will be presented including Data Mining technique for policy deduction, rule generalization, and (3) techniques to identify decaying as well as dominant rules. In Section IV, the result of the experimental design and implementation using Linux operating system firewall is illustrated as an example. Finally in Section V, the conclusion of this research and potential areas for future work are presented.

II. Related Work

There has been a great amount of research and work in the area of firewall and policy based security management. Several models have been proposed for the anomaly detection using its firewall policy rules. However, in our work, we use data mining technique to generate firewall policy rules. Then these rules are generalized via a generalization model and further an anomaly discovery algorithm is applied to the rules. Two main areas related to our work are (1) data mining and (2) firewall policy rule anomaly detection. In the following section, we focus on related works on these two areas.

A. Data Mining Techniques

Data mining is a predictive model and technique to explore large amount of datasets in search of finding an interesting trend or a pattern in a large dataset to guide decision for further analysis. There are a few approaches such as decision tree [11], and Association Rule Mining (ARM) [13].

Decision Tree learns a function represented as a decision tree. Each node in the tree tests as an attribute, each branch corresponds a value comparison, and each leaf node assigns classification. The disadvantage of this approach includes handling of continuous attributes, growing tree, and computational efficiency. Association Rule Mining algorithm searches the space of all possible patterns for rules that meet the user-specified support and confidence thresholds. One example of an association rule algorithm is the Apriori algorithm designed by Srikant and Agrawal [15] and a comprehensive survey of for Association Rule Mining can be found in [18]. A few recent and noteworthy advances in the area of intrusion and anomaly detection using data mining techniques, especially with Association Rule Mining technique, pioneered by Lee [32] and Mahoney [33].

Association Rule Mining (ARM) is a nontrivial process of identifying valid, potentially useful, and ultimately desirable pattern (rule) in data. An Apriori algorithm may result in a large number of rules which may turn out to be useless or impractical for analysis. Further some rules may have no impact on firewall action such as repeated rules or rules which does not yield any action on the right hand side. As a result, these rules are filtered for further analysis. The major objective is to generalize specific and unique rules to more general rules. Thus Association Rule Mining algorithms typically only identify patterns that occur in the original form throughout the database. One limitation of many association rule mining algorithms such as the Apriori algorithm is that only the database entries of the exact match for the candidate patterns may contribute to the support of the candidate pattern. This creates a problem for databases containing many small variations between otherwise similar patterns. This limitation is also true for our study and results, with Association Rule Mining (ARM).

B. Firewall Policy Rules and Anomaly Detection

Current study will be focused on four areas of (1) data mining of packet filtering rules from firewall log file, (2) aggregation of packet filtering rules, (3) anomaly detection of firewall policy rules in study of firewall policy rules and data mining of filtering packets, and (4) to discover decaying or dominant firewall policy rules through data mining.

Several models have been proposed for filtering rules. Ordered binary decision diagram is used as a model to optimize packet classification in [11]. Another model using tuple space is developed in [16], which combines a set of filters in one tuple and stored in a hash table. The model in [17] uses bucket filters indexed by search trees. Multi-dimensional binary trees are also used to model filters [15]. In [16] a geometric mode is used to represent 2-tuple filtering rules. The reason is that these models were designed particularly to optimize packet classification in high-speed networks too complex to use for firewall policy analysis. The analysis in [2] showed that the tree-based model was simple and powerful enough to use for packet filtering. Instead diagrams and pre-processing of firewall rules are used in [10] to resolve rule overlapped and to compact firewall policy rules, but it cannot be used for anomaly detection. There are a number if papers with emphasis on filtering performance [9, 10] and few related work [7, 11] address rule combination in filtering policies. Other approaches [3, 10] propose using a high level language to define rules. This may avoid rule anomalies but not practical.

For our study, we have implemented a tree-based filtering representation to detect anomaly discovery algorithm similar to the work performed in [1]. In [1, 2], developed a tool for analyzing firewall polices called "Policy Advisor" that has widely disseminated in academic and industrial communities. However, this tool does not consider network traffic or device logs. This results in uncovering other critical and non-systematic anomalies such as blocking traffic to existing

legitimate services, permitting or blocking traffic to non-existing services. In this proposed research, we will develop traffic data mining techniques to go beyond static analysis of these polices to discover other unrevealed misconfigurations. This mining will uncover any hidden or embedded anomaly that may not be detectable from the anomaly detection methods using only the current firewall policy rules.

III. MINING FIREWALL LOG FILE

In this section, the overall process of mining log file to generate policy rules and its architecture will be presented as shown in Fig. 1.

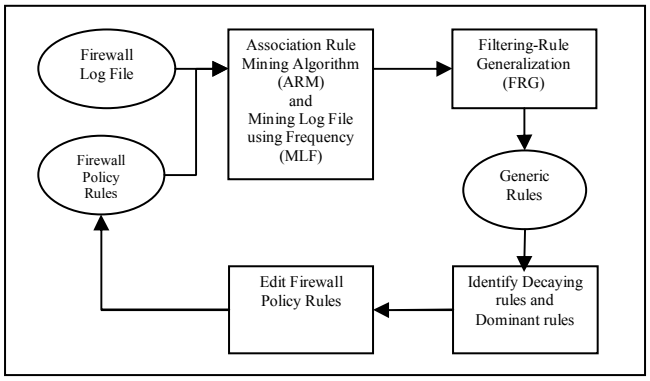

Figure 1. Flow Diagram of Our Approaches

The process consists of the following five iterative components in sequence: (1) to analyze and generate an initial set of firewall policy rules, to collect firewall log raw data, and to extract the attributes for data mining, (2) to perform data mining of Association Rule Mining and Mining firewall Log using Frequency) along generalization to discover a new set of the updates to the initial firewall policy rules, (3) to identify decaying rules and dominant rules, and (4) to edit and generate the updated firewall policy rules, generalized and anomaly-free.

A. Analysis of Firewall Policy Rules

Initially the firewall contains a set of firewall policy rules specified by network security administrator in knowledge of its organization's security policy. Each rule specifies to accept or drop (deny) a packet of certain attributes. For example, a rule could specify a packet for Hyper Text Transport Protocol (HTTP) used to transfer HTML documents of the World Wide on port 80 with a specific Source IP Address as shown in Fig. 2.

```
-A INPUT -s 129.110.96.117 -p tcp -m tcp --dport 80 -j DROP
```

Figure 2. An Example of a Linux Firewall Policy Rule

Each rule could consist of up to seven attributes. These attributes are in the following format of "<direction> <protocol> <source IP> <source port> <destination IP> <destination port> <action>". The first attribute of direction is the packet direction whether it is in-coming or out-going. The second attribute is the protocol of packet such as TCP or UDP. The next attributes are two pairs of IP address and its port for source and destination of packet. Finally the last attribute is the action upon packet being accepted or dropped by firewall. Further an attribute may not be present in a firewall rule to imply all or any. An attribute may be a range such as the range of TCP port from 1024 to 65535 or the range of IP address in a local area network by mask.

