# Fine Grained Distributed Firewalls Anomaly Detector and Solver With Policy-Based Segmentation Technique

[1]Rajesh B, [2]Kiran Kumar D, [3]Sunil Kumar B, [4]Poorna Chander V
[1]M.Tech Persuing
[2]Assistant Professor in VBIT
[3]Assistant Professor in JNIT
[4]Assistant Professor in Guru Nanak

**Abstract**
To provide the business application services over internet, Firewalls are among the most pervasive network security mechanisms, deployed extensively from the borders of networks to end systems.With the advent of global Internet connection, network security has gained significant attention in research and industrial communities. Day to day increasing threat of various network attacks, firewalls have become important elements for various sizes of applications. But still our business services(web applications) are suffering from unintended security leakages by unauthorized actions and designing and managing Web access control policies are often error-prone due to the lack of effective analysis mechanisms.Firewall policy configuration is an important factor to determine the firewall security efficiency. In this paper, we introduced Distributed Firewall Anomaly Detector and Solver (DFADS) to detect all anomalies that could exist in a single- or multi-firewall environment. We also present some techniques and approaches to automatically discover policy anomalies in centralized and distributed legacy firewalls.DFADS usea policy-based segmentation technique to accurately identify policy anomalies and derive effective anomaly resolutions, along with an intuitive visualization representation of analysis results. Our experiments also supporting that this mechanism is efficiently resolving the policy configuration problems.

**Keywords**
Distributed Firewalls, Firewall Policies, Firewall Anomalies, Network Security, FADS

## I. Introduction

Firewalls are widely deployed in the current Internet as defense mechanisms to block unwanted traffic. With this nature, many factors have led firewall configurations to become increasingly complex, including the introduction of new protocols and the regular discovery of worms. These factors may reflect the increased functionality and utilization of the Internet. As one would expect, the computational requirements for enforcing these policies have risen over the years.

Firewall is a widely deployed mechanism for improving the security of enterprise networks. However, configuring a firewall is daunting and error-prone even for an experienced administrator. As a result, misconfigurations in firewalls are common and serious. Firewalls are the mechanisms that enforce network policies. The complexity of managing firewall policies might limit the effectiveness offirewall security.Firewalls enforce these policies by mediating the communication among hosts in different networks. Upon receiving a packet, a firewall checks the packet's header against a set of user-specified rules (inspection) and forwards/drops the packet if it is desired/undesired (filtering). Through packet inspection and filtering, firewalls can intercept suspicious packets and prevent them from passing through. A firewall can enforce a complete network-wide access policy if all incoming/outgoing packets are configured to pass through the firewall. In a single firewall environment, the localfirewall policy may include intra-firewall anomalies, wherethe same packet may match more than one filtering rule.Moreover, in distributed firewall environments[2,3], firewalls mightalso have inter-firewall anomalies when individual firewalls inthe same path perform different filtering actions on the sametraffic. Therefore, the administrator must give special attentionnot only to all rule relations in the same firewall in orderto determine the correct rule order, but also to all relationsbetween rules in different firewalls in order to determine theproper rule placement in the proper firewall.As the number of filtering rules increases, the difficulty of adding a new rule or modifying an existing one significantly increases.

