C-SWF Incremental Mining Algorithm for Firewall Policy Management

Ray-I Chang
Department of Engineering Science and Ocean Engineering
National Taiwan University
Taipei, Taiwan, Republic of China
rayichang@ntu.edu.tw

Keng-Wei Chang
Department of Engineering Science and Ocean Engineering
National Taiwan University
Taipei, Taiwan, Republic of China
r94525051@ntu.edu.tw
(Mr. Keng-Wei Chang is now with Trend Micro Incorporated, Taipei, Taiwan, Republic of China)

Abstract

As the number of security incidents had been sharply growing, the issue of security-defense draws more and more attention from network community in past years. Firewall is known as one of the most popular security-defense mechanism for corporations. It is the first defense-line for security infrastructure of corporations to against external intrusions and threats. A firewall will filter packets by following its policy rules to avoid suspicious intruder executing illegal actions and damaging internal network. Well-designed policy rules can increase the security-defense effect to against security risk. In this paper, we apply association rule mining to analyze network logs and detect anomalous behaviors, such as connections those shown frequently in short period with the same source IP and port. From these anomalous behaviors, we could inference useful, up-to-dated and efficient firewall policy rules. Comparing with the method proposed in [18], we utilize incremental mining to handle the increasingly changed traffic log data. The proposed method can highly enhance the execution performance in data analyzing. Experimental results show that the execution efficiency of our method is better than that of traditional methods when dealing with large-sized log files.

Keywords: Computer Security, Firewall, Policy Management, Data Mining, Association Rule

I. Introduction

Computer network brings out many benefits to facilitate corporate development, but also cause some serious security problems incurring critical damages to spoil corporate operations. CERT (Computer Emergency Readiness Team) Coordination Center statistic report [5] indicates that the number of security incidents had been sharply growing at over 150% per year. With the rapid growth of security risks, network security-defense draws more and more attention from network community. As we know, there are various protection mechanisms for network security risks, including firewall, honey-pots, intrusion detection system, etc. According to the report from Computer Security Institute (CSI) of FBI (Federal Bureau of Investigation) [21], the utility rate of firewall system almost amounts to 97% for corporations. Hence, surely firewall has been the most popular mechanism for corporations, which is the first defense-line for security infrastructure to against external intrusions and threats.

Firewalls usually function as routers which connect different network segments together. It is simply a perimeter defense device splitting network environment into internal (trusted) and external (distrusted) network for controlling and filtering incoming and outgoing network traffic. Its packet filtering decision depends on a set of policy rules (also named policy rule table) describing the security policy and posture the corporation takes, and thus to effectively avoid suspicious intruder executing illegal actions and damaging internal network. Based on the configuration, firewalls restrict the traffic flowing between the different networks. Often, techniques on the protocol layers such as packet filters, circuit proxies, and application level proxies are employed together.

A firewall is a network element that controls the traversal of packets across the boundaries of a secured network based on a specific security policy. A firewall security policy is a list of ordered filtering rules that define the actions performed on matching packets. Filtering actions are either to accept, which passes the packet into or from the secure network, or to deny, which causes the packet to be discarded. Simple packet filters usually use simple ordered lists of rules. A rule is composed of filtering fields such as protocol type, source IP (Internet Protocol) address, destination IP address, source port and destination port, and an action field. Each network field could be a single value or range of values. When a packet is received, the list is scanned from the start to the end, and the action associated with the first match is taken if the packet header information matches all the network fields of this rule. Often a “deny all” rule is included at the end of the list.

Usually, a firewall should block the following types of traffic. (1) Inbound traffic from a non-authenticated source address with a destination address of the firewall itself. It represents some type of probe or attack against the firewall. (2) Inbound traffic with a source address indicating that the packet originated on a node behind the firewall. It represents some type of spoofing attempt. (3) Inbound traffic containing ICMP (Internet Control Message Protocol) traffic. (4) Inbound or outbound traffic from a source address that falls within the address ranges set aside in RFC 1918 as being reserved for private networks. (5) Inbound traffic from a non-authenticated source address containing SNMP (Simple Network Management Protocol) traffic. (6) Inbound traffic containing IP source routing information. From a security standpoint, source routing has the potential to permit an attacker to construct a network packet that bypasses firewall controls. (7) Inbound or Outbound network traffic containing a source or destination address of 0.0.0.0 or 127.0.0.1 (local host). Such traffic is usually some type of attack against the firewall system itself. (8) Inbound or Outbound traffic containing directed broadcast addresses. It is often used to initiate a broadcast propagation attack. Some firewalls include internal functionality to combat these attacks, but this particular type of network traffic should still be blocked with rule set entries.

Since the first matching rule is always used, firewall policy is definitely the most critical component for determining overall firewall efficiency. Well-designed policy rule potentially can increase the security-defense effect, as well as provide protection against security risk. Without policy rules to guide firewall, the firewall itself may become a security problem. A policy rule dictates how the firewall should handle applications traffic. Before a policy rule is created, some form of risk analysis must be performed on the applications that are necessary for the organization. The results of this analysis will include a list of the applications and how those applications will be secured.

Ideally firewalls should have a general picture of what security threats the organization network face. However, this will require knowledge of the vulnerabilities associated with each

application. It is definitely not a easy job. In this case, the deployment of firewall system only provides basic security defense. It cannot avert unknown intrusion behavior. The firewall is belonging to static mechanism and totally depending on the configuration of policy rule to against known abnormal connection behavior. Thus, for practical concern, firewall policy table must be continuously updated to improve its defending efficiency. On a small system, it can be done manually to keep the rule set as simple as possible. However, as the number of security incidents had been sharply growing, network managers may not build firewall rule sets to be as specific as possible with regards to the network traffic they control. How to utilize the high-speedy computing ability of computer to design a set of policy rules for improving firewall efficiency is an important research topic.

Firewall system provides large amount of log data to inform what is going on at the entire network environment. Firewall logs can be collected and analyzed to determine what types of traffic have been permitted or denied, what users have accessed various resources, and so on. Therefore, the analysis of firewall log not only could enable network administrators to get insight into their network security state, but also to find out whether rule table exist any leaks or error-configurations. Researches [18] [30, 31] show that using firewall logs to analyze necessary policy rule is a feasible method. However, when planning to analyze firewall log data to discover other up-to-date policy rules for security efficiency improvement, there are two following important challenges should be confronted.

(I) How to clearly discover useful policy rules from complex and unorganized firewall log data? Due to the network log data contains a lot of noisy information, it is not easy to effectively determine suspicious behavior rule. For example, infer connection pattern in Protocol = X, Source IP = Y and Source Port = Z is possibly an abnormal connection behavior.

(II) How to efficiently handle the large amount of log data because of the ever-changing network environment, the firewall log data in need of constantly managing, reviewed and analyzing by expertise experiences. Furthermore, as the increasing of popularity of internet, the logging information also becoming more complicated. Conventional hand-written way to analyze log data is error prone, costly and inefficient.