B. Association Rule Mining(ARM)

This step is to collect and extract these attributes from the collected log file data, to do Apriori analysis. The extracted attributes consist of protocol (TCP or UDP), direction (incoming or outgoing), source IP, destination IP, source port, destination port, and action (accept or deny). Furthermore, these attributes are defined as nominal to avoid any functional significance for its values. A sample line of Linux firewall log is shown in Fig. 3. Also the protocol is limited to be either TCP or UDP for our study.

```
May 31 17:53:51 utd52075 kernel: ACCEPT_LOGIN=
OUT=eth0 SRC=129.110.96.80 DST=129.110.31.7 LEN=62
TOS=0x00 PREC=0x00 TTL=64 ID=10849 DF PROTO=UDP
SPT=32789 DPT=53 LEN=42
```

Figure 3. An Example of Linux Firewall Log

In Association Rule Mining (ARM), an association rule is of the form X => Y where X and Y are disjoint conjunctions of attribute-value pairs. The *confidence* of the rule is the conditional probability of Y given X, Pr(Y|X), and the *support,* of the rule is the prior probability of X and Y, Pr(X and Y). Here probability is taken to be the observed frequency in the data set. The traditional association rule mining problem can be described as follows. Given a database of transactions, a minimal confidence threshold and a minimal support threshold, the goal is to find all association rules whose confidence and support are above the corresponding thresholds. Hence, ARM generates the largest item-set and the best rules from log dataset. By using small values for minimum support such as (0.001), the algorithm would generate more rules. Also by using large value (e.g., 0.9), it would generate fewer rules. Also the confidence value is set to 1 for 100 percent confidence. Therefore, using minimum support and minimum confidence, ARM algorithm finds the largest itemset followed by the best rules found, considering only rules with action field (i.e., =Y). It is because the rule with no action has no effect on firewall policy rule.

C. Mining firewall Log using Frequency (MLF)

Along with Association Rule Mining, we have developed and successfully applied much simpler and efficient data mining algorithm using simple frequency - Mining firewall Log using Frequency (MLF). MLF algorithm reads each line of firewall log file, extracts the attributes for each log record, counts its occurrence and outputs the count for each unique combination of these attribute-values. The frequency of each rule discovered is kept and summed up for these rules are being aggregated, and used for its probability and statistical processing. Thus each log record of a packet in firewall log file is then processed to be a primitive rule. Here *Primitive rule* is a specific firewall policy rule where all of its attributes are instantiated or specified with its value being observed in firewall log file. Thus the initial process of extracting the attributes of a packet of each log record is to discover and

generate its corresponding primitive firewall rule, with a set of the instantiated attributes of (1) direction such as incoming or outgoing packet, (2) protocol such as TCP or UDP, (3) source IP, (4) source port, (5) destination IP, (6) destination port, and (7) the action to accept or deny for a packet satisfying these attributes. MLF algorithm is shown in Fig. 4.

Mining firewall Log using Frequency (MLF) Algorithm

Input: Firewall Log file
Output: Unique Rules and their Frequency

```
1. Packet# ← 0
2. FOR EACH Line in Firewall Log file
3.         FirewallRule[Packet#] ← Protocol, Direction,
              SRC-IP, SRC-Port, DST-IP, DST-Port, Action
4.         Increment Packet#
5. END FOR
6. FOR EACH i WHERE 0 ≤ i < Packet#
7.         Frequcney ← 0
8.         FOR EACH j WHERE i < j < Packet#
9.               IF FirewallRule[i] = FirewallRule[j]
10.                    Increment Frequency
11.              END IF
12.        END FOR
13.        IF FirewallRule[i] NOT discovered previously
14.              Write FirewallRule[i] and Frequency
15.        ELSE
16.              Continue
17.        END IF
18. END FOR
```

Figure 4. Mining firewall Log using Frequency (MLF) Algorithm

D. Filtering-Rule Generalization (FRG)

Filtering-Rule Generalization (FRG) is an aggregation algorithm to generate a minimum number of firewall policy rules for efficiency and Anomaly detection purpose. FRG generates a decision tree where each level or branching represents one of the attributes of a primitive rule from log record of a packet which is an instance of a rule to be processed. Each of the rules extracted from firewall log file is then examined to be branched and analyzed, and then to be grouped to be aggregated by combining the common fields to get the superset rules that match with a group of unique rules.

For example, let's consider to have a few rules of all matching fields with an exception of source port field. In this case, these rules are then aggregated by combining all the matching fields, that is, by combining all the source ports as a range of these source ports to get one generalized rule. The algorithm is further extended to produce more general rules in an incremental manner by appending new rules from the firewall log file in different time period with time information of a packet. For the simplicity and explanation, let's consider a case where its final action is "DENY." The process for generalization for action of "ACCEPT" is essentially same as for "DENY" action.

Associated with each rule are seven attributes with the following format of "<action> <protocol> <direction> <destination-port> <source-port> <source-IP> <destination-IP>". Each primitive rule starts with the Action parameter and the protocol with binary decision of whether it is UDP or TCP. Next node in the tree describes direction of the packet (either INPUT or OUTPUT) with a destination port. Then a decision is made at each destination port. One of the observations is that the aggregation process over these attributes is binary for action, protocol and direction, and a range for destination-port, source-port, source-IP and destination-IP.

One of the limitations or the boundary of current data mining approach to extract and aggregate IP addresses or its port numbers from the firewall log is that at best its result is ultimately bound or limited by what is observed and kept in the log data, and nothing more. And this observation (or the mining result) is or should have been governed by the firewall policy rules under the scope of current study. Unless it is an ideal or thoroughly-simulated situation, one may not expect to observe all the IP addresses and all the port numbers in the firewall log, thus further to be aggregated for its corresponding range of IP addresses or port numbers or even to be "ANY" or "*" to cover all the specified range, for the corresponding firewall rules. At best the result of the rules resulting from the data mining is expected to be a subset of the firewall rules.

The challenge is then how to bridge the gap between the rules mined and aggregated from the firewall log and the firewall policy rules defined for the firewall, or how to accommodate or collaborate its difference. This challenge and problem may present a classical opportunity of data modeling in network or *network data modeling* for firewall and firewall policy rules, to model a *network data space* of packets expressed by a set of firewall policy rules to represent a schematic view of a conceptual network data space, to identify any anomalies of the firewall policy rules (defined and expressed as the logical view of the network data space) and by what has been observed and data mined from its firewall log data.

Another Similar problem in the network data analysis and classification is the aggregation of IP addresses (e.g., [35] RFC950, variable-length subnet masking of IPv4 or IPv6, which has been also widely adapted and used in various firewall policy rule coding scheme using "*" for 8-bit IP address-range or with "-" to represent a range of IP addresses or port numbers). This unique addressing scheme for a group of contiguous IP address is designed to provide very simple and convenient way of expressing of a group of contiguous IP addresses (RFC950 [35]) even though it was a recommendation to be contiguous from the least significant bits and not a requirement.