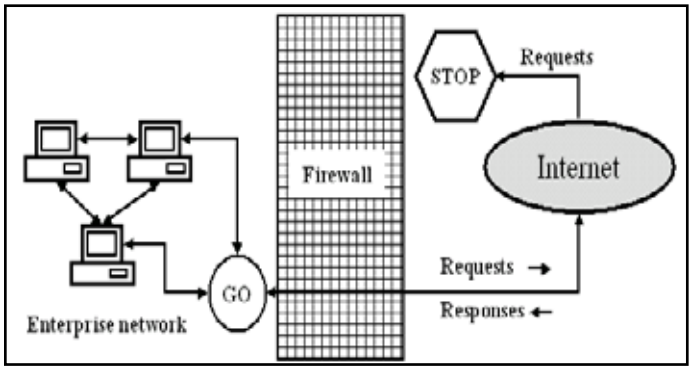


Fig. 1: Distributed Firewall for Enterprise Network

Therefore, the effectiveness of firewall security is dependent on providing policy management techniques and tools that network administrators can use to analyze, purify and verify the correctness of written firewall filtering rules. In this paper, we introduce a policy-based segmentation technique named as Distributed Firewalls (Fig. 1) Anomaly Detector and Solver (DFADS) to detect all anomalies that could exist in a single- or multi-firewall environment, which adopts a binary decision diagram (BDD) based data structure to perform set operations, for policy anomaly discovery and resolution. Based on this technique, an authorization space defined by an XACML policy or policy set component can be divided into a set of disjoint segments. Each segment associated with a unique set of XACML components indicates an overlapping relation among those components. We finally develop a rule editor to produce anomaly-free firewall policies, and greatly simplify adding, removing and modifying filtering rules.Accurate anomaly information is crucial to the successof anomaly resolution. In this paper, we introduced a formal definition of filtering rule relations and then identify all anomalies that might exist in any firewall policy in both centralized and distributed firewall environments. We also use a tree-based filtering representation to develop anomaly discovery algorithms for reporting any intra and inter-firewall anomaly in any general network. In order to evaluate the practicality of our tool, our extensive experiments deal with a set of example firewall policies.

## II. Related Work

### A. Firewalls and Policies

A firewall is a network element that controls the traversal of packets across the boundaries [6] 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 packets that satisfy specific conditions. A rule is composed of set of filtering fields (also called network fields) such as protocol type, source IP address, destination IP address, source port and destination port, as well as an action field. The filtering fields of a rule represent the possible values of the corresponding fields in actual network traffic that matches this rule. Each network field could be a single value or range of values. Filtering actions are either to accept, which permits the packet into or from the secure network, or to deny, which causes the packet to be blocked. The packet is permitted or blocked by a specific rule if the packet header information matches all the network fields of this rule. Otherwise, the following rule is examined and the process is repeated until a matching rule is found or the default policy action is performed.

### B. XCAML

XACML has become the de facto standard for describing access control policies and offers a large set of built-in functions, data types, combining algorithms, and standard profiles for defining application-specific features. At the root of all XACML policies[3] is a policy or a policy set. A policy set is composed of a sequence of policies or other policy sets along with a policy combining al- gorithm and a target. A policy represents a single access control policy expressed through a target, a set of rules and a rule combining algorithm. The target defines a set of subjects, resources and actions the policy or policy set applies to. For an applicable policy or policy set, the corresponding target should be evaluated to be true; otherwise, the policy or policy set is skipped when evaluating an access request. A rule set is a sequence of rules. Each rule consists of a target, a condition, and an effect. The target of a rule decides whether an access request is applicable to the rule and it has a similar structure as the target of a policy or a policy set; the condition is a Boolean expression to specify restrictions on the attributes in the target and refine the applicability of the rule; and the effect is either permit or deny. If an access request satisfies both the target and condition of a rule, the response is sent with the decision specified by the effect element in the rule. Otherwise, the response yields NotApplicable which is typically considered as deny.

### C. XACML Anomalies

XACML policy may contain both policy components and policy set components. Often, a rule anomaly occurs in a policy component, which consists of a sequence of rules. On the other hand, a policy set component consists of a set of policies or other policy sets, thus anomalies may also arise among policies or policy sets. We address XACML policy anomalies at both policy level and policy set level.

### D. Anomalies at Policy Level

A rule is conflicting with other rules, if this rule overlaps with others but defines a different effect.

### E. Anomalies at Policy Set Level

Anomalies may also occur across policies or policy sets in an XACML policy.Most prior anomaly detection work only treat a policy anomaly as an inconsistent or redundant relation between two rules. However, a policy anomaly may involve in multiple rules.

