Reducing packet delay through filter merging

Paul Comerford
Department of Computing and Mathematics
University of Derby
Derby, United Kingdom
p.comerford@derby.ac.uk

John N. Davies
Creative and Applied Research for the Digital Society (CARDS)
Glyndwr University, Wrexham, United Kingdom
j.n.davies@glyndwr.ac.uk

Vic Grout
Creative and Applied Research for the Digital Society (CARDS)
Glyndwr University, Wrexham, United Kingdom
v.grout@glyndwr.ac.uk

ABSTRACT
The use of packet filters has increased considerably due to the growth of Internet users and network services. A number of header fields must be examined by the filter, causing delay for each packet processed. The problem is compounded when considering multiple filters across a network. To maximize network performance, it would be desirable to minimize the number of packet filters for each path across a domain. Due to the interactions of rules between filters, the underlying network topology and the actions of dynamic routing protocols, it is computationally infeasible to implement this strategy by collectively considering all packet filters across the network. A simpler approach is the elimination of a filter by merging two filters on a common network segment. This work presents a novel packet filter merging algorithm using decision diagrams. A large number of practical and simulated experimental results are provided to demonstrate the effectiveness of the technique and possible enhancements are considered in the conclusion. The results show an average 20% performance improvement can be obtained using the technique.

Keywords
Network performance; Packet filter; ACLs; Network security.

1. INTRODUCTION
Packet filters are typically referred to as Access Control Lists (ACLs) and are an important tool for specifying inter-domain or intra-domain access control. They are also commonly used to select traffic for policy routing, Quality of Service and Network Address Translation. ACLs compare address fields in the Layer 3 and 4 headers of a packet and use this information to determine if a packet should be dropped or further processed by subsequent routers. ACLs are represented as an ordered list of rules, which typically examine 5 header fields in a packet: source and destination IP addresses, source and destination port numbers and protocol type. The first matching rule defines the action to take on a packet. Formally, an ACL defines a mapping between any possible packet and the decision to be taken on it, which can be permit or deny.

ACLs can be deployed at the edge of a domain, in this capacity they act as packet filtering firewalls to prevent untrusted traffic from entering a network and control the traffic that is permitted to leave a network.

Another common use of ACLs is for controlling access between different parts of a domain. The security policy for an organization may dictate that a particular network or service should not be accessible to another. ACLs can be configured for many types of network devices, such as routers, switches, hosts and dedicated firewall units.

Packet filtering performance has come under increasing scrutiny from the research community in recent years. In the early days of the Internet, networks and the number of running services were relatively small in comparison with today's modern networks. Link speeds for Local and Wide Area networks were also much lower and this was considered the dominant factor contributing to packet delay. As link speeds have increased, the delay associated with packet processing in routers has become more significant and received greater attention. The amount of delay will depend on a number of factors, including the hardware platform, underlying software implementation and the number of fields which must be examined. The number of networks and services has increased dramatically in recent years. This increase has led to the use of larger ACLs to control access to these networks and services.

The cumulative delay for a packet will increase considerably each time a packet is required to be checked against an ACL. It would be desirable to improve network performance by reducing the number of times a packet must be checked from source to destination. Any optimization would need to ensure that the network security policy is not compromised.

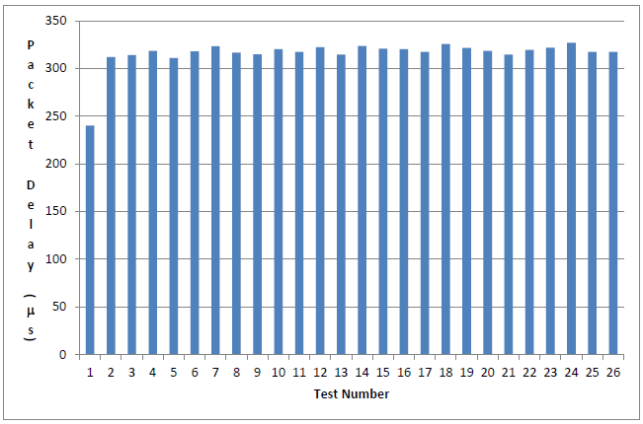

Figure 1. Packet delay subject to ACLs of increasing size.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