Its advantage is to represent a subnet using a similar IP address format, to represent a subnet consisting of a group of contiguous IP addresses, to isolate subnet or inside-network traffic, thus to reduce incoming or outgoing network traffic in and out of a subnet, to provide secure or limited access and control to a subnet, to provide subnet-localized network protocol, to align a subnet with business-based or policy-based hierarchy or segmentation of network configuration, and to administer a group of hosts as a logical unit. The problem is similar to find a least upper bound covering a set of contiguous IP addresses, in 32-bit integer, of Version Space [11].

Two approaches of (1) conservative aggregation and (2) aggressive aggregation have been proposed to handle this problem and challenge for current study. First, the conservative aggregation is to do the aggregation as much as it can be done through mining the log data over a sufficient period of time, by observing and accumulating its intermediate mining results over the time, continuously and gracefully. For example, data collection and its mining process is done daily over a period of a month or a year, accumulating its daily mining results. In contrast, the aggressive approach is to compromise the aggregation process by a brute-force simplification given the log data at hand.

A few of the compromise for this approach have been considered and implemented to simplify the aggregation process with the limited amount log data for our prototype system. One is to provide a table of IP addresses inside of the firewall (to simplify the source IP for outgoing packets and the destination IP for incoming packets) to the rule mining algorithm, to aggregate two source IP addresses to be "*" for its subnet or its octal for those outgoing packets with the same destination IP addresses, given the fact that these are the only two IP addresses inside of the firewall or its subnet. For example, the IP addresses of 129.110.10.7 and 129.110.10.1 can be aggregated to 129.110.10.* for its subnet. The IP addresses of 10.110.96.255 and 255.255.255.255 can be aggregated to *.*.*.* (aggressive one) for the broadcasting, and so on.

The similar approach can be used for the aggregation of the port numbers. One approach is to set a tier of port numbers such as 1-1023, 1024-1999, 2000-2999, and so on, to aggregate any port number falling into one of the tiers to be its range of the tier (for example, 2000:2999) with its subrange incrementing in the unit of 1000, then incrementing in the unit of 10000, and so on. The final port ranges for our experiment has been (1) two tiers of 1-1023 and 1024-65535, and (2) one range of port 1-65535, to be further simplified. The last one is a drastic but with much simpler for the prototype implementation to aggregate two distinct port numbers to be "ANY". Further a predefined table of the important or critical port numbers (for example, as shown in Table I) could be used to simplify the process of aggregation, instead of checking all the ports available (from port 1 to port 65535 where the well-known or reserved ports are port 1 through port 1023).

TABLE I. SOME OF IMPORTANT PORT NUMBERS

Port Name	Port # Protocol	Alias
ftp	21/tcp	
telnet	23/tcp	
Smtp	25/tcp	Mail
Nickname	43/tcp	Whois
Domain	53/tcp	Nameserver
Finger	79/tcp	
http	80/tcp	www www-http
https	443/tcp	

Filtering-Rule Generalization (FRG) algorithm in pseudo code is shown in Fig. 5.

```
Generalization Algorithm Pseudo Code
Input: Rules
Output: The Tree
1.FOR EACH attribute ∈ {Action, Protocol, Direction, DST Port, SRC Port, SRC IP, and DST IP}
2.   IF attribute doesn't exist in the tree AND attribute is NOT DST Port
3.       Create new branch with attribute value
4.   ELSE
5.       IF DST Port exists in the table
6.           IF DST Port doesn't exist in Tree
7.               Create new branch with DST Port Value
8.           ELSE
9.           Follow existing branch with DST Port Value
10.      ELSE IF DST Port does not exist in Table
11.          IF DST Port doesn't exist in Tree
12.              Create new branch with DST Port Value
13.          ELSE
14.              Determine Range of DST Port
15.              Update existing branch with DST Port Range Value
16.          END IF
17.      END IF
18.      IF SRC Port exists in the table
19.          IF SRC Port doesn't exist in Tree
20.              Create new branch with SRC Port Value
21.          ELSE
22.              Follow existing branch with SRC Port Value
23.      ELSE IF SRC Port does not exist in Table
24.          IF SRC Port doesn't exist in Tree
25.              Create new branch with SRC Port Value
26.          ELSE
27.              Determine Range of SRC Port
28.              Update existing branch with SRC Port Range Value
29.          END IF
30.      END IF
31.      IF SRC IP doesn't exist in Tree
32.          Create new branch with SRC IP Value
33.      ELSE
34.          Determine Superset of SRC IP
35.          Update existing branch with SRC IP Superset Value
36.      END IF
37.      IF DST IP doesn't exist in Tree
38.          Create new branch with DST IP Value
39.      ELSE
40.          Determine Superset of DST IP
41.          Update existing branch with DST IP Superset Value
42.      END IF
43.  END IF
44.END LOOP
45.PRINT the generated Rule (Each path of the tree will produce a new generated Rule).
```

Figure 5. Filtering-Rule Generalization (FRG) Algorithm

It takes the rules as input, and generalizes them. This algorithm loops over each attribute field of these two rules (attributes are seen in line #1). For all the existing values of "Action," "Protocol," and "Direction" attributes the algorithm creates a new leaf if the leaf is not already created. Otherwise, it would skip to the next attribute (line #2 to #4). When reaching the "DST Port" attribute of the rules, the algorithm checks if there is an associated leaf for the value of this attribute. Conditions checked in lines #5 to 17 determines whether the algorithm need to create, replace, or skip the "DST Port" leaf. The "SRC Port" leaf in lines #18 to #30 is

determined based on the same procedure as "DST Port" leaf. Next, in lines #31 to #36, the algorithm decide to either create a leaf for the value of "SRC IP" attribute or not based on whether an IP of equal value or superset value already exist in the tree or not. Same logic would apply, in lines #37 to #42, for the "DST IP" attribute of the rules. After all the attributes for which a generalized leaf has been created, every path on the generated tree would print a generalized rule that supersets all the existing rules in the input. The output of this algorithm would simply be all the paths on the tree. The resulting rules are shown in Table II.

TABLE II. UNIQUE RULES

R #	Proto	Direction	SRC IP	SRC Port	DST IP	DST Port	Action
1	TCP	INPUT	129.110.96.117	1160	129110.96.80	22	DENY
2	TCP	INPUT	129.110.96.117	1170	129110.96.80	22	DENY
3	TCP	INPUT	129.110.96.117	1160	129110.96.80	80	DENY
4	TCP	INPUT	129.110.96.117	1162	129110.96.80	80	DENY
5	UDP	INPUT	10.110.49.115	67	10.110.96.255	1211	DENY
6	UDP	INPUT	129.110.96.187	48668	10.110.96.255	67	DENY
7	UDP	INPUT	129.110.10.7	53	129.110.96.80	32790	ACCEPT
8	UDP	INPUT	129.110.10.1	53	129.110.96.80	32799	ACCEPT
9	UDP	OUTPUT	129.110.96.80	32799	129.110.96.92	53	ACCEPT
10	UDP	OUTPUT	129.110.96.80	32789	129.110.96.187	53	ACCEPT
…	…	…	…	…	…	…	…

An example of the decision tree in action for Filtering-Rule Generalization algorithm from Table II is shown in Fig. 6.