## III. Distributed Firewalls Anomaly Detector and Solver (DFADS)

Our policy anomaly management framework DFADS is composed of two core functionalities: conflict detection and resolution, and redundancy discovery and removal.Both functionalities are based on the rule-based segmentation technique. Modeling of firewall rule relations is necessary for analyzing the firewall policy and designing management techniquessuch as anomaly discovery and policy editing.

### A. Distributed Firewalls Anomaly Detector (DFAD)

First, DFADS parses a firewall configuration into an compact representation based on the operational semantics of a firewall. It is very common to have multiple firewalls installed in the same enterprise network. This has many network administration advantages. It gives local control for each domain according to the domain security requirements and applications. For example, some domains might demand to block RTSP traffic or multicast traffic,however, other domains in the same network might request to receive the same traffic. Multi- firewall installation also provides inter-domain security, and protection from internally generated traffic. Moreover, end users might use firewalls in their personal workstations for other reasons. However, because of the decentralized nature inherent to the security policy in distributed firewalls [4], the potential of anomalies between firewalls significantly increases. Even if every firewall policy in the network does not contain rule anomalies here could be anomalies between policies of different firewalls [8]. For example, an upstream firewall might block a traffic that is permitted by a downstream firewall or vice versa. In the first case, this anomaly is called inter-firewall "shadowing".In the other case, the resulted anomaly iscalled "spurious traffic" because it allows unwanted traffic[8] to cross portions of the network and increases the network vulnerability to denial of service attack.

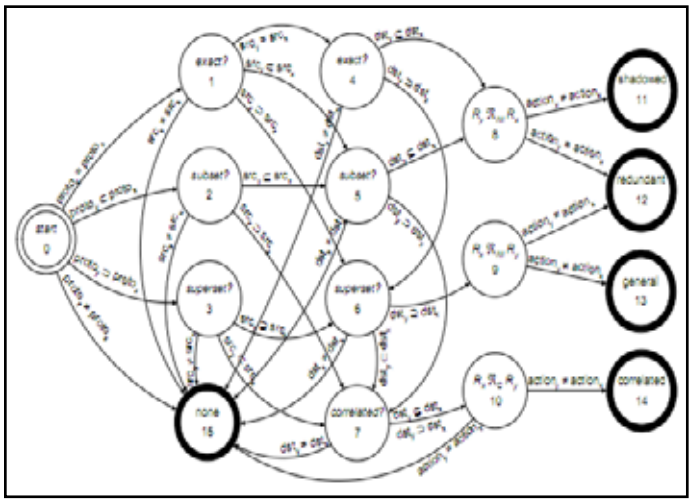


Fig. 2: State Diagram for Anomalies Detection Based on Rules Rx and Ry

From the Above Diagram,Firewall anomaly discovery states for any two rules, Rx and Ry, where the two rules are in the same firewall and Ry follows Rx.For simplicity, the address and port fields are integrated in one field for both the source and destination. This reduces the number of states and simplifies the explanation

of the diagram.Initially no relationship is assumed. Each field in Ry is compared to the corresponding field in Rx starting with the protocol, then source address and port, and finally destination address and port. The relationship between the two rules is determined based on the result of subsequent comparisons. If every field of Ry is a subset or equal to the corresponding field in Rx and both rules have the same action, Ry is redundant to Rx, while if the actions are different, Ry is shadowed by Rx. If every field of Ry is a superset or equal to the corresponding field in Rx and both rules have the same action, Rx is potentially redundant to Ry, while if the actions are different, Ry is a generalization of Rx. If some fields of Rx are subsets or equal to the corresponding fields in Ry, and some fields of Rx are supersets to the corresponding fields in Ry, and their actions are different, then Rx is in correlation with Ry. Identifying irrelevant rules requires the knowledge of the network connectivity. Discovering this intra-firewall anomaly is discussed later in Section V-C along with the inter-firewall anomaly discovery algorithm. If none of the preceding cases occur, then the two rules do not involve any anomalies.The basic idea for discovering anomalies is to determine if any two rules coincide in their policy tree paths. If the path of a rule coincides with the path of another rule, there is a potential anomaly that can be determined based on intra firewall anomaly definitions. If rule paths do not coincide, then these rules are disjoint and they have no anomalies. The detailed description of the intra-firewall anomaly discovery algorithm is available in [1-3]. Our conflict detection mechanism examines conflicts at both policy level and policy set level for XACML policies. In order to precisely identify policy conflicts and facilitate an effective conflict resolution, we present a policy-based segmentation technique to partition the entire authorization space of a policy into disjoint authorization space segments.

