ISSN:0975-9646

Segment Generation Approach for Firewall Policy Anomaly Resolution

Dr.S.Madhavi, G.Raghu

Department of CSE,
PVP Siddhartha Institute of Technology,
Vijayawada, Krishna Dist, Andhra Pradesh.

***Abstract*— Firewall Policy Anomalies are situations where pre-defined and applied policy settings fail to impose during packet filtrations due to heavy loads experienced by the firewall. Prior approaches to handle these anomalies suffered from rule mismanagement and inaccurate results issues. So, we have used the earlier development of anomaly management framework for firewalls based on a rule-based segmentation technique that facilitates not just accurate anomaly detection but also effective anomaly resolution. A grid based visualization technique is introduced to represent policy anomaly diagnosis information in an intuitive and effective way. As a performance optimization parameter we would like to extend the anomaly management framework with access control policies(ACP). This kind of ACP based approach to the proposed framework turns a security implementation device such as a firewall into a bastion host like machine leading to better management of the host and a practical implementation validates our claim.**

***Keywords*—WirelessNetworks, Packet Classification, Denial-of-Service, Selective Blocking, Access Control Policies.**

I. INTRODUCTION

Firewall Policy Management(FPM) is one of intensive and expensive aspects of managing almost any network infrastructure and requires a high level of expertise by network administrators to get right customized configurations for each organization unique needs. A single glitch in such an FPM and network applications lose communications, transactions are not saved, processed, and application consoles quickly goes out of control. A firewall holds thousands of rules, more complex environments where security is an issue and customization are regular these firewalls may hold rules ten times that many. These firewall management complexities are true across all major systems regardless of major firewall vendors such as Cisco, Juniper, CheckPoint, Fortinet, IBM/ISS Linux, or Nortel. On average it takes about three hours of testing and analysis to implement a single rule change which signifies the magnitude of the management burden. One rule may get involved in multiple policy anomalies. In these situations, this anomaly resolution in isolation may trigger handling delays or the reason behind other anomalies. It is very difficult to deal with all these conflicting rules by only reordering these conflicting rules. Hence, it is necessary to detect the dependency relationships among packet space segments for efficiently resolving policy anomalies.

Each conflicting segment indicates a policy conflict as well as a set of conflicting rules involved in the conflict. Once conflicts are identified the system administrator resolves them manually by changing the conflicting rules which is a tedious task and even impractical due to the complicated nature of policy conflicts. An effective method to resolve a policy conflict is to determine which rule should take precedence when a network packet is matched by a set of rules involved in the conflict automatically without human involvement.

An automated firewall anomaly management framework[5] for firewalls are based on a rule-based packet segmentation technique which is used to facilitate an effective anomaly detection and resolution. Using this technique, a network packet space defined by a firewall policy can be divided into a set of disjoint packet space segments. Each packet segment associates with any of the unique set of firewall rules defined for various protocols accurately indicates the threat packets (either conflicting or redundant) using those rules. It involves a conflict resolution method with the help of several effective resolution strategies for various network protocols with respect to the risk assessment of protected networks and the intention of policy definition. The technique introduces that an action constraint is assigned to each of these fracas segment. An action condition for each conflicting segment defines a desired action (either Allow or Deny) that the firewall policy should take when any packet within the conflicting segment comes to the firewall. To resolve a conflict, the action constraint has to be satisfied by the action taken for each packet within the conflicting segment.

II. RELATED WORK

A firewall policy consists of a sequence of rules that define the actions performed on packets that satisfy certain conditions. The rules are specified in the form of (condition, action). A condition in a rule is composed of a set of fields for identifying the matching packets. Table 1[5] gives an example of a firewall policy, which includes firewall rules ranging from r1 to r5. Note that the symbol "*" utilized in firewall rules denotes a domain range. For instance, a single "*" appearing in the IP address field represents an IP address range from 0.0.0.0 to 255.255.255.255. For demonstrational feasibility consider the following Firewall Policy results table.