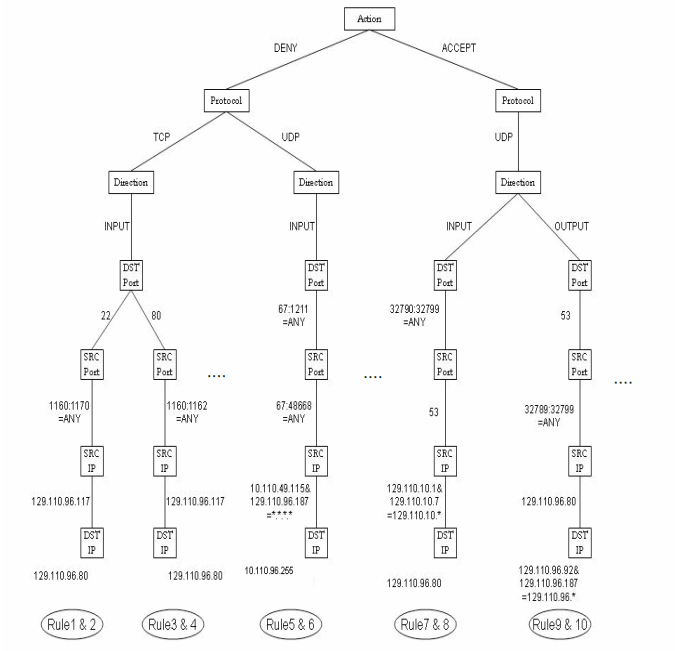

Figure 6. Generalization Method Tree Example

Each rule is defined as the path that it takes from the root (Action node) to the leaf (Destination IP) where the rule resides in. For example, the Rule 3&4 is shown with the following path: DENY→ TCP→ INPUT→ 80→ 1160:1162→ 129.110.96.117→ 129.128.92.80. The generalization method for above example produces the results shown in Fig. 7.

```
1 TCP,INPUT,129.110.96.117,ANY,129.110.96.80,22,DENY
2 TCP,INPUT,129.110.96.117,ANY,129.110.96.80,80,DENY
3 UDP,INPUT,*.*.*.*,ANY,10.110.96.255,ANY,DENY
4 UDP,OUTPUT,129.110.96.80,ANY,129.110.10.*,ANY,DENY
5 TCP,INPUT,*.*.*.*,ANY,129.110.96.80,80,ACCEPT
6 UDP,INPUT,129.110.10.*,53,129.110.96.80,ANY,ACCEPT
7 UDP,OUTPUT,129.110.96.80,ANY,129.110.96.*,53,ACCEPT
```

Figure 7. Generalized Filtering Rules

Here Line 1 shows the rule that denies all TCP packets from source IP d=address of 129.11.96.117 with any port to destination IP address of 129.110.96.80 at port 22. The next task will be the ordering these rules as discussed in the next section.

E. Rule Ordering

The ordering of the rules can affect significantly the outcome of the firewall filtering as well as for its performance. However, current stage of the process is not to generate a set of the order of the rule can be based on its generalization. The more specific a rule is, the sooner it should be used among its rules, to differentiate itself from more general rules. This is based on the assumption that specific rule should be applied first; otherwise, it may not be applied at all due to a conflict between the rules. A revised version of this ordering approach is to score each rule based on how much it has been generalized. Then we sort rules based on ascending order of these scores.

One may score a generalized rule based on its degree of the range of IP address or of port number, to add 1 point for each "*" of source or destination IP address, and 1 point for "ANY" for port. For example, Rule 3 in Fig. 7 gets 3 points whereas Rule 6 gets 2 points. Thus Rule 3 will be placed ahead of Rule 6 as Rule 3 and Rule 6 have the same attributes except for source IP address. As a result of this ordering, the rule 3 and 4 will be reversed for the anomaly detection.

F. Anomaly Detection

In a centralized firewall policy, it is important for packet filtering process whether its filtering rules are disjoint or not, or even worse in conflict. Thus it is very common to discover that some of these filtering rules are interrelated and thus its ordering may create very different results or anomalies, thus resulting in an incorrect firewall policy.

Firewall policy anomaly will occur (1) when two or more rules match the same packet or (2) when a rule never matches any packet that goes through the firewall. As the number of rules increases, it tends to include more anomalies. Thus it is very important to detect and remove these anomalies in a timely manner, at least to keep firewall policy rules free of anomaly. The firewall policy anomalies can be classified into 4 categories [1, 2]:

1) Shadowing anomaly: A rule is shadowed when a previous rule matches all the packets that match this rule, such that the shadowed rule will never be activated.

2) Correlation anomaly: Two rules are correlated if they have different filtering actions and 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: A 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: A redundant rule performs the same action on the same packets as another rule, such that if the redundant rule is removed, the security policy will not be affected.

In addition, our log based mining approach can discover the following non-systematic misconfiguration anomalies.

5) Blocking existing service anomaly: A common misconfiguration case is blocking a legitimate traffic from a trusted network to an "existing" service. This for example might happen as a result of misconfigurating the port number or deleting by mistake the exception rule that allows the traffic from the trusted network. This type of anomaly can be simply detected when mining the log file as the analyst would know that there is a traffic from a trusted network is being denied to access an existing (legitimate) service/port.

6) *Allowing traffic to non-existing services anomaly*: Another case of the misconfiguration is to permit a traffic destined to non-exiting service. For example, the administer configures rules to pass traffic at port 79; however, there is no "finger' service available with port 79. In that case this passed traffic with port 79 will be useless. In that case, one option is we need to block traffic with port 79. This anomaly can be detected after mining log files of both the firewall and the remote hosts.

For an example of Shadowing anomaly, the rules presented in Fig. 8 illustrates that Rule 4 is shadowed by Rule 3. As a result, Rule 4 will never be activated.

Order	Protocol	Direction	Source IP	Source Port	Destination IP	Destination Port	Action
1:	TCP	INPUT	129.110.96.117	ANY	*.*.*.*	80	DENY
2:	TCP	INPUT	129.110.96.*	ANY	*.*.*.*	80	ACCEPT
3:	TCP	INPUT	*.*.*.*	ANY	129.110.96.80	80	ACCEPT
4:	TCP	INPUT	129.110.96.*	ANY	129.110.96.80	80	DENY
5:	TCP	OUTPUT	129.110.96.80	22	*.*.*.*	ANY	DENY
6:	TCP	INPUT	129.110.96.117	ANY	129.110.96.80	22	DENY
7:	UDP	INPUT	129.110.96.117	ANY	129.110.96.*	22	DENY
8:	UDP	INPUT	129.110.96.117	ANY	129.110.96.80	22	DENY
9:	UDP	INPUT	129.110.96.117	ANY	129.110.96.117	22	ACCEPT
10:	UDP	INPUT	129.110.96.117	ANY	129.110.96.117	22	DENY
11:	UDP	OUTPUT	*.*.*.*	ANY	*.*.*.*	ANY	DENY

Figure 8. Firewall Policy Rules for Anomaly Illustration

Further any matching packet to Rule 4 will be accepted by the shadowing rule, in this case Rule 3, instead of denying as Rule 4 states.