In order to solve these limitations, it is worthy to know how to utilize the high-speedy computing ability of computer and design a set of effective algorithms to analyze huge amount of log data, and further to mining important and useful information for improving firewall overall efficiency and maintaining network security condition. It is the purpose of this paper to utilize data mining technique to process these complex firewall logs and extract valuable information, such as connections those shown frequently in short period with the same source IP and port. With those characteristics of frequent connection pattern in network traffic, our approach could discover and generate most useful policy rules for helping network administrator to timely optimize firewall policy table instantaneously and to insure protected network in security.

The approach proposed by Golnabi et al. [18] processing large volume of dynamic log data by conventional Apriori algorithm, consuming lots of computing costs and causing efficiency overhead of overall system. Besides, their method only analyze high-frequent traffic pattern thus can not completely infer implied information. In this paper, we apply association rule mining to analyze network logs and detect anomalous behaviors, such as connections those shown frequently in short period with the same source IP and port. From these anomalous behaviors, we could inference useful, up to dated and efficient firewall policy rules. Comparing

with the method proposed in [18], we utilize incremental mining to handle the increasingly changed traffic log data. The proposed method can highly enhance the execution performance in data analyzing. Experimental results show that the execution efficiency of our method is better than that of traditional methods when dealing with large sized log files.

II. Proposed Method

1. System Architecture

Firewall system needs a series of procedures, including log analysis, rule update, and configuration, to continuously maintain inner policy table for facilitating its security efficiency, hence, it is definitely a costly and error prone job for large networked organization. As a result, the purpose of research is trying to design a more operative architecture with the use of data mining technique and enabling system with the ability to reflect current traffic characteristic, and practically supporting corporations to optimize and validate their firewall policy. Our work would mainly focus on developing a set of fortified methods to improve the deficiencies existing in past work [18], including use incremental mining algorithm to advance the inefficient Apriori when dealing with the dynamic log data. Not only can analyze suspicious traffic behavior more precisely, but also can process large volume of log data more effectively, further to facilitate overall performance. Our proposed architecture could be presented as shown in Figure 1.

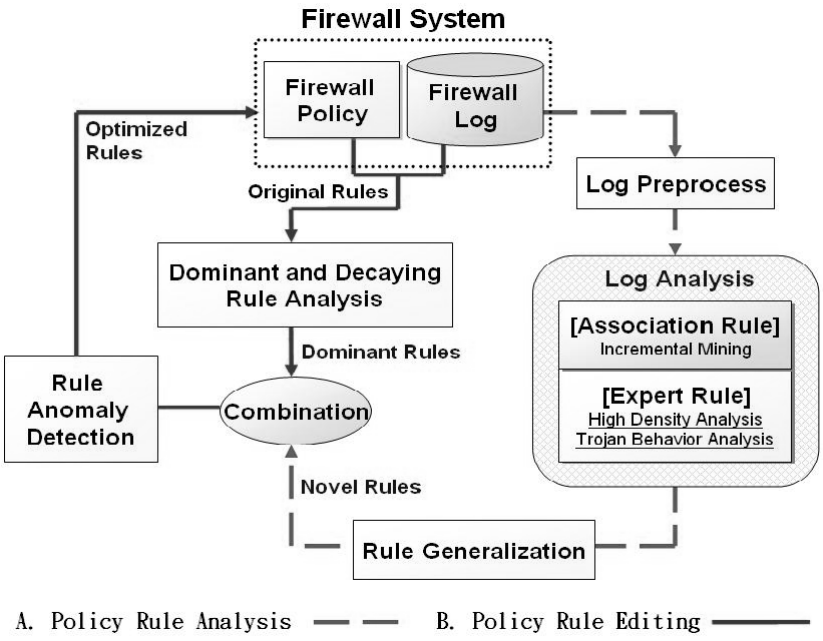

Figure 1: Proposed firewall policy management system architecture

In Rule Analysis procedure, the process consists of following three modules: log preprocess, log analysis, and rule generalization. Firewall log data would be firstly parsed into log preprocess module to extract primary attributes of log data: [Date], [Time], [Protocol], [Source IP] (Src_IP), [Source Port] (Src_Port), [Destination IP] (Dst_IP), [Destination Port] (Dst_Port), and [Action], simplifying raw firewall log data for facilitating the processing efficiency. Then, system will utilize our proposed log analysis methods to derive valuable

traffic rules from preprocessing log data. After log analysis step, a collection of traffic rules would be generated, and then system would perform rule generalization to generalize a set of novel rules reflecting current environment from previous results [18]. Above all, the essential intention of this phase is mainly to apply data mining techniques for discovering most significant policy rules, but how completely and efficiently process the large amount of log data will be explicitly introduced in latter section.

After rule analysis procedure, system would get into rule editing procedure. In this phase, system would firstly perform the dominant and decaying rule analysis [18] to identify each rule's weight for reordering its order and optimize firewall rule filtering efficiency. Here, "decaying" means rule is inactive or useless in this period of time, and "dominant" means rule is significant and active for covering most portion of firewall traffic. Afterwards, system will combine the original rules (after dominant and decaying rule analysis) with novel rules (from rule analysis phase). After this old new combination, system will apply the anomaly detection module to detect any conflicting rules existing in this integrated rule set. This is the most important part of this phase, because after there would be exist several policy errors between rules, such as blocking an legal traffic or approving traffic going to an inexistent service. Through this module, system could correctly update firewall policy with a set of newly optimized rules, and ensure rules without any conflicts and redundancies.

In practice, network administrator's supposed to iteratively and regularly perform these two system procedure. Furthermore, proper and sufficient log data is required for analysis, because only in light of the right data could rightly derive valuable information from log data for firewall policy optimization (e.g., everyday to analyze log data during latest three day).

2. Firewall Log Analysis

As stated above, firewall policy management for corporations is definitely a complex and intangible job. In our work, the main procedures of log analysis as shown in Figure 2, through two levels of processes to analyze and generalize a set of up to date policy rules.