TABLE 1
An Example Firewall Policy

Rule	Protocol	Source IP	Source Port	Destination IP	Destination Port	Action
r_1	UDP	10.1.2.*	*	172.32.1.*	53	deny
r_2	UDP	10.1.*.*	*	172.32.1.*	53	deny
r_3	TCP	10.1.*.*	*	192.168.*.*	25	allow
r_4	TCP	10.1.1.*	*	192.168.1.*	25	deny
r_5	*	10.1.1.*	*	*	*	allow

Several related work has categorized different types of firewall policy anomalies [1], [2], [5]. On the basis of following classifications, typically encountered firewall policy anomalies are:

1. Shadowing. A rule can be shadowed by one or a set of preceding rules that match all the packets which also match the rule which is shadowed, while they entirely initiate a different action. In this situation, all the packets that unique rule intends to deny (accept) can be accepted (denied) by the previous one; therefore the shadowed rule will never be effective enough. In Table 1, rule r4 is shadowed by rule r3 because r3 allows every TCP packet coming from any one port of the nodes at 10.1.1.* to the port 25 of the nodes at 192.168.1.*, and the rule r4 denies all the packets if it comes before rule r3.
2. Generalization. A rule is a generalization of one or a set of previous rules if a subset of the packets matched by this rule is also matched by the preceding rule(s) but taking a different action. For example, in Table 1 rule r5 is a generalization of rule r4.The two rules highlight that all the packets coming from 10.1.1.* are allowed, but the TCP packets coming from 10.1.1.* to the port 25 of 192.168.1.* are denied.Generalization might not be an erroneous condition.
3. Correlation. One rule is correlated with many other rules, if a rule converges with others but defines a different action entirely. In this situation, the packets are matched by the intersection of those rules may be accepted by one rule, but denied by other rules. In Table 1,rule r2 correlates rule r5, and all UDP packets arriving from any port of node at 10.1.1.* to the port 53 of node at 172.32.1.* match these rules at the intersection. Since rule rule r5 comes after rule r2,the rule r2 denies every packet within the intersection of these rules. Unless, their positions are swapped, the same packets will be accepted.
4. Redundancy. A rule is redundant if there is another same or more general rule available that has the same effect. For example, in Table 1 rule r1 is redundant with respect to rule r2 specified, since all UDP packets coming from any nodes of port at 10.1.2.* to the port 53 of node at 172.32.1.* matched with r1 can also match r2 as well resulting with the same action twice.

Anomaly detection algorithms and corresponding tools were introduced previously in [1], [2] as well. However, existing conflict classification and detection approaches only treat a policy conflict as an inconsistent relation between one rule and other rules leading to redundant inconsistent results and high processing time which are addressed through our approaches.

Compared to prior approaches specified in [1][2][3] and their prototypes like Firewall Policy Advisor [1] and FIREMAN [2], a more effective redundancy elimination mechanism used in this framework, and through the experimental results, redundancy discovery mechanism achieved approximately 70 percent improvement compared to prior approaches of [1], [2]. Also the outcomes of prior policy analysis tools [2], [1] are list of possible anomalies, which shows a view to the system administrators regarding the origination of policy anomalies. Using the information visualization technique [4] and our rule-based packet segmentation technique they developed a visualization-based firewall anomaly management environment (FAME). A simulation with respect to the real-life firewall policies highlights the efficiency of our system with respect to automated network anomaly conflict resolutions.

III. PROPOSED SCHEME

Prior anomaly detection methods could not accurately point out the anomaly portions caused by a set of overlapping rules causing redundancy and high processing times making them inadequate for high end dynamic network environments. In order to precisely identify policy anomalies and enable a more effective anomaly resolution, an earlier technique which is based on rule-based segmentation, which adopts a binary decision diagram (BDD)-based data structure to represent rules and perform different operations such as set operations and transforms a list of rules into a set of disjoint network packet spaces to resolve overlapping conflicts arising due to similarity of various network protocols and their payloads. The Segment Generation algorithm[5] of network packet which is at the core of this framework is specified here in the form of a conflict reordering flowchart.