For an example of Correlation anomaly, Rule 1 and Rule 3 are correlated because the source IP in rule 1 matches the source IP in rule 3 whereas the opposite is not true. The destination IP in rule 3 matches the destination IP in rule 1 whereas the opposite is not true. The two rules with this ordering imply that all traffic that is coming from 129.110.96.117 and going to 129.110.96.80 is denied. However, if their order is reversed, the same traffic will be accepted. Correlation is considered an anomaly warning because the correlated rules imply an action that is not explicitly stated by the filtering rules.

For an example of Generalization anomaly, Rule 2 is a generalization of Rule 1 in Figure 8. These two rules imply that all traffic that is coming from the address 129.110.96.* will be accepted, except the traffic coming from 129.110.96.117. Generalization is often used to exclude a specific part of the traffic from a general filtering action. It is considered only an anomaly warning because the specific rule makes an exception of the general rule. This might cause an accepted traffic to be blocked or a denied traffic to be permitted, and thus it is important to highlight its action to the administrator for confirmation.

Finally for an example of Redundancy anomaly, Rule 7 is redundant to Rule 6 in Fig. 9. Redundancy is considered an error in the firewall policy because a redundant rule adds to the size of the filtering rule list, and, therefore, increases the search time and space requirements of the packet filtering process [4]. Thus Anomaly Discovery algorithm is then to identify all the relationship among the rules, to reveal any rule conflict which is one of (1) shadowing anomaly, (2) correlation anomaly, (3) generalization anomaly or (4) redundant anomaly. The state diagram in Fig. 9 illustrates the firewall anomaly discovery states for any two rules RX and RY where RY follows RX.

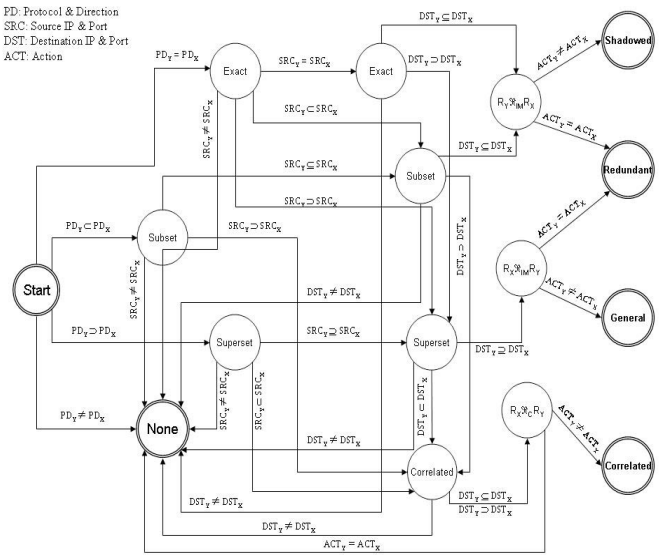

Figure 9. Firewall Anomaly Detection State Diagram

IV. MINING RESULTS

In this section, the result of the MLF process is presented with an example using a LINUX firewall log data to obtain the correct firewall policy rules generalized and anomaly-free. A firewall log file consists of 33,172 packets filtered was obtained from a LINUX operating system firewall, initially with 10 firewall policy rules. Current study uses only the

packet filtering log file. The application layer log files are not implemented in current prototype.

A. Dataset Analysis & Process

The steps are used to generate, filter, and generalize policies without any anomaly by using Anomaly Discovery Algorithm.

First Step (1) is to process Firewall Policy Rules to generate firewall traffic log file using LINUX firewall. LINUX firewall policy rules and firewall log data of 33,172 log records. Further to reduce the number of states for our research and prototype, we will consider only the seven major fields in the firewall policy rules as well as in IP packet header: protocol (TCP or UDP), packet direction (incoming or outgoing), source IP address and source port, destination IP address and destination port, and action. Each rule could consist of up to seven attributes. These attributes are in the following format of "<direction> <protocol> <source-IP> <source-port> <dest-IP> <dest-port> <action>". The absence of any of the above attributes in the policy rule indicates that such a rule is not impacted by that attribute. For example in line 24, if any packet arrives as an input to firewalls from a source IP address 129.110.96.117 on any port to any destination IP address on port 80 using TCP protocol, it should be denied (dropped).

A listing of Linux firewall policy rules which is used for our study is shown in Fig. 10.

```
COMMIT
# Completed on Mon Jun  6 18:46:14 2005
# Generated by iptables-save v1.2.11 on Mon Jun  6 18:46:14 2005
*mangle
:PREROUTING ACCEPT [265144:123826498]
:INPUT ACCEPT [259110:123137467]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [185663:69820616]
:POSTROUTING ACCEPT [182657:69005504]
COMMIT
# Completed on Mon Jun  6 18:46:14 2005
# Generated by iptables-save v1.2.11 on Mon Jun  6 18:46:14 2005
*filter
:INPUT ACCEPT [169897:115595620]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [180376:68855166]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -s 129.110.96.117 -p tcp -m tcp --dport 80 -j LOG --log-prefix "DROP_LOG"
-A INPUT -s 129.110.96.117 -p tcp -m tcp --dport 80 -j DROP
-A INPUT -p tcp -m tcp --dport 80 -j LOG --log-prefix "ACCEPT_LOG" --log-tcp-sequence
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j LOG --log-prefix "DROP_LOG" --log-tcp-sequence
-A INPUT -p tcp -m tcp --dport 443 -j DROP
-A INPUT -s 129.110.96.117 -p tcp -m tcp --dport 22 -j LOG --log-prefix "DROP_LOG"
-A INPUT -s 129.110.96.117 -p tcp -m tcp --dport 22 -j DROP
-A INPUT -p tcp -m tcp --dport 22 -j LOG --log-prefix "ACCEPT_LOG" --log-tcp-sequence
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -i eth0 -p udp -m udp --sport 53 --dport 1024:65535 -j LOG --log-prefix "ACCEPT_LOG"
-A INPUT -i eth0 -p udp -m udp --sport 53 --dport 1024:65535 -j ACCEPT
-A INPUT -i eth0 -p udp -j LOG --log-prefix "DROP_LOG" --log-tcp-sequence
-A INPUT -i eth0 -p udp -j DROP
-A OUTPUT -s 129.110.96.80 -p tcp -m tcp --dport 22 -j LOG --log-prefix "DROP_LOG"
-A OUTPUT -s 129.110.96.80 -p tcp -m tcp --dport 22 -j DROP
-A OUTPUT -o eth0 -p udp -m udp --sport 1024:65535 --dport 53 -j LOG --log-prefix "ACCEPT_LOG"
-A OUTPUT -o eth0 -p udp -m udp --sport 1024:65535 --dport 53 -j ACCEPT
-A OUTPUT -o eth0 -p udp -j LOG --log-prefix "DROP_LOG" --log-tcp-sequence
-A OUTPUT -o eth0 -p udp -j DROP
COMMIT
# Completed on Mon Jun  6 18:46:14 2005
```

Figure 10. LINUX Firewall Policy Rules Example

Second step (2) is to process firewall log dataset to extract packet features for data mining using ARM and MLF. Each line of firewall log dataset indicates the following information for each packet: timeframe, action (DROP_LOGIN), direction (OUT=), source IP address (SRC=), destination IP (DST=), size of the packet (LEN=), time to leave (TTL=), packet ID (ID=), protocol (PROTO=), source port (SPT=), and destination port (DPT=). A partial result of MLF process for IP address of 129.110.* is shown in Fig. 11.