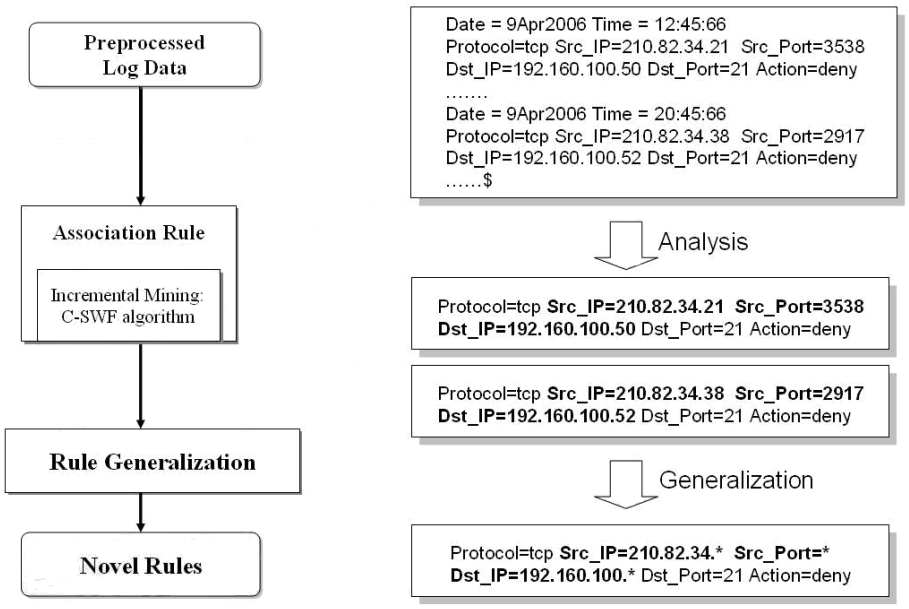

Figure 2: Rule analysis procedure for firewall policy rules generalization

Firstly, utilize our proposed log analysis methods to analysis the pre processing log data. It would generate a set of primitive rules with complicated and duplicated outcomes. Accordingly, in order to minimize the size of primitive rules, we use a generalization method to aggregate redundant rules without changing the policy results, and thus we could obtain a set of newly and practical firewall policy rules. However, in real system environment, the firewall log data is belonging to dynamical database accumulating as firewall system in operation. Particularly, for corporate large network, firewall system would generates millions log data at each day, and further, the need of dealing with large amount of logs certainly imposes additional work burden on policy management system, and causes log analysis step would be easily becoming the efficiency overhead of system. In this section, therefore, we would try to discuss several critical factors that will affect system efficiency, and to propose a set of methods to address these problems. Association rule analysis is a data mining technique that conducted the database and summarized meaningful relation among log items to help generate practical rules. However, using the conventional static mining method to process dynamic log would to restrict the execution performance for overall system.

As shown in Figure 3(a), using the conventional static method to mining association rule is not an efficient way for dealing with incremental accumulated log data. When new entries are being continuously added, static method can not just handle the new added entries, and then need to scan the whole database as mining scope changing. At the same time, considering the illustration in Figure 3(b).

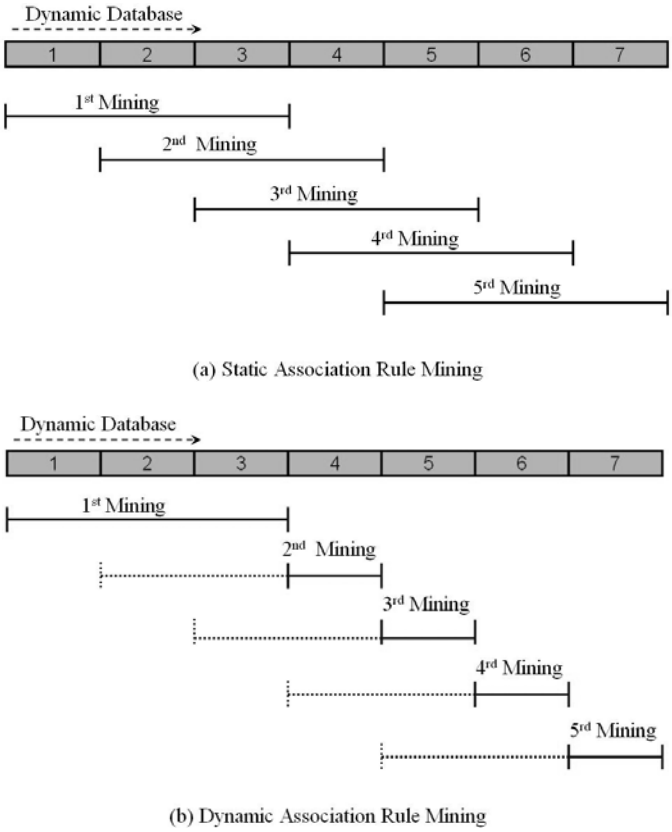

Figure 3: The difference between static and dynamic mining

If we change to apply dynamic mining algorithm for processing the dynamical log data, it could effectively update the association rule results without repeating to scan the entries that

had been processed early. As stated above, in order to practically enhance the efficiency of firewall policy management system, we propose to utilize an incremental association rule mining method to substitute for conventional static method [18]. Such an improvement would be able to effectively speed up the system performance. The concept of dynamic mining was proposed to address the problem for processing dynamic database, which is called incremental mining. Such as the web log data would be dynamically changed with time, and thus if we could use the rule results previously acquired to facilitate successive discovery process, it would be successful to enhance the processing efficiency.

In 1996, the first algorithm FUP (Fast Update Algorithm), proposed by Cheung et al. [8], was basically an Apriori liked algorithm introduced to handle the dynamic database. By storing related counts of all frequent item sets found in previous mining process to reduce the cost of scanning database. But unfortunately, FUP without the ability to confront the changes of deletion of database and only could be used in the changes of insertion. Afterward, they proposed an extension work named FUP2 [10] to address this problem of deletion case, however, in essence, this algorithm need to generate large set of candidates and cost many times of scanning database ,and thus diminish its mining efficiency. In recent year, Lee et al. [4] proposed a brand new method, SWF (Sliding Window Filtering), to remedy these problems. With the technique of sliding window filtering process, SWF partitions the database into several partitions to reduce the running time by iteratively filtering unnecessary candidate item sets with its support threshold during the scanning of each partition.

As shown in Figure 4, the concept of SWF algorithm is to maintain 2 item sets table, named CF (Cumulative Filter). Specifically, cumulative filter would be preserved to the next mining process, and be updated as the database incrementally changing. Moreover, SWF also equips with a scan reduction technique, which was firstly proposed in DHP (direct hashing with pruning) algorithm [17], to generate all K candidate item sets: CK (K > 2). Hence, SWF algorithm not only could shorten its cost for generating the large item sets, but also reduce the times of scanning database to twice.

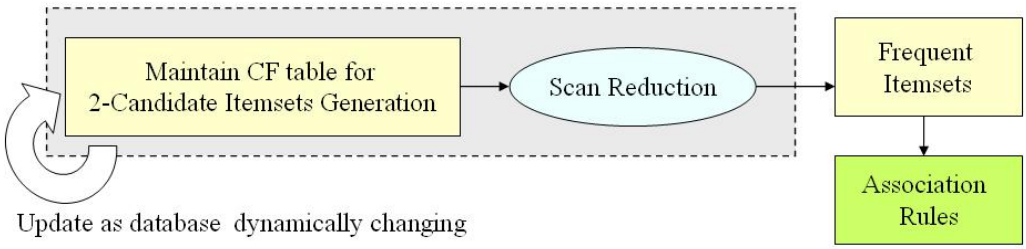

Figure 4: The concept of SWF algorithm for mining association rule

Through the sliding filtering method to incrementally scan the database, it would effectively be applied to address the problem for dealing with the dynamic log database. In this paper, therefore, on the basis of SWF algorithm, we proposed an advanced log analysis method to explore firewall log data and speed up its processing efficiency. As we know, when dealing with large size of log data, association rule algorithms are almost prone to cost lots of running time for scanning overall database, and thus the inefficiency of algorithms would be easily restricted. However, if aiming at the firewall log data to analyze its composition property, we could find out there are a lot of connection patterns repeating among firewall log data. This is

due to firewall log data is generally composed of few fields with limited value rage in practice, and lead to the repetitive patterns highly covered most of the entries of the log data [19].