UCC '16, December 06-09, 2016, Shanghai, China © 2016 ACM. ISBN 978-1-4503-4616-0/16/12…$15.00

2. BACKGROUND

2.1 Improving filtering performance

The majority of previous work is concerned with improving packet filtering performance using a variety of different methods. Linear search remains the most space efficient form of packet filtering and several schemes have strived to improve performance by rule reordering and removing redundant rules. When using alternative data structures, a compromise must be made between filtering performance and the space required to store the classifier. A packet classifier for security filtering will typically use 5 fields: protocol number, source IP address, destination IP address, source port number and a destination port number. The total number of possible packets from these fields is the Cartesian product of the number of bits for each field = $2^8*2^{32}*2^{32}*2^{16}*2^{16}$ =2104 possible packets; it would be computationally infeasible to represent each packet as a single bit in an array or hash table.

2.2 Geometric representations

Packet classification can be represented geometrically. Filtering on a single field is equivalent to finding the highest priority interval on a number line (0-255 for protocol, 0-2^{32} for an IPv4 address and 0-65535 for a port number). Filters which use two fields form rectangles which enclose the set of packets which will match the filter. Each rectangle has a priority based on its position in the list. Formally, two-dimensional filtering is equivalent to finding a point which falls on the highest priority axes-parallel rectangle. This concept can be generalized to higher dimensions for filters which examine 3 or more packet fields. For a filter with d fields, a rule is a d-dimensional hyperrectangle in d-dimensional space with a priority based on its position in the original list [1]. The address fields of a packet can be considered a single point in geometric space as each address in the packet header is a single, discrete value.

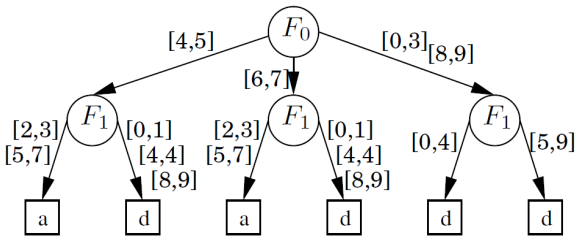

Figure 2. A Firewall Decision Diagram (FDD) [28].

3. RELATED WORK

3.1 Firewall verification and reduction

Most verification techniques use a form of decision diagram such as binary decision diagrams (BDDs), Interval Decision Diagrams (IDDs) or firewall decision diagrams (FDDs). Binary decision diagrams are data structures used to represent Boolean functions [2] and have been used to represent packet filters, first demonstrated by Hazelhurst [3] [4]. One of the limitations of BDD representation is their use of individual bits to represent rules. This is effective using FPGAs but less so on generic CPUs due to the overhead associated with extracting individual bits from a 32 or 64-bit word [5].

An off-line anomaly detection tool was developed in [6] by statically analyzing packet filters represented by BDDs. SAT-solvers have been used to test for anomalies in firewalls policies [7]. A method to measure and quantify firewall policies using BDDs was presented in [8]. A number of metrics were defined to quantify firewall security, management and performance. These included complexity of the rule set, average comparisons per rule and number of unused rules. A firewall verification algorithm requiring $O(n^d)$ (n = number of rules, d = number of dimensions) space complexity is provided in [9]. This approach requires probe packets to test reachability. An equivalence relationship between verification and redundancy protection is defined in [10]. Further anomalies which may exist for rules that filter on TCP connection flags were identified in [11] and off-line tool was produced to detect and eliminate such conflicts.

3.2 Rule reduction techniques

It is desirable to minimize the number of rules required to represent a security policy. The two primary reasons for this are improving filtering performance and reducing the amount of memory required to store the filter. List reduction is also important when considering hardware classification using Ternary Content Addressable Memory (TCAM) due to their limited capacity.