# of Count	Protocol	Direction	SourceIP	SourcePort	DestinationIP	DestinationPort	Action
6969	UDP	INPUT	129.110.96.242	1059	255.255.255.255	1211	DENY
3	UDP	INPUT	129.110.96.92	61469	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61473	129.110.96.255	177	DENY
180	UDP	OUTPUT	129.110.96.80	123	129.110.10.100	123	DENY
348	UDP	INPUT	129.110.96.187	138	129.110.96.255	138	DENY
3	UDP	INPUT	129.110.96.92	61479	129.110.96.255	177	DENY
1158	UDP	INPUT	129.110.96.187	137	129.110.96.255	137	DENY
3	UDP	INPUT	129.110.96.92	61485	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61493	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61501	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61511	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61517	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61527	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61533	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61541	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61551	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61557	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61565	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61569	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61575	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61581	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61585	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61587	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61589	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61595	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61597	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61601	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61605	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61609	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61611	129.110.96.255	177	DENY
3	UDP	INPUT	129.110.96.92	61613	129.110.96.255	177	DENY
141	UDP	OUTPUT	129.110.96.80	68	129.110.10.1	67	DENY
3	UDP	INPUT	129.110.96.92	61617	129.110.96.255	177	DENY

Figure 11. Filtering Rules Example from MLF

It's also possible to use any other field in protocol headers in this step to extend its preprocessing and analysis for the next step of Generalization.

Third step (3) is to generalize rules using Filtering-Rule Generalization. After this step, each primitive rule with frequency will be further generalalize or aggregated. For example, IP address in a rule can be a particular host (a unique IP address such as 129.110.96.100) or resulted in an aggregated network address (e.g., 129.110.96.*) by combining a group of similar rules with the same IP address. A port can be either a single specific port number or combined to be "ANY" for being any port number.

It is always possible to split the filtering rule of a multi-valued field into several rules where each rule will have a single-valued field. The attributes consist of the seven fields to analyze and discover knowledge from traffic log data where its instances are the packet log information from firewall log file. The rules are grouped and generalized for the superset of all rules identified. For example, grouping all the rules of IP address of 129.110.96.80 (whether it is source IP or destination IP) with the destination port (=53), and action (=accept) is resulted in Rule 7 shown in Fig. 7.

Fourth Step (4) is first to combine the discovered policy rules from the log file with the initial firewall policy rules. The resulting combined rules (of 17 rules) are shown in Fig. 12.

```
TCP,INPUT,129.110.96.117,ANY,*.*.*.*,80,DENY
TCP,INPUT,*.*.*.*,ANY,*.*.*.*,80,ACCEPT
TCP,INPUT,*.*.*.*,ANY,*.*.*.*,443,DENY
TCP,INPUT,129.110.96.117,ANY,*.*.*.*,22,DENY
TCP,INPUT,*.*.*.*,ANY,*.*.*.*,22,ACCEPT
TCP,OUTPUT,129.110.96.80,ANY,*.*.*.*,22,DENY
UDP,OUTPUT,*.*.*.*,ANY,*.*.*.*,53,ACCEPT
UDP,INPUT,*.*.*.*,53,*.*.*.*,ANY,ACCEPT
UDP,OUTPUT,*.*.*.*,ANY,*.*.*.*,ANY,DENY
UDP,INPUT,*.*.*.*,ANY,*.*.*.*,ANY,DENY
TCP,INPUT,129.110.96.117,ANY,129.110.96.80,22,DENY
TCP,INPUT,129.110.96.117,ANY,129.110.96.80,80,DENY
UDP,INPUT,*.*.*.*,ANY,129.110.96.80,ANY,DENY
UDP,OUTPUT,129.110.96.80,ANY,129.110.10.*,ANY,DENY
TCP,INPUT,*.*.*.*,ANY,129.110.96.80,80,ACCEPT
UDP,INPUT,129.110.*.*,53,129.110.96.80,ANY,ACCEPT
UDP,OUTPUT,129.110.96.80,ANY,129.110.*.*,53,ACCEPT
```

Figure 12. Combined Firewall Policy Rules

Fifth Step (5) is to detect policy anomalies from these combined firewall policy rules as shown in Fig. 13.

```
Rule 1, Rule 2:   ==> GENRERALIZATION
Rule 1, Rule 15:   ==> CORRELATED
Rule 2, Rule 1:   ==> SHADOWED
Rule 2, Rule 12:   ==> SHADOWED
Rule 4, Rule 5:   ==> GENRERALIZATION
Rule 5, Rule 4:   ==> SHADOWED
Rule 5, Rule 11:   ==> SHADOWED
Rule 11, Rule 5:   ==> GENRERALIZATION
Rule 12, Rule 2:   ==> GENRERALIZATION
Rule 12, Rule 15:   ==> GENRERALIZATION
Rule 15, Rule 1:   ==> CORRELATED
Rule 15, Rule 12:   ==> SHADOWED
```

Figure 13. Anomaly Detection Alorithm Output

Using Anomaly Discovery Algorithm as discussed in Section III.F, it detected that 5 rules are generalization, 2 rules are correlated, and 3 rules are shadowed rules. These results can be utilized two different ways. The first is to use the more general rules to update the firewall policy rules. The second option is to have more specific rules in the final firewall policy rules, to detect unwanted traffics. After all, the existing firewall policy rules (of 10 initial rules) are combined with the generalized rules (of resulting 7 rules resulting from the firewall log file of 33,172 packet records), then to detect the anomalies (among 8 rules) for the final firewall policy rules (of 10 rules). The final result is new policy rules ordered, generalized and anomaly-free to guarantee the correctness and efficiency of policy rule configuration and filtering process.

B. Decaying Rules and Dominant Rules

Further one of the intriguing questions that this paper investigates is that how a rule in the initial firewall policy rules is useful or effective for current network traffic pattern. This information would provide valuable insight to understand whether (1) a rule is useless or decaying for this period of time, (2) a rule is a dominating rule which covers a significant portion of the network traffic.

Another promising result is the probability distribution from the firewall log file of 33,172 records of packets is shown in Table III and Fig. 14.