### B. Distributed Firewalls Anomaly Solver (DFAS)

Similar to the solution of anomalies detection at policy set level, we handle the redundancy removalfor a policy set based on an XACML tree structure representation. We employ following four steps to identify and eliminate rule redundancies at policy level: authorization space segmentation, property assignment for rule subspaces, rule correlation break, and redundant rule removal. After each component of a policy set PS performs redundancy removal, the authorization space of PS can be then partitioned into disjoint segments by performing Partition() function [9].

### 1. Space Segmentation

We first performthe policy segmentation function Partition_P() defined in [4] to divide the entire authorization space of a policy into disjoint segments. We classify the policy segments in following categories: non-overlapping segment and overlapping segment, which is further divided into conflicting overlapping segment and non-conflicting overlapping segment. Each non-overlapping segment associates with one unique rule and each overlapping segment is related to a set of rules, which may conflict with each other (conflicting overlapping segment) or have the same effect (non-conflicting overlapping segment).

### 2. Fine Grained Conflict Resolver

Our conflict resolution framework introduces an effect constraint that is assigned to each conflicting segment as shown in fig. 3. An effect constraint for a conflicting segment defines a desired response (either permit or deny) that an XACML policy should take when any access request matches the conflicting segment. The effect constraint is derived from the conflict resolution strategy[6] applied to the conflicting segment, using a similar process of determining the effect of a conflicting segment. A policy designer chooses an appropriate conflict resolution strategy for each identified conflict by examining the features of conflicting segment and associated conflicting components. In our conflict resolution framework, a policy designer is able to adopt different strategies to resolve conflicts indicated by different conflicting segments. In addition to four standard XACML [4-5] conflict resolution strategies, user-defined strategies [11], such as Regency-Overrides, Specificity-Overrides and High Majority-Overrides, can be implied in our framework as well. For example, applying a conflict resolution strategy, High-Majority-Overrides, to the second conflicting segment cs2 of policy P1 depicted in Figure 3, an effect constraint Effect = "Deny" will be generated for cs2.