In this regard, we could definitely consider that most of entries are almost composed of small numbers of connection patterns in log data. For this reason, if we could use compression technique to condense the frequent connection patterns, all the information of complex log data would be easily summarized without losing any important value. Through this simplified structure of log data, association rule mining algorithms could effectively reduce the cost of time for searching large item sets as counting the support value. In the field of information retrieval, suffix tree [13] and PAT tree (Patricia Tree) [15] are two common approaches applied to compress the processing data for set up an index table, which is helpful for finding out the repeated string and counting its count. Thus, on the basis of conception, we try to propose a method extending from SWF algorithm. Before the first time of database scanning, we try to add on a log compression procedure for enhancing the execution efficiency for log data mining.

In this paper, we extending the SWF algorithm by compressing the overall database into a simple tree structure (as shown in Figure 5) to efficiently improve the efficiency of mining process. This Compression log tree based SWF incremental algorithm (abbreviated as C-SWF) would make the analysis performance of firewall log data more effectively with the use of tree structure to sequentially condense each field of log entries and record the cumulative count of each item. In this compression structure, connection patterns in common filed of value would be intuitively combined and thus algorithm could apparently reduce the size of overall database without losing any original information among each log.

Protocol	Src_IP	Src_Port	Dst_IP	Dst_Port	Action
A2	B1	C1	D1	E3	F2
A1	B3	C1	D2	E2	F1
A2	B2	C1	D2	E3	F1
A1	B3	C2	D4	E1	F1
A1	B3	C1	D2	E2	F1
A1	B3	C2	D4	E1	F1
A1	B3	C2	D4	E1	F1
A2	B2	C1	D2	E3	F2
A2	B2	C1	D2	E3	F2
A1	B3	C1	D2	E2	F1

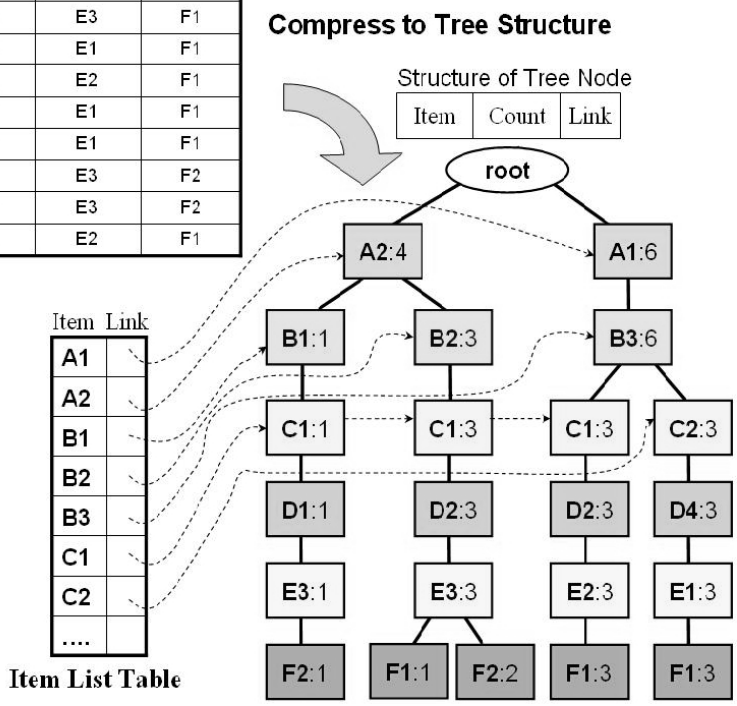

Figure 5: The structure of compression log tree

As illustrated in Figure 5, the most particular feature of compression tree structure is that different level represents different field of log: level 0 is [Root], level 1 is [Protocol], level 2 is [Source IP], level 3 is [Source Port], level 4 is [Destination IP], level 5 is [Destination Port], and level 6 is [Action], and thus the same value within distinct level would be regarded as different item in the mining process. Such as, 140.112.26.188 in field of [Source IP] and [Destination IP] mean two different items. And accordingly, different tree path indicates different connection pattern, and thus the height of tree would be keep in constant.

Except for the root of compression tree, other nodes are the item nodes including the field of "item value", "cumulative count", and a "node link", where item value registers the value this node represents, cumulative count writes the number of item represented by portion of the path reaching this node, and node link would link to next node in the compression tree carrying the same item value, or null if there's none. Further, in order to facilitate tree traversal, an item list is constructed, where each entry in item list consists of two fields: the field of item value and a node link, for recording each unique item, and thus using the node link of each entry to point out the node whose item value firstly occurs in the compression tree. As new item value inserting into the compression tree, algorithm would firstly check item list to make out if it is a first occurrence or not. If not, it would trace the last occurrence node then point out the newly added node via node link of the last one. Thus, the link path of a specific item value could be gradually forming up. By utilizing this constructed item list to traverse the compression tree would be more effectively integrating the information representing from firewall log data and increasing the processing performance of analysis algorithm.

With the utilization of compression tree structure, the proposed C-SWF algorithm only needs scanning probable tree path involved with corresponding value of item set for discovering C2 from cumulative filter table by accounting its cumulative frequency and extracting frequent item sets from C2 ~ CK (K > 2) by accounting each support value. For example, in Figure 5, the process of discovering the support of item set [C1, E3] firstly would get to scan the Item List to search for every item node from the constructing C1 link path, and then accordingly traverse down each associated path to check whether another item E3 is exist on this path. If existing, return to its cumulative count in compression tree. Finally, the support of [C1, E3] is 4.

With the utilization of this proposed compression tree, the proposed C-SWF algorithm is much less costly than just scanning an uncompressing firewall log data, not only to minimize the memory using for data item but also effectively to diminish the repeated data and unnecessary scanning process, and thus facilitating the performance of frequent item sets generation of SWF algorithm. Notice that the proposed C-SWF algorithm would firstly start to partition the firewall log data into several partitions for scanning respectively. Thus, here we take Date as a partition unit. And each partition would be indicated as P(month, day) that contains the firewall log entries on "day of month". Such as, P(5, 2) represents the firewall log partition on the 2nd day of May.

The incremental property of the proposed C-SWF algorithm is composed of two procedures. In preprocessing procedure, which is an initial execution process of whole incremental step, each partition would be firstly compressed to be a compression tree: PT(month, day). PT(5, 1) would be scanned sequentially to discover promising 2 item sets among every tree paths and accumulated their frequency count in cumulative filter table for the generation of candidate item sets C2 as shown in Figure 6.