TABLE III. DISTRIBUTION OF PACKETS OVER INITIAL FILTERING RULES

#	SRC-IP	SRC-Port	DST-IP	DST-Port	Prob.
1	129.110.96.117	ANY	*.*.*.*	80	0.000100553
2	*.*.*.*	ANY	*.*.*.*	80	0.00144126
3	*.*.*.*	ANY	*.*.*.*	443	0.000234624
4	129.110.96.117	ANY	*.*.*.*	22	0.000100553
5	*.*.*.*	ANY	*.*.*.*	22	0.62600972
6	129.110.96.80	ANY	*.*.*.*	22	0
7	*.*.*.*	1024:65535	*.*.*.*	53	0.013541143
8	*.*.*.*	53	*.*.*.*	1024:65535	0.013541143
9	*.*.*.*	ANY	*.*.*.*	ANY	0.010759175
10	*.*.*.*	ANY	*.*.*.*	ANY	0.334271828
Total Probability					1

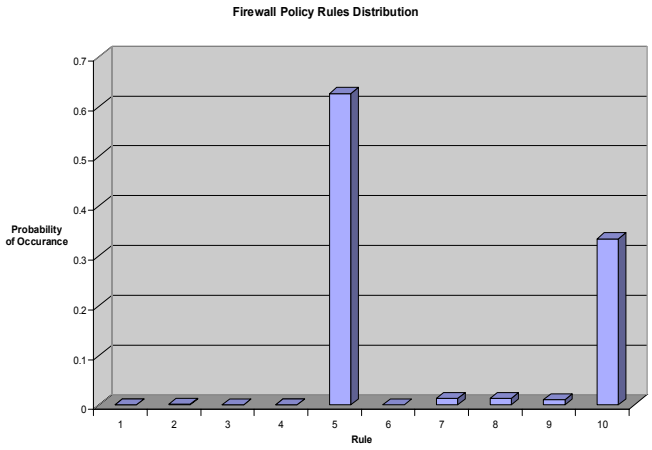

Figure 14. Distribution of Packets Over Initial Filtering Rules

This bar chart illustrates the probability distribution of each rule from the initial 10 firewall policy rules where each point of the x-axis is a rule number and the y-axis represents the scale of the probability. Ranking each rule is based on the frequency of occurrence of each rule in a firewall log file. The probability of occurrence of each packet is calculated as follows: P = f / N where the probability (P) equals the frequency of occurrence of the packets (f) divided by total number of packets (N) recorded in the firewall log file. In our experiment, we found the frequencies of the initial 10 firewall policy rules from the firewall log file generated for one week.

This clearly indicates that Rule 5 and Rule 10 are the dominant rules, covering over 96% of the network traffics and that Rule 6 could be an obsolete or decaying rule. For each general policy rule we also found the frequency of each specific rule (specific source and destination), useful for further analysis. For this sample, the specific rules match with Rule 10

which is one of the most heavily used rules. This information can help to update the rule ordering in firewall in order to optimize the filtering matching at real-time. Further the overall distribution of network traffic pattern by the firewall policy rules reveals yet another interesting insight for the further study. This following line-graph of the probability distribution of the packets over the final 16 rules is clearly a heavily skewed distribution over 16 rules as shown in Fig. 15.

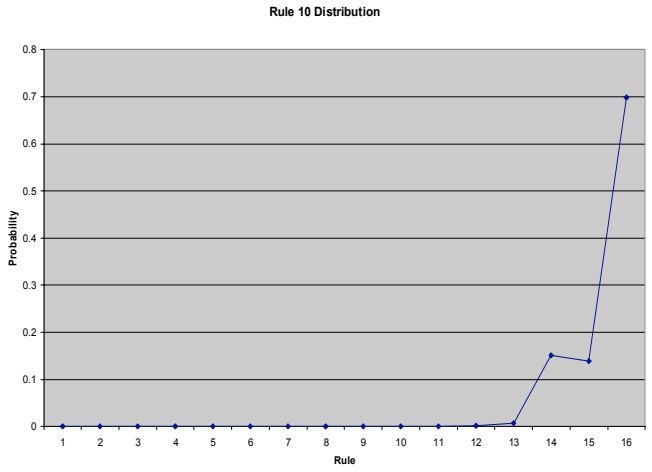

Figure 15. Distribution of Packets Over Resulting 16 Rules

Here the point of x-axis corresponds to each rule and the y-axis is the scale of probability. The probability distribution of each unique rule illustrates a population of network traffic patterns which are not equally distributed but heavily concentrated into a few rules. This is very positive and strong indicator for the potential gain by focusing on a few handfuls unique rules. By reordering or prioritizing a small portion of firewall policy rules, one may expect a tremendous performance gain as a result. In this example, Rule 16 will be the first candidate to be considered as the dominant rule.

Another promising indicator is of decaying rules for its usage or usefulness for the packet traffic flow over a period of time for Rule 6, shown in Fig. 16.

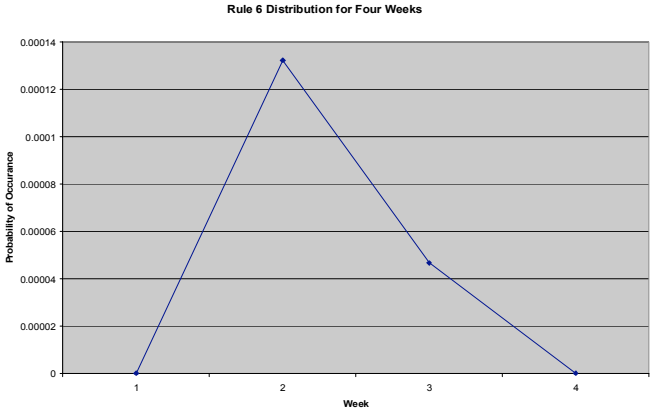

Figure 16. Frequency of Packets over Rule 6 Over 4 Weeks

This figure shows the probability distribution of Rule 6 mined from a log file generated in each week during a month where the x-axis is the week number and the y-axis is the scale of the probability. The distribution confirms the possibility of a rule which follows a life cycle and is eventually decaying after a certain period of time. This is one of the promising indicators to confirm one of the objectives of this research to discover decaying rules and for the timely update to the firewall policy rules.

V. Conclusion and Future Work

Firewall policy rules are one of most important element of network security system. It plays the vital role in management of any organization's network and its security infrastructure. Thus the management of policy rule is a significant task for the network security. There have been a number of tools and techniques used to perform anomaly detection and rule editing by utilizing given set of existing policy rules. However, one of the assumption and thus its limitation is that firewall and its rules are set to be static and thus without an ability to reflect the network behavior observed by firewall.

In our paper, we have presented a new process of managing firewall policy rules, consisting of anomaly detection, generalization and policy update using Association Rule Mining and frequency-based techniques. The advantages are in summary, (1) to provide a capacity to reflect current trend of network traffic and thus to update firewall policy rules in real time from firewall log data files, (2) to provide a tool to analyze its traffic patterns for the further analysis and anomaly detection including those hidden ones, and for the decision making, (3) to apply various data mining techniques to handle both discrete and continuous attributes with operational efficiency and flexibility, and (4) to demonstrate the merit of data mining based algorithms not only feasible but also more accurate and effective (as firewall rules and log dataset gets larger in size and variation in projection). The anomaly detection based on the mining exposes many hidden ones, not only those four types of the anomalies detectable by analyzing the firewall policy rules [1] but also those anomalies not detectable by analyzing the firewall policy rules. As a result, this analysis concluded with two new types of the anomalies. In conclusion, data mining is shown to be not only one of the viable options but also a practical, effective and critical approach in firewall policy rules analysis and optimization in real time.