Yoon et al. proposed a novel method for ACL reduction using an algorithm called substitutional optimization [12]. Its purpose is to reduce the size of a rule set by substituting a group of rules with specific properties for a smaller group. The authors provide a simple example showing a reduced rule set. It is unclear if these specific conditions occur commonly in real-life classifiers.

TCAM Razor is a series of algorithms developed to minimize the number of entries required to represent a rule set in a TCAM [13]. The scheme obtains the smallest number of prefixes required to represent a single field using dynamic programming techniques. Results show that it provides a significant reduction in the number of rules. Other TCAM compression schemes use alternate forms of prefix encoding to reduce the number of prefix rules generated [14] [15] however these suffer from additional overhead for packet lookups.

Bit weaving is a technique which exploits the use of ternary masks in TCAMs [16]. This is achieved by bit merging and swapping techniques to reduce the number of entries. The overall compression achieved by this technique will vary depending on the structure of the original classifier; however, it is possible to use bit weaving with other list reduction techniques to further reduce the number of rules.

The firewall compressor framework uses integer intervals instead of prefixes to represent the rule set. This problem is reformulated as a scheduling problem where each single field rule is a task and the goal is to obtain a schedule with the minimum number of tasks [17]. This technique was previously used by Suri to compress the number of entries in a routing table [18]. This approach is less suited to prefix-based rules.

Liu, et al. combined the techniques used in TCAM Razor and firewall compressor. This allows the compression to be customized to the constraints of the software representation [19]. If a prefix rule is required, then the prefix minimization algorithm is used; if range rule is required, then the optimal scheduling algorithm is used instead [19].

An early attempt at rule compression using geometric techniques is provided by Applegate et al. [20]. The problem is mapped to the rectilinear picture compression problem found in graphics software applications. The goal is to find the shortest list of rules required to generate a given pattern on a canvas where black and white represent permit and deny respectively [20]. The authors consider the special case in which all rules must be defined as strips which

span the entire canvas horizontally or vertically. A formal proof of NP hardness and an approximation algorithm is given for the 2-dimensional form of the problem. The paper assumes that ACLs are usually specified using 2 fields which is insufficient for security filtering.

Daly et al provide a framework for ACL range compression [21]. The ruleset is converted into a series of hyperplanes. Differences between adjacent planes are resolved by adding resolving rules followed by a merging of the planes. The process is repeated until all the planes are merged and then the entire process is repeated recursively until a 1-dimensional pattern remains which can be solved by using existing optimal algorithms. Results obtained from this technique show that it consistently outperforms the earlier firewall compressor algorithms by up to 30% for larger rulesets [21].

3.3 Optimizing Distributed Packet filters

The use of distributed ACLs or firewalls in a network can lead to suboptimal configuration. It is possible some of the rules configured across multiple routers may be redundant due to rules defined on another router in the network. In an extreme case, a large proportion of the rules may be redundant. Another possible cause of rule redundancy is due to the traffic being processed by the router. If a router has no route to a destination network, then any rule with the same destination address field will be redundant. Detecting this form of redundancy requires knowledge of the routing data and topology of the domain.

The first attempt to identify and remove inter-firewall redundant rules is given by Al-Shaer, et al [22]. The conditions which cause intra-firewall and inter-firewall anomalies are defined and discussed. An inter-firewall anomaly is present when individual firewalls along the same path perform different filtering actions on the same traffic [22]. Yuan, et al. developed a method for detecting and removing redundant rules across distributed firewalls using Binary Decision Diagrams (BDDs) and static analysis techniques [6]. The distributed firewalls are represented as a tree structure and set intersection and union operations are performed using BDDs [6]. Chen et al considers the problem of identifying redundant rules between 2 firewalls belonging to different administrative domains [23]. It is assumed that each domain cannot disclose its security policy to the other. The work identifies redundant rules that can be removed on a firewall due to the upstream firewall in the neighboring domain.