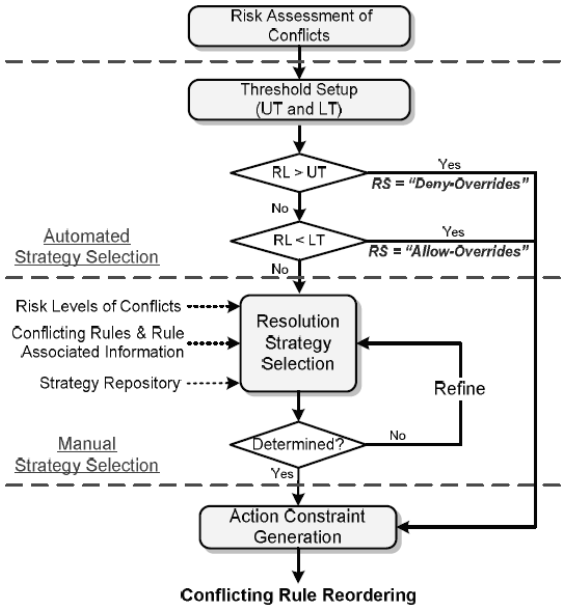


Fig. 1. Strategy-based conflict resolution.

Algorithm 1: Segment Generation for Network Packet Space of a Set of Rule R: Partition(R)

```
   Input: A set of rules, R.
   Output: A set of packet space segments, S.
1  foreach r ∈ R do
2      s_r ⟵ PacketSpace(r);
3      foreach s ∈ S do
4          /* s_r is a subset of s*/
5          if s_r ⊂ s then
6              S.Append(s \ s_r);
7              s ⟵ s_r;
8              Break;
9          /* s_r is a superset of s*/
10         else if s_r ⊃ s then
11             s_r ⟵ s_r \ s;
12         /* s_r partially matches s*/
13         else if s_r ∩ s ≠ ∅ then
14             S.Append(s \ s_r);
15             s ⟵ s_r ∩ s;
16             s_r ⟵ s_r \ s;
17     S.Append(s_r);
18 return S;
```

Algorithm 1[5], shows the pseudo code of generating packet space segments for a set of firewall rules R. This algorithm works by adding a network packet space s derived from a rule r to a packet space set S. A pair of network packet spaces should satisfy one of the following relations: subset (line 5), superset (line 10), partial match (line 13), or disjoint (line 17). Therefore [5], one can utilize set operations to separate the overlapped spaces into disjoint spaces to classify whether the packet can be allowed or not. The strategy based conflict resolution[5], which is adapted in FAME is represented by the event flows.

Access control policy signifies a framework that represents authorizations, actions, and their effect in a networked system. Access control systems can be changed by a policy, which is having a set of objects and the corresponding substitutions. We define $\sum$ as a finite set of those objects such that each object in that $\sum$ has a type. $\sum_t \subseteq \sum$ is the set of objects of that type t. If V is the set of variables that are acted upon an action event, then a substitution σ is a function V → $\sum$ that respects types. The set of atomic propositions P is defined as the set of predicates instantiated with the objects in $\sum$ thus

P = {w(v) σ | w Є Pred, v Є V* and σ is a substitution}

The system state is an evaluation of atomic propositions defined in in P. A state s can be defined as a function of P influencing the outcome of events. We use s[p → m] to denote the state that is like s except that it maps the event proposition p to value m.

A variety of access control policies are implemented :

For start and stop updating the packets in the table we will use “Start”, “Stop” options.

a. Lookup(ipAddress, hostname)

```
//Finds the name of the ipAddress that is present in the table
    1. if(length of ipAddress >0)
    2. get the host name;   //gets the name of ipaddress
    3. retrieves by finding the suspicious host;
```

b. Find Local Process(port, processid)

```
// finds which process id uses which port to receive / send  packets
    1. if(length of the port>0){
    2. args[]={“netstat -aon| find \"" + port + "\"""}
    3. if(file does not exist)
    4. Create a new file;
    5. }
```

c. Service Control(action, service name)

```
   //to view /start /stop any service
    1. if(opt==1 || opt==0)
    2. action=(opt==1? “start” : “stop”);  // 1/0 - start/stop the
           service
    3. else
    4. print statement;
    5. if(length of service name>0){
    6. Request to create a new file;
    7. if(file does not exists)
    8. create new file;
    9. }
```