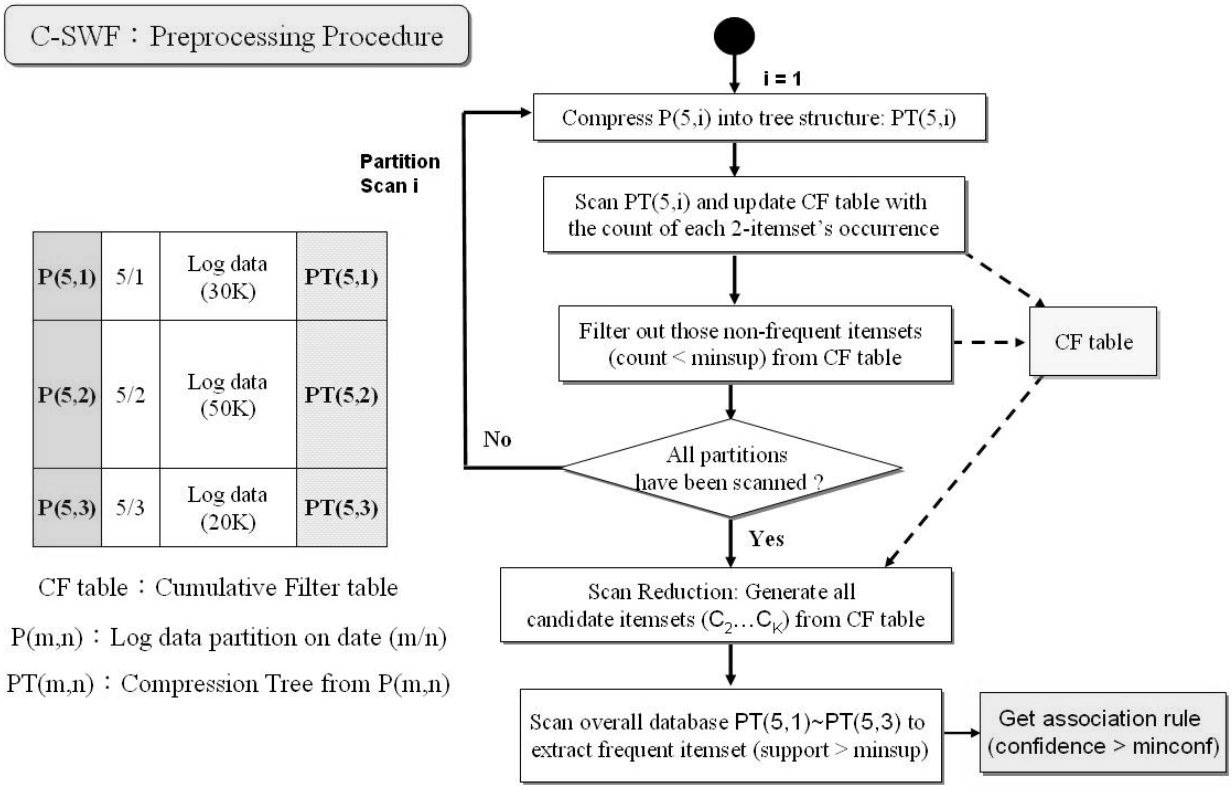

Figure 6: C-SWF algorithm: flow diagram of preprocessing procedure

After scanning the partition tree, the cumulative filter table would be examined to filter out the item sets whose counts are less than "minsup" (minimum support) threshold, and its gathering information would be retained to next partition. Similarly, after scanning other partitions, PT(5, 2) and PT(5, 3) and the cumulative filter table would be maintained sequentially until the last partition scanning is finished. Finally, we get a set of candidate item sets C2, which is closing to large item sets L2, from the cumulative filter table. Afterward, applying the scan reduction technique to generate all other candidate item sets: CK (K > 2), and then proceeding to second time of scanning for discovering each support value of whole candidate item sets: C2 ~ CK. Eventually, algorithm could derive the effective frequent item sets from filtering out unsatisfied item sets by the minsup value. After that, we could finally produce the association rule with suspicious behavior from the valuable frequent item sets.

Consider Figure 7, when the database is updated from P(5, 1) ~ P(5, 3) to P(5, 2) ~ P(5, 4) where P(5, 1) is deleted and P(5, 4) is added to the database, the mining algorithm would transform into Incremental Procedure. Firstly, this incremental step would load the cumulative filter table from the last mining procedure and reuse its information, and then check if the counts of item sets in the cumulative filter table will be affected by the deletion of P(5, 1). So, we would decrease the counts of affected item sets that are supported by P(5, 1) and reset its start position. Then, update the cumulative filter table to filtering out the item sets that are unsatisfied the minsup threshold.

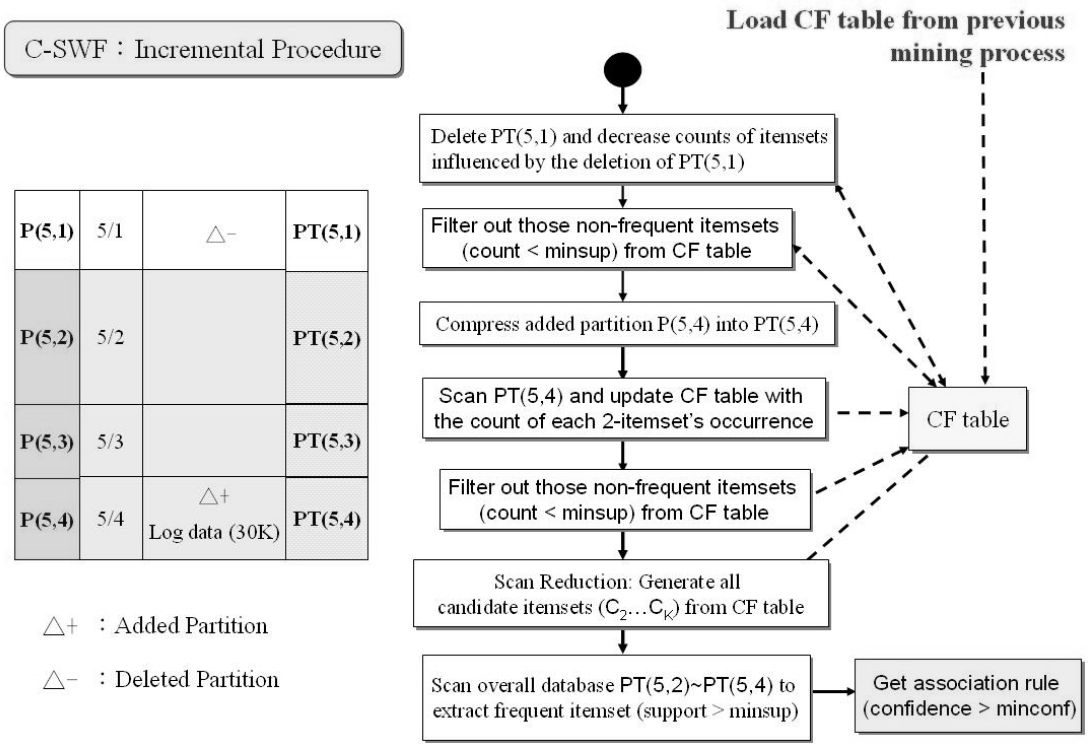

Figure 7: C-SWF algorithm: flow diagram of incremental procedure

Afterward, similar to the step in Preprocessing Procedure, algorithm would compress the new added partition P(5, 4) into compression tree PT(5, 4), and scan every tree path to count each possible 2 item set for the update of the cumulative filter table. Base on this concept, once the database is updated, updating the cumulative filter table and utilizing its valuable information remaining from last mining process would be effectively mining large amount of firewall log data without repetitively processing whole database. Finally, we could apply the extracted frequent item sets to discover latest association rule from abnormal behavior after database is updated.