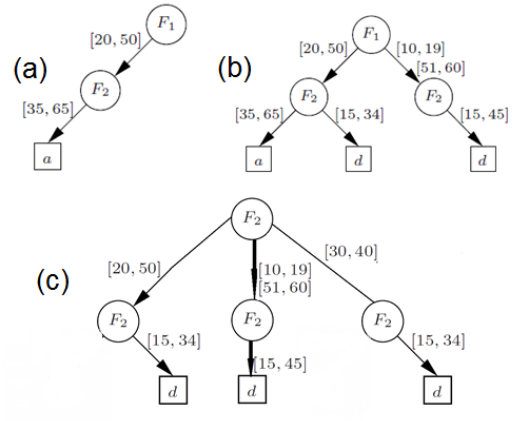

Figure 3. (a) and (b) FDD construction (c) pruning of permit space.

4. MEASURING FILTER DELAY

A number of independent tests were performed to evaluate the performance of trie-based ACLs using low end routers. Using a trie-based filtering technique, the packet delay is independent of the number of rules in the ACL. Due to the difficulty in obtaining large, real rulesets, synthetic ACLs were generated using a custom-built generation tool. The experiments were carried out using a Cisco 2600 router with 32mb RAM and advanced enterprise IOS installed which uses trie-based filtering. The results show that even a low-end router is capable of storing in excess of 50,000 rules. This provides the motivation for developing a list merging technique to reduce delay over a pair of routers by eliminating one of the filters. Wireshark was used to obtain a timestamp of the packet before and after ACL processing. Fig. 1 shows the packet delay for ACLs from 1000-50000 rules. The filtering delay remains constant due to the use of binary tries in the filtering process.

```
FDD pruning algorithm
1: Initialise edgeStack, lastNodes
2: while accept node incoming edge list is not empty
3:     add the acceptNode.IncomingEdges[0].SourceNode to lastNodes
4:     edge to remove = acceptNode.IncomingEdges[0]
5:     if edge to remove ∈ acceptNode.IncomingEdges[0].SourceNode.OutgoingEdges
6:           remove edge to remove from
acceptNode.IncomingEdges[0].SourceNode.OutgoingEdges
7:     remove first edge from acceptNode.IncomingEdges
8:     clear source, destination and label pointers of edge to remove
9: for each node in lastNodes
10:    if node has no outgoing edges
11:          push all incoming edges of node onto edgeStack
12: while edgeStack is not empty
13:    topEdge = pop first edge from edgeStack
14:    remove topEdge pointer from topEdge source node outgoing edges
15:    remove topEdge pointer from topEdge destination node incoming edges
16:    if topEdge destination node has 1 incoming edge
17:          remove topEdge destination node from nodelist
18:          deallocate all memory associated with topEdge destination node
19:    delete topEdge pointer
20:    if topEdge source node has no outgoing edges
21:          push all incoming edges of topEdge source node onto edgeStack
```

Figure 4. Pseudocode for FDD pruning algorithm.

4.1 Firewall Decision Diagrams

A Firewall Decision Diagram (FDD, Fig. 2) is a decision tree used for firewall verification by storing integer intervals at its edges. For any outgoing edge from a non-leaf node in the FDD, the integer intervals are completely disjoint [24]. The depth of the tree is d dimensions; this is usually bounded at 5 for a typical packet filter. Each leaf node of the FDD stores a decision: accept or discard. A packet will only match a single path from the root node to a leaf decision node [24]. The FDD represents the semantics of a packet filter [24]. Liu [25] provides algorithms to reduce the tree and generate rules from it.

4.2 A theoretical algorithm for removing ACLs

The merging of two separate ACLs into a single, semantically equivalent list requires that the exact set of packets which will be permitted or denied is known for both ACLs. Firewall Decision Diagrams are the most efficient method for representing the semantics of a filter. In the paper, firstly an FDD is constructed which represents the firewall, reduced and marked to reduce the number of generated rules. Finally, the rules are generated by performing a depth-first traversal of the FDD and removing any redundant rules.