d. Block access(hostname, host)

```
  Host file is the file to be blocked
    1. if(host file does not exists)
    2. create a new host file;
    3. if(hostname==null)
    4. return;
    5. temphosts=gets path of the temphost;
    6. if(temphosts exists( )){
    7. temphosts.delete( );
    8. create a new temphost file;
    9. }
    10. if(file &&hosts exists){
    11. read the host file;
    12. boolean done=false;
    13. while(line!=null){
    14. if(line.contains(”localhost”)&&!line.startsWith(“#”)&
           &!done){
    15. done=true;
    16. add the hostname to hosts;
    17. }
    18. else
    19. write the line in output;
    20. }
    21. enter the hostname to hosts;
    22. }
    23. else
    24. print there is an error;
```

e. Allow access(hostname, temphost, host)

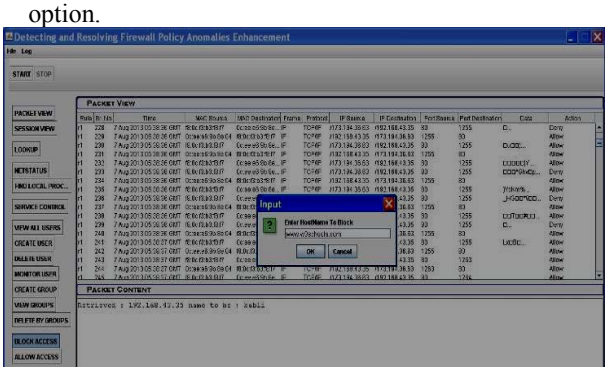
```
//the website has been given access, which is blocked
    1. get the file of blocked host
    2. if(file exists){
    3. read the file ;
    4. boolean done=false;
    5. while(!line=null){
    6. append this line to web access details;
    7. }
    8.   }
    9. if(hostname==null)
    10. return;
    11. if(host file exists){
    12. boolean done=false;
    13. while(line!=null){
    14. if(line equals hostname)
    15. output in new line;
    16. else
    17. write in a new line;
    18. }
```

```
19. delete the host file;
20. rename hostfile to thostfile;
21. delete thostfile;
22. }
23. host=get path of hosts;
24. temphost=get path of temphost;
25. if(temphosts exists){
26. delete the temphost;
27. append newfile to temphost;
28. }
29. if(file and host exists){
30. boolean done=false;
31. while(line!=null){
32. if(line contains only local hostname)
33. done=true;
34. output in new line;
35. else
36. write in the new line;
37. }
38. delete the host;
39. rename temphost to host;
40. }
41. else
42. print there is an error in the output;
43. }
```

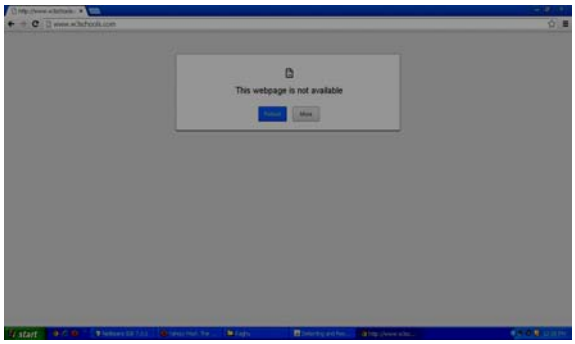
The following are the screens for blocking a website and giving access to the blocked website.

a. Initially, we take "www.w3schools.com" site and we will block the site.

b. Now, we will block the site by using ACP- Block Access option.



c. From the below screen, we can see that the website is blocked.



d. Now, we unblock the site by using ACP-Allow Access.

e. Now, we can see from the following screen that the website blocked is given access.

The class diagram describes the attributes and operations performed on the attributes. The system interacts with login_panel and it consists of anomaly detector, user_details, traffic status. Inturn they interacts with alerts.