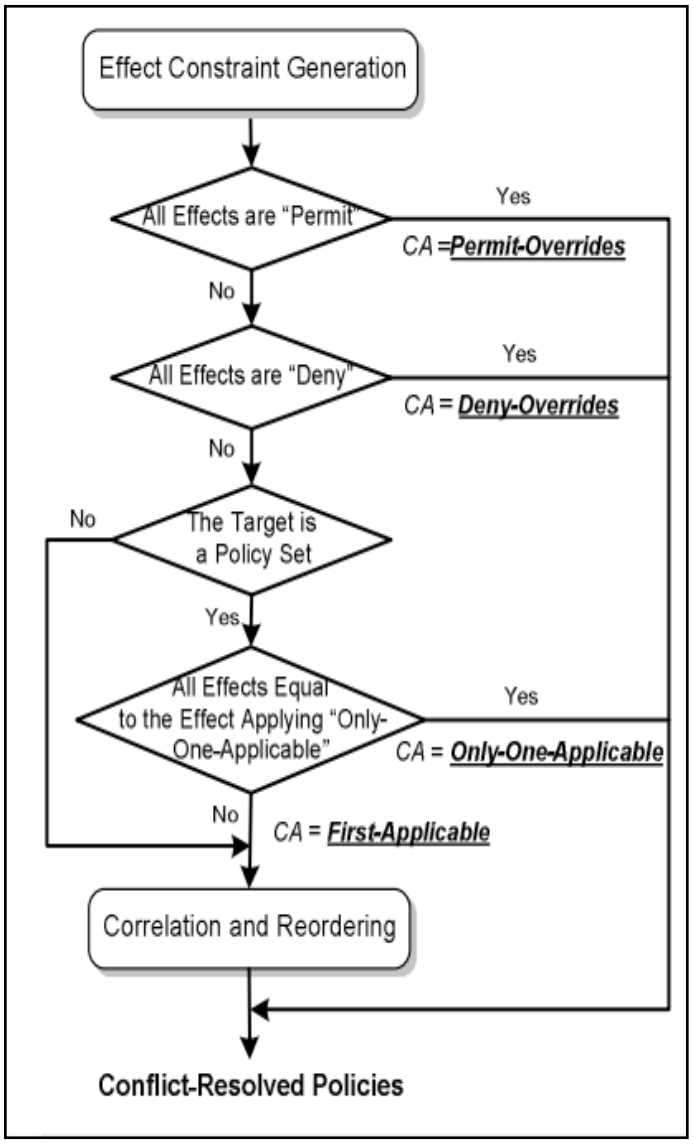


Fig. 3: Fine-Grained Conflict Resolution Framework

### IV. Experiments

We implemented the techniques and procedures described in Sections II and III in a software tool called the "Firewall Policy Advisor" or FPA 1. The tool implements the inter firewall and intra-firewall anomaly discovery approach, as well as the distributed firewall policy editor. The FPA was developed using Java programming language and it includes a graphical user interface.To assess the practical value of our techniques, we first used the FPA tool [12] to analyze real firewall rules in our

university network as well as in some local industrial networks in the area. In many cases, the FPA has shown to be effective by dis- covering many firewall anomalies that were not discovered by human visual inspection. We then attempted to quantitatively evaluate the practical usability of the FPA by conducting a set of experiments that consider the level of network administrator expertise and the frequency of occurrence of each anomaly type. In this experiment, we created two firewall policy exercises and asked twelve network administrators with varying level of expertise in the field to complete each exercise. The exercises include writing filtering rules in centralized and distributed firewalls based on a given security policy requirements. We then used the FPA tool [12] to analyze the rules in the answer of each one and calculated the ratio of each anomaly relative to total number of rules. The average total number of rules was 40 in the centralized firewall, and 90 in the distributed firewall for a network having only three firewalls.

In the last phase of our evaluation study, we conducted number of experiments to measure the performance and the scalability of firewall anomaly discovery under different filtering policies[9] and network topologies. Our experiments were performed on a Pentium PIII 400 MHz processor with 128 MByte of RAM. we installed a random set of filtering rules in each firewall. The generated topology information and the firewall setup of each network are used as inputs for our experiment. We then used the FPA to run the inter-firewall policy analysis algorithm on each network with a different number of rules (10-50 rules) for each firewall. We measured, in each case, the processing time required to produce the final policy analysis report. We noticed that for small and mid-size networks (such as Network 1 that has 8 sub-domains and Network 2 that has 12 sub-domains), the processing time ranges from 3 to 40 seconds. However, in case of large networks (such as Networks 3 and Network 4 that have 18 and 27 sub-domains respectively), the firewall anomaly discovery requires much higher processing time ranging from 11 to 180 seconds depending on the rule complexity.

## V. Conclusion

In the field of firewalls, current research mainlyfocuses on the management of distributed firewall policies. Firewall security, like any other technology, requires proper management in order to provide proper security services. We have proposed an innovative mechanism(Distributed Firewalls Anomaly Detector and Solver (DFADS)) that facilitates systematic detection [10] and resolution of XACML policy anomalies.A policy-based segmentation mechanism and a fine grained Conflict Resolver representation technique were introduced to achieve the goals of effective and efficient anomaly analysis.When an anomaly is detected, users are prompted with proper corrective actions. We intentionally made the tool not to automatically correct the discovered anomaly but rather alarm the user because we believe that the administrator should have the final call on policy changes. Finally, we presented a user-friendly Java based implementation of Firewall Policy Advisor.We believe our systematic mechanism and tool will significantly help policy managers support an assurable Web application management service.Moreover, we would explore how our anomaly analysis mechanism can be applied to other existing access control policy languages.