In [25] a single ACL is translated to a FDD and then rules are generated from it. In this work, the first two steps are carried out, however this is followed by a pruning stage to remove all permit space (Fig. 3). Next, the second ACL is processed and appended to the partial FDD using the same construction and reduction algorithms. The remaining algorithms from the original work are

used to generate and compress the rules from the FDD. Additionally, a rule redundancy removal algorithm [26] is used to reduce the size of the ruleset after merging is complete. This is an essential step in the merging process as it is likely that many of the merged rules are redundant following a merge operation.

The construction algorithm builds the FDD using the packet header fields in each rule and the intersection relations between them. After construction, the FDD is reduced to minimize the number of outgoing edges at each outgoing node of the tree. This helps reduce memory consumption and the number of rules generated from the tree. The marking algorithm is used to identify tree edges which contain the largest number of intervals and wildcard them to further reduce the number of rules. A generation algorithm generates the rules from the FDD by traversing the fields at each edge of the tree from the root to leaf nodes. The compaction algorithm can detect and remove some redundant rules in the list. Lastly, the simplification algorithm simplifies rules which comprise of disjoint intervals for a single field, as these are not permitted by packet filter frameworks such as IPTables or Cisco ACLs.

4.3 A practical approach to list reduction

It is not possible to simply append the second list to the end of the first due to completeness property of an ACL. It is possible, to append the second list, without modification if and only if the first list contains rules from the deny packet space of the original first ACL. It is straightforward to prove this method will work for any combination of rules. A packet d which should be denied by the first ACL must match one of the rules from the deny packet space. Conversely, a packet p which should be permitted will never be matched by one of these rules. The first matching rule of p must be a rule in the second list as a result. This technique will work if and only if the set of deny rules exactly match the set of packets to be denied in the original ACL.

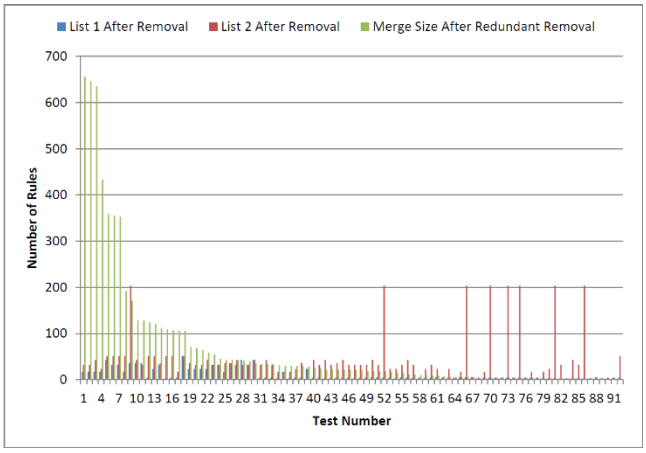

Figure 5. Merge sizes for remaining tests with real extended ACLs.

4.4 FDD Pruning

After the FDD is constructed from the first ACL, it is necessary to prune all paths which lead to an accept node. It is important to ensure that nodes with outgoing edges which do not belong to a path which leads to a discard node are also deleted. The resulting algorithm must also ensure that all discard paths from the original reduced FDD remain. The partial FDD remaining after the pruning stage is equivalent to the set of packets discarded by the first ACL. The algorithm performs a depth-first search starting from the terminal accept node. During the search, the edges and nodes leading to the accept node are removed from the FDD. The pseudocode is shown in Fig. 4. A stack is used to store the edges to be traversed during the search. A vector of node pointers is used to store the source nodes associated with the traversed edges. Lines 4-8: The first incoming edge of the accept node is examined and its source node is placed in the last nodes vector. After all incoming edges have been evaluated; all incoming edges of the nodes stored in the lastNodes vector are added to the stack to continue the search. Lines 12-21: While the stack still has edges to process, an edge is removed from the top of the stack and its pointer is removed from its source and destination nodes (lines 14-15). If the destination node of the edge only has a single incoming edge, then the node can be safely deleted as it cannot be part of a discard path. Firstly, the node is removed from the node list and its memory is de-allocated (lines 17-18). The process repeats until the stack becomes empty. After the pruning process is complete, the append algorithm is used again to append the second list to the pruned FDD. The resulting FDD represents the combined filtering action of the two ACLs.