III. Experimental Results

In this work, we have implemented our proposed log analysis methods, including C-SWF association rule analysis method to incrementally process the firewall log data. These proposed methods would be applying to generalize practical and up to date policy rule for enhancing the firewall efficiency. To evaluate the performance gain of our proposed log analysis methods, we compare them with previous methods by processing a set of firewall log data to obtain the improvement results. Our experimental platform is Pentium D 3.0 GHz with 2G of main memory running on Windows Server 2003. The simulation program was coded in C++. The evaluation dataset is a set of firewall log files provided by CSIST (Chung Shan Institute of Science and Technology), and using this large amount of dataset (including nearly 200000 firewall log records) to evaluate the performance of proposed algorithms.

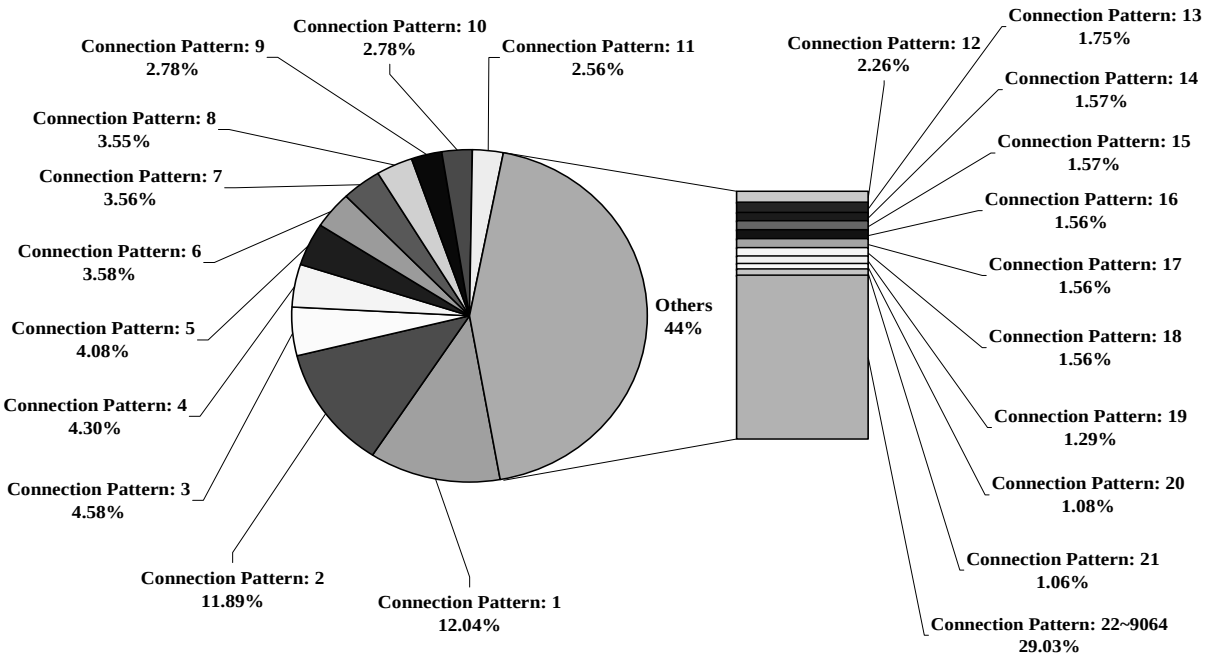

Figure 8: Statistics of connection pattern distribution in firewall log data

In order to realize the characteristic of firewall log data, we try to find out the each connection pattern of firewall log data, and conclude the distribution of patterns among them. After this study, the result of our investigations are shown in Figure 8, we could realize that there are totally 9064 different kinds of connection pattern in this firewall log data containing nearly 200000 log entries, and further, learn that most of log entries are composed of small numbers of connection patterns. Thus, the preceding 11 patterns constitute over half numbers of firewall log file, the others are composed by remaining 9053 patterns. Through this statistical analysis, we consider that more certainly, the log compressing step is helpful for processing the firewall log data and enhance the performance of algorithm.

In this section, we evaluate the performance of our proposed C-SWF Algorithm for processing the firewall log dataset. The purpose of experimental design is try to compare the speed of three algorithms, Apriori, SWF, and C-SWF for mining our tested log dataset and to generate a set of same results (frequent item sets). In the experiment one, we test the mining efficiency when handling large amount of log data, using nearly 200000 entries to compare the performance of these three algorithms in single run process. The minsup is set to different level (from 0.1% to 2%) to observe the speed variation.

Figure 9 show that when the minsup is high (2%), the running times of three algorithms are similar. That is because of a small number of frequent item sets produced that limited the execution time. However, when the minsup is low, the difference in the execution time becomes obviously very large. Because when the minsup is low (0.1%), there are many candidate item set produced that scaling up the processing cost. In view of that, the performance of the proposed C-SWF algorithm is surpassingly faster than Apriori algorithm, and also outperform original SWF algorithm. And further, the details of execution time of three algorithms in minsup = 2% shown that Apriori and SWF algorithms have to scan the whole raw dataset and need spending a lot of time to discover frequent item sets. As for the proposed C-SWF algorithm, it could save large amount of time in updating the cumulative filter table and

producing large item sets by scanning the compressed log tree due to the preprocessing procedure of algorithm. As a result, this experiment shows that the proposed C-SWF algorithm could efficiently enhance the log mining performance when mining large amount of log data.

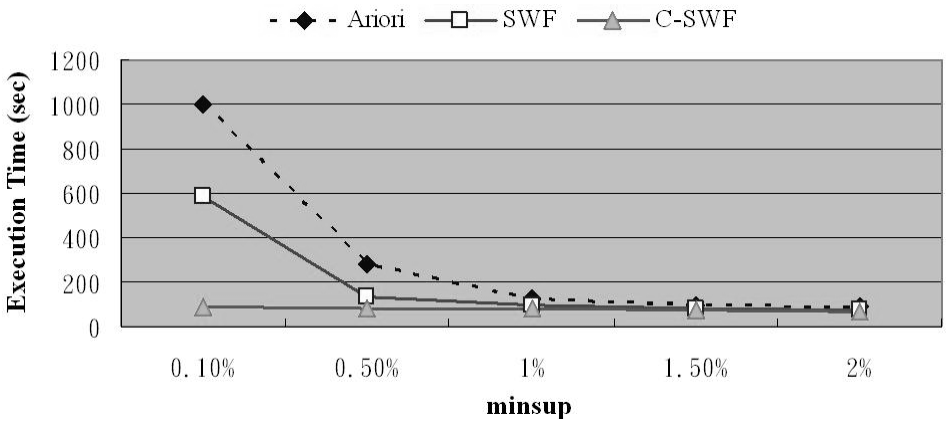

Figure 9: Experimental results when handling large amount of log data

In the experimental design of experiment two, in order to evaluate the performance changes of three algorithms (Apriori, SWF, and C-SWF) as the size of processing data increasing dynamically, we assumed that the firewall log data would be accumulating and the size of data would be scaling up at each round of processing (increasing 20 K / Round). Explicitly, we set the size of log data increasing from 20 K to 100 K with minsup = 0.1% to comparing these three algorithms.