## Refernces

[1] L. Bauer, S. Garriss, M. Reiter,"Detecting and resolving policymisconfigurations in access-control systems", ACM Transactions onInformation and System Security (TISSEC), 14(1), 2, 2011.

[2] A. Birgisson, M. Dhawan, U. Erlingsson, V. Ganapathy, L. Iftode. "Enforcing authorization policies using transactional memory introspection", In Proceedings of the 15th ACM conferenceon Computer and communications security, pp. 223–234. ACMNew York, NY, USA, 2008.

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

[4] E. Al-Shaer, H. Hamed,"Design and Implementation of Firewall Policy Advisor Tools", DePaul CTI Technical Report, CTI-TR-02-006, August 2002.

[5] Wool,"Trends in Firewall Configuration Errors: Measuring the Holes in Swiss Cheese", IEEE Internet Computing, Vol. 14, No. 4, pp. 58-65, July/Aug. 2010.

[6] J. Alfaro, N. Boulahia-Cuppens, F. Cuppens,"Complete Analysis of Configuration Rules to Guarantee Reliable Network Security Policies", Int'l J. Information Security, Vol. 7, No. 2, pp. 103- 122, 2008.

[7] F. Baboescu, G. Varghese,"Fast and Scalable Conflict Detection for Packet Classifiers", Computer Networks, Vol. 42, No. 6, pp. 717-735, 2003.

[8] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, P. Mohapatra, C. Davis, "Fireman: A Toolkit for Firewall Modeling and Analysis", Proc. IEEE Symp. Security and Privacy, pp. 15, 2006.

[9] "Java BDD", [Online] Available: http://www.javabdd.sourceforge.net, 2012.

[10] "Buddy Version 2.4, [Online] Available: http://www.sourceforge.net/projects/buddy, 2012.

[11] "TENABLE Network Security," [Online] Available: http://www.nessus.org/nessus, 2012.

[12] N. Li, Q. Wang, W. Qardaji, E. Bertino, P. Rao, J. Lobo, D. Lin, "Access Control Policy Combining: Theory Meets Practice", Proc. 14th ACM Symp. Access Control Models and Technologies, pp. 135- 144, 2009.

Rajesh Boyanapalli pursuing M.tech from Nimra Institute of Science and Technology. His areas of interest are Networking, Data Mining and cloud computing.

Dasari.Kiran Kumar M.Tech from ANU Guntur, completed M.C.A from KU, Warangal,. I am presently working as Assistant Professor in Department of Computer Science Engineering in Vignana Bharathi Institute of Technology, Ghatkesar, Aushapur, Hyderabad. I am having 5years of Teaching Experience. My interested subjects are Software Engineering, Data mining, Web services, Web Technologies, Java, C, and C++.

Mr. B. Sunil Kumar Pursing Mtech(CSE) from Vignan Bharathi Institute of Technology affiliated to JNTU-H and completed MCA from Lokamanya Tilak PG College Affiliated to Osmania University. Iam Presently working as Assistant Professor in Department of Computer Science & Engineering in Jawaharlal Nehru Institute of Technology, I am having 3years of Teaching Experience. My interested subjects are Web Technologies, Web services, Mobile computing, cloud computing, Computer Networks, Operating System, Computer Organisation, Java, C, and C++.

V. Poorna Chander working as an Assistant Professor in MCA Department for Guru Nanak Institute of PG Studies, Hyderabad, He had 5 years of teaching Experience. He completed MCA through Kakatiya University, Warangal and his interested subjects are C, C++, Java, Web Technologies, Network and Information Security, Data Mining, Software analysis and Design Engineering.