The algorithm can be used by a router to merge two ACLs on a common link into a single ACL placed on the egress port of the most upstream router. The merge algorithm can be implemented in the operating system of each router and run every time an ACL configuration change takes place.

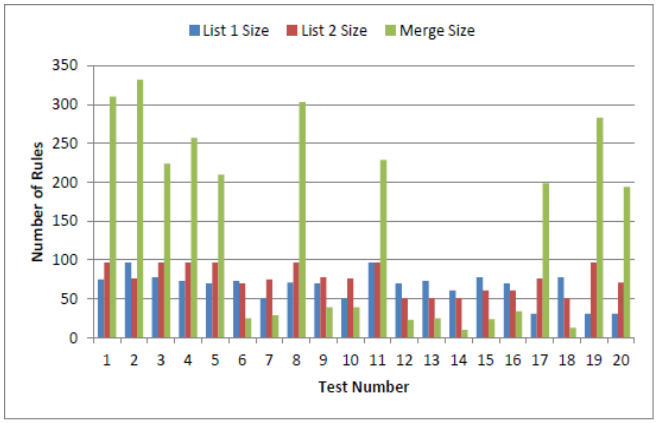

Figure 6. Merged rule sizes for 100 rules each in the inbound direction.

5. RESULTS AND ANALYSIS

Testing was performed to evaluate the effectiveness of the techniques used. The testing was carried out using both real ACLs sourced from a service provider and synthetic ACLs generated using an established model.

5.1 Tests using real ACLs

A small number of real Cisco ACLs were obtained from a service provider network ranging from a few to ~200 rules. Some only filter on the source IP address, however, most used multiple fields. The ACLs were tested against each other to generate merged lists. For some tests, there was no intersection between the set of permitted packets for either filter. This may be due to the original ACLs filtering on traffic from a different customer of the service provider. The time taken to perform the merge and the average size of the merged lists is recorded for comparison with the original lists. This process is repeated for each pair of ACLs in the set.

5.2 Tests using synthetic ACLs

Synthetic rule sets were used to provide further evaluation of the merge algorithm. To provide more realistic results, a typical

approach is to implement a model based on the statistics gathered from a large pool of real rules. The Perimeter rules model [27] produces random, non-uniform rule bases used for evaluating the performance of FDDs and similar data structures [27] [19]. The statistics used by the model are based on a total of 8500 rules acquired by the authors from various service providers.

Average results are used for analysis of each test set of ACLs. As expected, the average results show an increase in the number of rules as the size of the original lists increases (Fig. 5) the average size of the merged list stays relatively small up to 500 rules per original list. There was a markedly greater increase in the number of rules for the set of tests using 1000 rules and a similar increase for 2000 rules. The average packet delay (Fig. 6) increased slightly as the number of rules increased, which contradicts some of the earlier results which show little variation in delay for any number of rules. The number of rules generated was highly dependent on the degree of overlap between the original rules. Fig. 7 shows a large variation in the sizes of the merged list in comparison with the original. Generally, the time taken to perform the merge operation increase linearly with the number of rules (Fig. 8). The variation is again due to the difference in overlap between the rulesets.

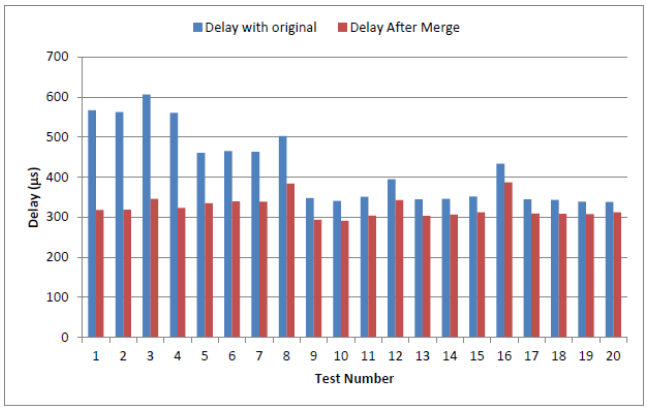

Figure 7. Packet delay for tests using real ACLs.