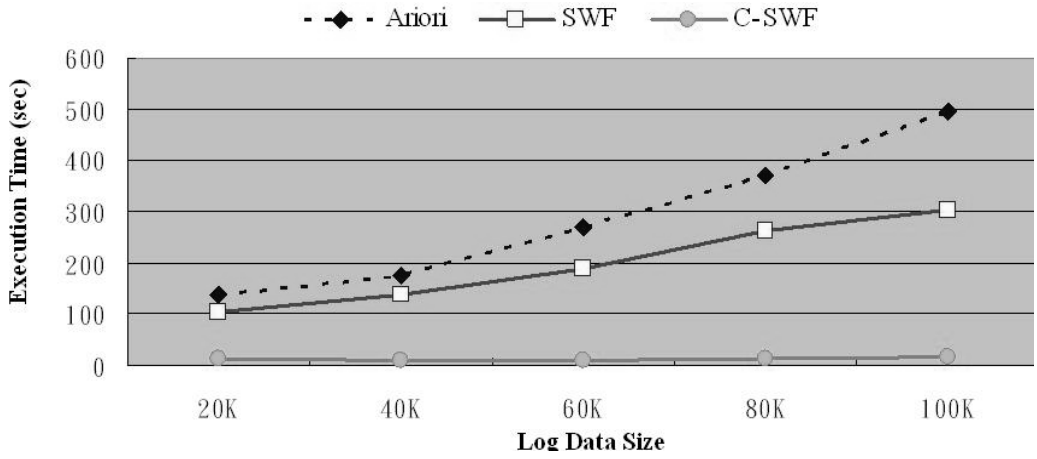

Figure 10: Experimental results when handling incremental log data

From the experimental results as shown in Figure 10, we observe that as log data increase the processing efficiency of Apriori algorithm was less than SWF based algorithms. This is due to Apriori need to repeated process old entries when new entries are added to the dataset, however, SWF based algorithms could utilize its incremental mining advantage and just have to process the newly added entries for omitting the consumption of repetition. Notably the execution time of the proposed C-SWF algorithm would not sharply grow as the data increasing and then leads to prominent performance improvement owing to its log compressed procedure.

In the view of detailed execution time, we could perceive that the proposed C-SWF algorithm cost less time for updating the cumulative filter table and producing large item sets than Apriori and SWF algorithms because of the utility of applying compressed log tree. Therefore, the proposed C-SWF algorithm is more suitable for handling the dynamically increasing data.

In experiment three, in order to evaluate the efficiency of SWF based methods in process of mining dynamically changing data, we test these three algorithms to examine the firewall log data in sliding window condition (delete and add) with minsup = 0.1%. Explicitly, test dataset was set to increase 10 K entries in each round with scanning window size = 5, hence we scanned the latest 5 part of data (50 K entries) and thus the algorithm would add the latest 10 K and delete the most beginning 10 K entries at each round of mining task.

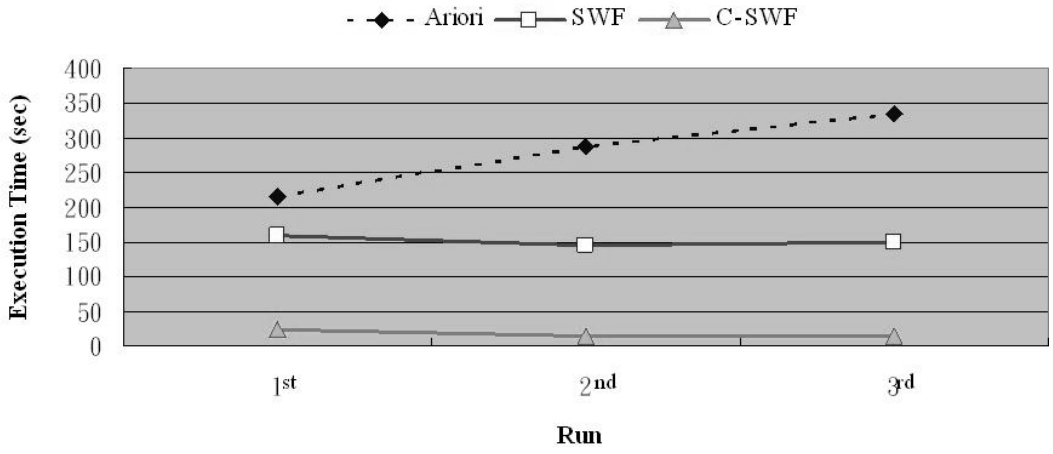

Figure 11: Experimental results when handling log data with dynamically sliding window change

The experimental results in Figure 11 show that Apriori is limited to its static feature which affects its efficiency. SWF and the proposed C-SWF algorithm bring out their advantage of incremental mining and only have to spend more time to process log data in first round of task. Consequently, the performances of SWF and C-SWF clearly outperform Apriori algorithm in dynamical data analysis. Same as preceding experimental observation, from the study of detailed execution time of each algorithm, the proposed C-SWF algorithm incorporates the compressed log procedure to efficiently enhance the scanning speed and reduce the cost of updating the cumulative filter table and producing large item sets. Relative to SWF algorithm, the proposed C-SWF algorithm is such an operative algorithm for processing firewall log data.

IV. Conclusion

In this paper, we present a set of reinforced log analysis approaches to address the problem of firewall policy management, which is proposed to apply data mining techniques to excavate appropriate policy rules from firewall log and to make policy rules with the ability to reflect current traffic characteristics. From the experimental results, the proposed log analysis method can efficiently enhance the execution performance for dealing with dynamic log data, and further, the proposed fast algorithm successfully avoid log analysis procedure becoming a bottleneck of the system. Enabling network administrator to immediately derive significant policy rules from firewall log and help firewall policy rules dynamically updated and optimized with current traffic conditions. In our future works, we intend to extend our current approaches

and architecture for advanced researches including to enhance the accuracy of association rule analysis for firewall policy mining, to deploy other data mining techniques for log analysis such as clustering algorithm, to develop a distributed architecture for multiple firewalls environment by applying distributed algorithms [9] [23, 24], and to integrating with other security mechanism such as IDS.

Acknowledgement

This research is partially supported by NSC (National Science Council) Taiwan under grants NSC 97 -2410-H-002-116- and NSC 95 -2623-7-002-004- D-. The author would like to thank CSIST (Chung Shan Institute of Science and Technology), Taiwan, to provide the evaluation dataset.

References

[1] http://www.infosecwriters.com/text/resources/pdf/top-log-analysis-mistakes.pdf. A. Chuvakin, “Five Mistakes of Security Log Analysis,” netForensics, 2004.