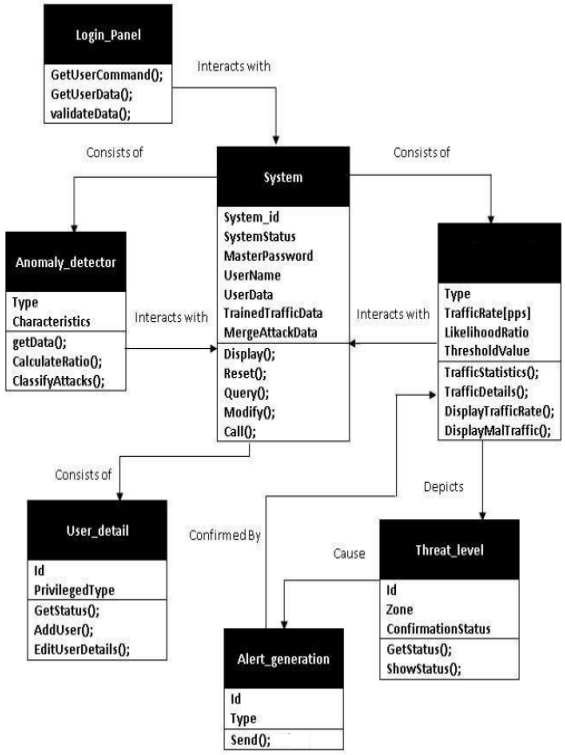


Fig 2: class diagram

The sequence diagram is used to show the interactions between objects in the sequential order.The sequence of actions performed are identification, implementation, classification and also has visualization tool.

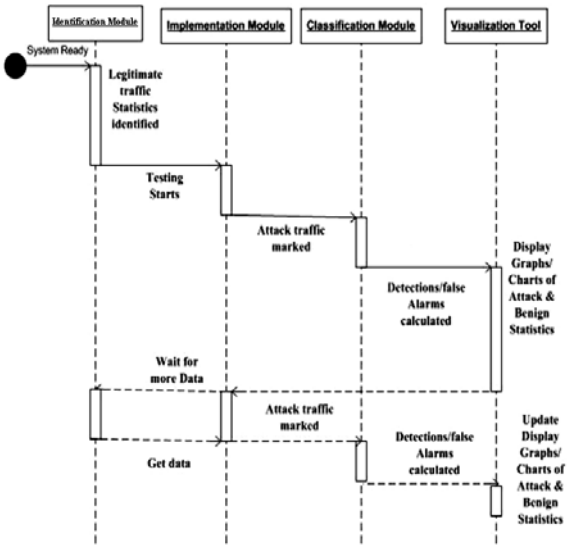


Fig 3: Sequence diagram

Activity diagram represents business process and also graphical representation for executed set of system activities.

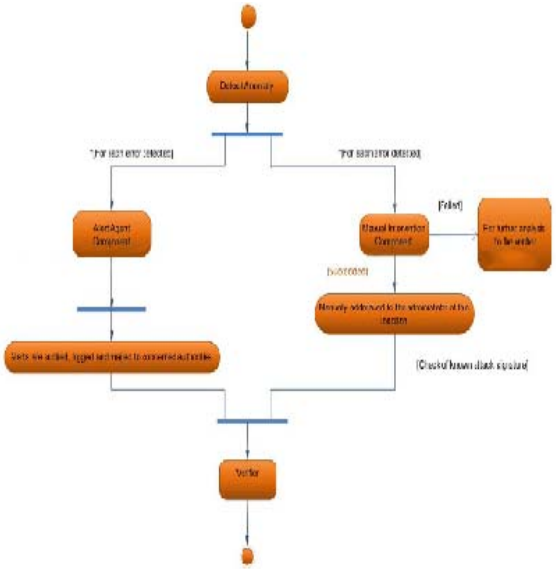


Fig 4: Activity diagram

IV. PERFORMANCE

Firewall Policy rules provides network traffic access control because they define which packets are permitted and which are denied. A firewall access policy(FPA) consists of a set of rules. Each packet is analyzed and its elements compared against elements in the rules of the policy in a sequential order. The rule that matches first, the packet will have its configured action initiated, and any processing specified in the rule's configured options will be implemented. The conflict resolution method that understands several risk assessment strategies deployed in protected networks and the intention of policy definitions is at the core of our framework. Besides monitoring and resolving anomalies using this optimization parameter the system administrators can control the service, user operations and manage processes. The visual statistics of the operations are represented here.