6. CONCLUSIONS AND FUTURE WORK

This work examined issue of packet latency due to filtering. Software-based packet filters use tries, a form of decision tree which stores a bit-based representation of a packet field at each edge of the tree. It was found that it is theoretically possible to merge two packet filters by firstly converting one of filters to a set of rules which represents the set of packets denied by the filter. The second filter can be appended to the first without modification as only packets permitted by the first filter will be processed by the second. A data structure used in firewall verification, the Firewall Decision Diagram (FDD), was used to convert a filter into its semantic equivalent. All possible permit rules are removed from the FDD which results in a list of rules representing the denied packets. Subsequently, the second filter is added by appending it to the FDD using the same construction algorithm. The resulting FDD represents a list which is semantically equivalent to the original pair of lists. An extensive set of test results was generated to demonstrate the effectiveness of the technique for reducing packet latency. The results show an average 20% performance improvement can be obtained using the technique.

Prefix fields use a different algorithm to provide an optimal number of prefixes for a single dimension using dynamic programming. For future work, it would be desirable to incorporate these algorithms into the framework and evaluate their effectiveness in comparison with the marking algorithm. Additionally, the complex substitutional optimization algorithm proposed by Yoon et al is also directly applicable to FDDs. It would be interesting to test the effectiveness of the technique using real world filters. Future work may also involve exploring other possibilities for merging filters, including grouping rules by mask sharing.

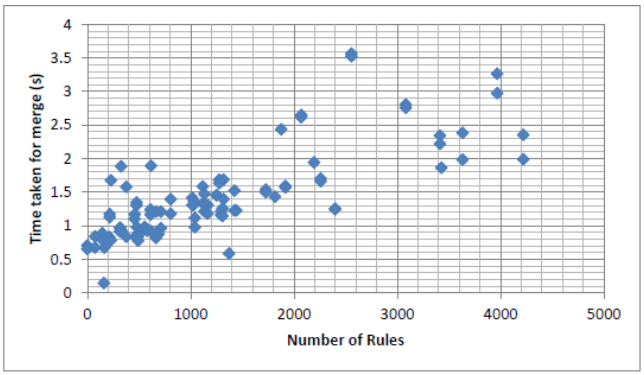

Figure 8. Scatter graph for merge time against number of generated rules

7. REFERENCES

[1] P. Gupta, “Algorithms for Routing Lookups and Packet Classification - PhD Thesis,” Stanford University, 2000.

[2] R. E. Bryant, “Symbolic Boolean manipulation with ordered binary-decision diagrams,” ACM Computing Surveys (CSUR) (September 1992), vol. 24, no. 3, pp. 293-318, 1992.

[3] S. Hazelhurst, “Algorithms for Analyzing Firewall and Router Access Lists,” 1999.

[4] S. Hazelhurst, A. Attar and R. Sinnappan, “Algorithms for improving the dependability of firewall and filter rule lists,” in DSN 2000 Proceedings International Conference on Dependable Systems and Networks, New York, USA, 25-28 June 2000.

[5] M. Christiansen and E. Fleury, “An interval decision diagram based firewall,” in 3rd IEEE International Conference on Networking (ICN'04), Guadeloupe, French Carribean, February 29- March 4 2004.

[6] L. Yuan, H. Chen, J. Mai, C. N. Chuah, Z. Su and P. Mohapatra, “FIREMAN: a toolkit for firewall modelling and analysis,” in Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, USA, 21-24 May 2006.

[7] Nelson T, C. Barratt, D. J. Dougherty, K. Fisler and S. Krishnamurthi, “The margrave tool for firewall analysis,” in USENIX Large Installation System Administration Conference, San Jose, USA, November 7-12 2010.

[8] S. Al-Haj and E. Al-Shaer, “Measuring firewall security,” in 2011 4th Symposium on Configuration Analytics and Automation (SAFECONFIG), Arlington, VA, USA, October 31 - November 1, 2011.