[2] A. Savasere, E. Omiecinski, and S. Navathe, “An Efficient Algorithm for Mining Association Rules in Large Databases,” Conference on Very Large Data Bases (VLDB 95), 1995, pp. 432-444.

[3] A. Wool, “A Quantitative Study of Firewall Configuration Errors,” IEEE Computer, Vol. 37, No. 6, 2004, pp. 62-67.

[4] H. Lee, C. R. Lin, and M. S. Chen, “Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining,” ACM 10th International Conference on Information and Knowledge Management (CIKM 01), Nov 2001, pp. 263-270.

[5] http://www.cert.org/stats/cert_stats.html. “CERT/CC Statistics 1988-2003 – incidents reported.” CERT (COMPUTER EMERGENCY READINESS TEAM)/CC

[6] A. Barbara, J. Couto, S. Jajodia, and N. Wu, “ADAM: A Test-bed for Exploring the Use of Data Mining in Intrusion Detection,” IEEE SMC Information Assurance Workshop, SIGMOD Record, Vol. 30, No. 4, 2001, pp.15-24.

[7] A. Barbara, J. Couto, S. Jajodia, and N. Wu, “ADAM: An architecture for anomaly detection,” Applications of Data Mining in Computer Security, ISBN 1-4020-7054-3, Kluwer Academic Publishers, Boston, 2002, pp. 63-76.

[8] D. W. Cheung, J. Han, V. T. NG, and C. Y. Wong, “Maintenance of Discovered Association Rules in Large Databases: An incremental Updating Technique,” International Conference on Data Engineering, Feb 1996, pp. 106-114.

[9] W. Cheung, V. T. Ng, A. W. Fu, and Y. Fu, “Efficient Mining of Association Rules in Distributed Databases,” IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, Dec 1996, pp. 911-922.

[10] D. W. Cheung, S. D. Lee, and B. Kao, “A General Incremental Technique for Maintaining Discovered Association Rules,” International Conference on Database System for Advanced Applications, Apr 1997, pp. 185-194.

[11] E. Al-Shaer and H. Hamed, “Firewall Policy Advisor for Anomaly Detection and Rule Editing,” IEEE/IFIP Integrated Management Conference (IM 03), Mar 2003, pp. 17-30.

[12] E. Al-Shaer and H. Hamed, “Discovery of Policy Anomalies in Distributed Firewalls,” IEEE INFOCOM 04, Vol. 23, No. 1, Mar 2004, pp. 2605-2616.

[13] E. Ukkonen, “Constructing Suffix-trees Online in Linear Time,” Algorithms, Software, Architecture: Information Processing 92, Vol. 1, Elsevier, 1992, pp. 484-492.

[14] E. W. Fulp, “Optimization of Network Firewalls Policies using Directed Acyclic Graphs,” IEEE Internet Management Conference, 2005, pp.66-82.
[15] H. Gonnet, R. A. Baeza-Yates, and T. Snider, “New Indices for Text: PAT Trees and PAT Arrays,” Information Retrieval: Data Structures and Algorithms, Prentice Hall, 1992, pp.66 - 82.
[16] J. Han, J. Pei and Y. Yin, “Mining Frequent Patterns without Candidate Generation,” ACM SIGMOD International Conference Management of Data, 2000, pp. 1 - 12.
[17] J. S. Park, M. S. Chen, and P. S. Yu, “Using a Hash-Based Method with Transaction Trimming and Database Scan Reduction for Mining Association Rules,” IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 5, Oct 1997, pp. 813 - 825.
[18] K. Golnabi, R. Min, L. Khan, and E. Al-Shaer, “Analysis of Firewall Policy Rules using Data Mining Techniques,” 10th IEEE/IFIP Network Operations and Management Symposium, Apr 2006, pp. 305 - 315.
[19] K. Hatonen, J. F. Boulicaut, M. Klemettinen, and M. Miettinen, C. Masson, “Comprehensive Log Compression with Frequent Patterns,” International Conference on Data Warehousing and Knowledge Discovery (DaWaK 03), Springer-Verlag LNCS 2737, Sept 2003. pp. 360 - 370.
[20] http://www.usmd.edu/usm/adminfinance/itcc/appfirepolicy.doc. L. Brown, “An Approach to Creating your Firewall Security Policy,” USM.
[21] http://www.cpppe.umd.edu/Bookstore/Documents/2005CSISurvey.pdf. L. A. Gordon, M. P. Loeb, W. Lucyshyn and R. Richardson, “2005 CSI/FBI Computer Crime and Security Survey,” Computer Security Institute, 2005.
[22] M. Berry and B. Linoff, “Data Mining Techniques: For Marketing, Sales and Customer Support,” New York: John Wiley & Sons, 1997.
[23] M. J. Zaki, “Parallel and Distributed Association Mining: A Survey,” IEEE Concurrency, Vol. 7, No. 4, Oct-Dec 1999, pp. 14 - 25.
[24] M. Z. Ashrafi, D. Taniar, and K. Smith, “ODAM: An Optimized Distributed Association Rule Mining Algorithm,” IEEE Distributed Systems, Vol. 5, No. 3, Mar 2004, pp. 14 - 25.
[25] P. Verma and A. Prakash, “FACE: A Firewall Analysis and Configuration Engine,” Symposium on Applications and the Internet (SAINT 05), 31 Jan - 4 Feb 2005, pp. 74 - 81.
[26] R. Agrawal, T. Imielinski, and A. Swami, “Mining Association Rules Between Sets of Items in Large Databases,” ACM SIGMOD Conference on Management of Data, 1993, pp. 207 - 216.
[27] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” International Conference on Very Large Data Bases (VLDB 94), Sept 1994, pp. 487 - 499.
[28] R. Agarwal, C. Aggarwal, and V. Prasad, “A tree projection algorithm for generation of frequent item sets,” Journal of Parallel and Distributed Computing, Vol. 61, No. 3, Mar 2001, pp. 350 - 371.
[29] R. I. Chang, L. B. Lai, W. D. Su, J. H. Wang, and J. S. Kouh, “Intrusion Detection by Backpropagation Neural Networks with Sample-Query and Attribute-Query,” International Conference on Neural Information Processing (ICONIP 06), Vol. 3, No.1, 2007, pp. 6 - 10.
[30] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg, “Simulation Study of Firewalls to Aid Improved Performance,” Annual Simulation Symposium, Apr 2006, pp. 18 - 26.

[31] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg, “Traffic Aware Firewall Optimization Strategies,” IEEE International Conference on Communications, June 2006, pp. 2225 - 2230.
[32] S. Brin, R. Motwani, J. Ullman and S. Tsur, “Dynamic Item set Counting and Implication Rules for Market Basket Data,” ACM SIGMOD International Conference Management of Data, ACM Press, New York, 1997, pp. 255 - 264.
[33] S. M. Bellovin, A. D. Rubin, and W. R. Cheswick, “Firewalls and Internet Security, Repelling the Wily Hacker,” 2nd ed., Addison Wesley, 2003.
[34] W. Lee, and S. J. Stolfo, “Data Mining Approaches for Intrusion Detection,” the 7th USENIX Security Symposium, 1998, pp. 79 - 94.