For future research, current simple model of single firewall should be to be extended to accommodate (1) complex distributed networks with multiple firewalls and perimeter layers [2, 22], (2) various and distinct firewall technologies available in various hardware and software platforms, (3) an effective mechanism to handle a feedback from application layers such as mail servers detecting incoming massive spam mails, (4) efficient and real-time algorithms to handle massive volume of log files and data mining [32, 33], (5) further investigation and analysis on time-dependent statistical behavior of network traffic and policy rules, and (6) for the faulty and leaky network inside of firewall perimeter [34]. The data mining component can be further extended for further analysis of discovering unusual network behaviors such as

spam-mail or misuse of bandwidth, and to discover network traffic patterns over time intervals, to be used in the firewall policy rules and its management.

[1] E. Al-Shaer and H. Hamed. "Firewall Policy Advisor for Anomaly Detection and Rule Editing." *IEEE/IFIP Integrated Management Conference (IM'2003)*, March 2003.

[2] Ehab Al-Shaer and Hazem Hamed, "Discovery of Policy Anomalies in Distributed Firewalls" in Proc. of IEEE INFOCOM'04, vol. 23, no. 1, March 2004 pp. 2605-2616.

[3] Y. Bartal., A. Mayer, K. Nissim and A. Wool. "Firmato: A Novel Firewall Management Toolkit." *Proceedings of 1999 IEEE Symposium on Security and Privacy*, May 1999.

[4] D. Chapman and E. Zwicky. *Building Internet Firewalls, Second Edition*, Orielly & Associates Inc., 2000.

[5] W. Cheswick and S. Belovin. *Firewalls and Internet Security*, Addison-Wesley, 1995.

[6] S. Cobb. "ICSA Firewall Policy Guide v2.0." NCSA Security White Paper Series, 1997.

[7] D. Eppstein and S. Muthukrishnan. "Internet Packet Filter Management and Rectangle Geometry." *Proceedings of 12thψAnnual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2001.

[8] P. Eronen and J. Zitting. "An Expert System for Analyzing Firewall Rules." *Proceedings of 6thψNordic Workshop on Secure IT-Systems (NordSec 2001)*, November 2001.

[9] Z. Fu, F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine and C. Xu. "IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution." *Proceedings of Policy'2001 Workshop*, January 2001.

[10] J. Guttman. "Filtering Posture: Local Enforcement for Global Policies." *Proceedings of 1997 IEEE Symposium on security and Privacy*, May 1997.

[11] Mitchell, T.M., *Machine Learning*. 1997, Sydney: McGraw-Hill.

[12] Salzberg, S.L., Book Review: C4.5: Programs for Machine Learning by J.Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. Machine

[13] Agrawal, R., T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. in Proceedings of the 1993 Webb, G.I., Association Rules, in Handbook of Data Mining, N. Ye, Editor, Lawrence Erlbaum: To appear.

[14] Agrawal, R., et al., Fast Discovery of Association Rules, in Advances in knowledge Discovery and Data Mining, U.M. Fayyad, et al., Editors. 1996, AAAI Press: Menlo Park, CA. p. 307-328.

[15] Srikant, R., Q. Vu, and R. Agrawal. Mining Association Rules with Item Constraints. in Proceedings of the 3rd International Conference on Knowledge Discovery in Databases and Data Mining. 1997. Newport Beach, California.

[16] Piatetsky-Shapiro, G., Discovery, analysis, and presentation of strong rules. Knowledge Discovery in Databases, 1991: p. 229-248.

[17] Webb, G.I. Discovering Associations with Numeric Variables. In Proceedings of the International Conference on Knowledge Discovery and Data Mining. 2001: ACM Press.

[18] Agrawal, R. and R. Srikant. Fast Algorithms for Mining Association Rules. in Proceedings for the 20th Int. Conf. Very Large Data Bases. 1994.

[19] Borgelt, C., Apriori (Computer Software). http://fuzzy.cs.unimagdeburg de/~borgelt/ Accessed via the Internet 17/06/2002.

[20] B. Hari, S. Suri and G. Parulkar. ''Detecting and Resolving Packet Filter Conflicts.'' *Proceedings of IEEE INFOCOM'00*, March 2000.

[21] S. Hazelhusrt. ''Algorithms for Analyzing Firewall and Router Access Lists.'' *Technical Report TR-WitsCS-1999*, Department of Computer Science, University of the Witwatersrand, South Africa, July 1999.

[22] S. Ioannidis, A. Keromytis, S Bellovin and J. Smith. ''Implementing a Distributed Firewall.'' *Proceedings of 7thψ ACM Conference on Computer and Comminications Security (CCS'00)*, November 2000.

[23] E. Lupu and M. Sloman. ''Model-Based Tool Assistance for Packet-Filter Design.'' *Proceedings of Workshop on Policies for Distributed Systems and Networks (POLICY'2001)*, January 2001.

[24] I. Lck. C. Schfer and H. Krumm. ''Conflict Analysis for Management Policies.'' *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM'1997)*, May 1997.

[25] A. Mayer, A. Wool and E. Ziskind. ''Fang: A Firewall Analysis Engine.'' *Proceedings of 2000 IEEE Symposium on Security and Privacy*, May 2000.

[26] L. Qiu, G. Varghese, and S. Suri. ''Fast Firewall Implementations for Software and Hardware-based Routers.'' *Proceedings of 9thψ International Conference on Network Protocols (ICNP'2001)*, November 2001.

[27] V. Srinivasan, S. Suri and G. Varghese. "Packet Classification Using Tuple Space Search." *Computer ACM SIGCOMM Communication Review*, October 1999.

[28] J. Wack, K. Cutler and J. Pole. "Guidelines on Firewalls and Firewall Policy." *NIST Recommendations, SP 800-41*, January 2002.

[29] T. Woo. "A Modular Approach to Packet Classification: Algorithms and Results." *Proceedings of IEEE INFOCOM'00*, March 2000.

[30] A. Wool. "Architecting the Lumeta Firewall Analyzer." *Proceedings of 10thψUSENIX Security Symposium*, August 2001.

[31] "Cisco Secure Policy Manager 2.3 Data Sheet." http://www.cisco.com/ public/cc/pd/sqsw/sqppmn/prodlit/spmgr ds.pdf

[32] W. Lee, "A Data Mining Framework for Constructing Features and Models for Intrusion Detection", Ph.D. Dissertation, Columbia Univeristy, 1999.

[33] M. Mahoney, ''A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic'', Ph.D Dissertation, Florida Institute of Technology, 2003.

[34] R. Smith and S. Bhattacharya, ''Fault and Leak Tolerance in Firewall Engineering,'' *Proc. Third IEEE International High-Assurance Systems Engineering Symposium (HASE98)*, Nov 1998.

[35] J. Mogul and J. Postel, "Internet Standard Subnetting Procedure", RFC-950, Standford University, 1985. http://www.ietf.org/rfc/rfc0950.txt