[9] A. Hrishikesh and M. G. Gouda, “Projection and division: Linear-space verification of firewalls,” in 2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS), Genoa, June 21-25 2010.

[10] A. Hrishikesh and M. G. Gouda, “Firewall verification and redundancy checking are equivalent,” in Proceedings IEEE INFOCOM, 2011, Shanghai, China, April 10-15 2011.

[11] S. Kim and H. Lee, “Classifying Rules by in-out Traffic Direction to Avoid Security Policy Anomaly,” Transactions on Internet and Information Systems, vol. 4, no. 4, pp. 671-690, August 2010.

[12] M. K. Yoon, S. Chen and Z. Zhang, “Reducing the size of rule set in a firewall,” in ICC'07. IEEE International Conference on Communications, Glasgow, Scotland, June 24-28 2007.

[13] C. R. Meiners, A. X. Liu and E. Torng, “TCAM Razor: A Systematic Approach Towards Minimizing Packet Classifiers in TCAMs,” in Proceedings of IEEE International Conference ICNP 2007., Beijing, China, 16-19 October 2007.

[14] B. Neji and A. Bouhoula, “NAF conversion: an efficient solution for the range matching problem in packet filters,” in 2011 IEEE 12th International Conference on High Performance Switching and Routing (HPSR), Taipei, Taiwan, July 8-11 2013.

[15] O. Rottenstreich and I. Keslassy, “Worst-case TCAM rule expansion,” in 2010 Proceedings IEEE INFOCOM, San Diego CA, USA, March 15-19 2010.

[16] C. R. Meiners, A. X. Liu and E. Torng, “Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs,” IEEE/ACM Transactions on Networking (TON), vol. 20, no. 2, pp. 488-500, 2012.

[17] A. X. Liu, E. Torng and C. R. Meiners, “Firewall compressor: An algorithm for minimizing firewall policies,” in IEEE INFOCOM 2008 The 27th Conference on Computer Communications, Pheonix, AZ, USA, April 15-17, 2008.

[18] S. Suri, T. Sandholm and P. Warkhede, “Compressing two-dimensional routing tables,” Algorithmica, vol. 35, no. 4, pp. 287-300, 2003.

[19] A. X. Liu, E. Torng and C. R. Meiners, “Compressing Network Access Control Lists,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 12, pp. 1969-1977, 2011.

[20] D. A. Applegate, G. Calinescu, D. S. Johnson, H. Karloff, K. Ligett and J. Wang, “Compressing rectilinear pictures and minimizing access control lists,” in In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, New Orleans, Louisiana, USA, January 7-9, 2007.

[21] J. Daly, A. X. Liu and E. Torng, “A Difference Resolution Approach to Compressing Access Control Lists,” in Proceedings of the 32th Annual IEEE Conference on Computer Communications (INFOCOM), Turin, Italy, April 14-19, 2013.

[22] E. S. Al-Shaer and H. H. Hamed, “Discovery of policy anomalies in distributed firewalls,” in Proceedings of INFOCOM 2004. 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, 7-11 March 2004.

[23] F. Chen, B. Bruhadeshwar and A. X. Liu, “A Cross-Domain Privacy-Preserving Protocol for Cooperative Firewall Optimization,” in Proceedings IEEE INFOCOM 2011, Shanghai, China, 10-15 April 2011.

[24] A. X. Liu, Firewall Design and Analysis, World Scientific Publishing Co Pte Ltd, 2010.

[25] M. G. Gouda and A. X. Liu, “Firewall design: Consistency, completeness, and compactness,” in IEEE 24th International Conference on Distributed Computing Systems, Hachioji, Tokyo, Japan, 24-26 March 2004.

[26] C. Meiners, A. X. Liu and E. Torng, Hardware-based Classification for High-Speed Internet Routers, 1st Edition ed., Springer, 2010.

[27] D. Rovniagin and A. Wool, “The Geometric Efficient Matching Algorithm for Firewalls,” IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 1, pp. 147-149, 2011.