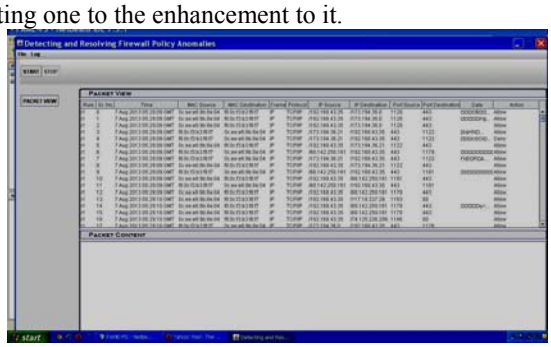
	Source	Destination	Service	Interface	Action
0	firewall, net-192.168.1.0	Any	Any	outside	Deny
1	Any	Any	Any	loopback	Accept
2	net-192.168.1.0	firewall	ssh	All	Accept
3	firewall	net-192.168.1.0	DNS	All	Accept
4	Any	firewall	Any	All	Deny
5	net-192.168.1.0	Any	Any	All	Accept
6	Any	Any	Any	All	Deny

Fig. 5. FAME Network Statistics.

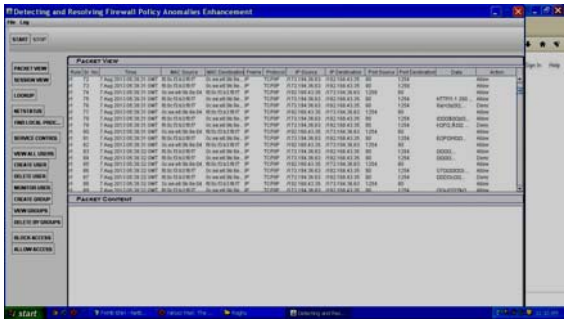
These results signify the packets denied capability of FAME framework that effectively identifies and blocks almost immediately redundant and in cohesive protocol-packet pairs thus justifying 70% duplicate elimination claim. Furthermore implementation of user access control policies of a network host specified earlier in Firewall rule priority(point 5) transforms a normal host to a bastion host like environments

which can be used in high end systems like servers. This transformation yields benefits like batch user manipulations, custom remote host blockages of outbound traffic etc. Also the grid based visual representation further aids network administrators by providing information in an intuitive way, enabling an efficient automated firewall policy anomaly management.

As a performance optimization, we are integrating the access control policies(ACP) to the existing firewall. By observing the following screens we can see the difference between the existing one to the enhancement to it.



From the following screen, we can see the ACP are integrated in the left side of the screen.By comparing these two outputs,our enhancement can give the administrator a better control over the user actions and also controls the service operations.



V. CONCLUSION

In this paper, an automated firewall policy anomaly management environment(FAME) framework is used, that can perform systematic detection and resolution of firewall policy anomalies arised and experienced during high network traffic scenarios. FAME's Rule-based segmentation mechanism and a visual grid-based representation technique achieves the goal of effective and efficient anomaly analysis and the results validates our claim. We implemented user access control policies of a network host which transforms a normal host to a bastion host like environments which can be used in high end systems such as servers. FAME results suggest that it is a practical and helpful system for system administrators to ensure a secured network environment. Although it can also be integrated into Intrusion Detection Systems, Centralized rule management schemes[6] can be regarded as a future research that has the potential to aid high end systems like Servers.

REFERENCES

[1]"Discovery of Policy Anomalies in Distributed Firewalls," IEEE INFOCOM, E. Al-Shaer and H. Hamed,'04, vol. 4, pp. 2605-2616, 2004.

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

[3] E. Lupu and M. Sloman, "Conflicts in Policy-Based Distributed Systems Management," IEEE Transactions. Software Eng., vol. 25, no. 6, pp. 852-869, Nov./Dec. 1999.

[4]"Graph Visualization and Navigation in Information Visualization: A Survey," I. Herman, G. Melanc¸on, and M. Marshall , IEEE Trans. Visualization and Computer Graphics, vol. 6, no. 1, pp. 24-43, Jan.-Mar. 2000.

[5] Hongxin Hu, Gail-Joon Ahn, and Ketan Kulkarni, "Detecting and Resolving Firewall Policy Anomalies" IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 3, MAY/JUNE 2012

[6]CentralizedRuleManagement, http://www.scmagazine.com/strategic-firewall-policy-management/article/119407/#