Implementation and Realization of Network Security Policy Based on Rule Engine

Chenghua Tang
College of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, P.R. of China
Email: tch@guet.edu.cn

Yi Xie
Department of Electrical and Communication Engineering, Sun Yat-Sen University, Guangzhou, P. R. of China
Email: xieyi5@mail.sysu.edu.cn

Abstract—**In order to solve the implementation and realization efficiency problem of the network information system security policy, an improved object-oriented Rete algorithm and its network structure model are proposed, and on this basis, the rule engine is introduced, where the implementation and realization steps and efficiency analysis are given. Result shows that the algorithm and network structure can effectively improve the efficiency of system enforcement and realization. The technology can be adapted to establishing and controlling the policy service in the extensive network environment.**

Index Terms—**Rete, production system, security policy, rule engine**

I. Introduction

The network and information system security policy knowledge is based on the production system, which is used in expert systems, intelligent simulation and many other aspects. As the application continues to expand, increasing its scale, making it more and more problems to run efficiently, in a large-scale cross-platform network security management system applications, a large policy knowledge base matching and reasoning efficiency also become a serious problem. On the other hand, facing the changing business rules in external conditions, more and more complex enterprise-level policy management system urgently requires the system security decision logic to separate from the application logic, and the administrators can manage policy system security in the runtime, without affecting the system's own abstract data types and technology algorithms, therefore, it is necessary and urgent to research such a policy dealing model and mechanism.

The implementation efficiency of production system has been the focus of expert system designers, but also becoming more and more obstacle to promote and popularize the application of production system [1], and the matching efficiency is production system core of the problem. The study found that in the production system, 90% of the time is spent on pattern-matching operation that is a bottleneck of the system running [2-4]. Therefore, the study of practical and efficient pattern matching algorithm, improving the efficiency of production system, has an important significance. The early production system generally use the index count matching method [5], and later found that the use of the knowledge about matching the two sides to guide the matching process, will effectively improve the pattern matching cost [6]. Rete [2] algorithm is classic and the winners in this regard, since the algorithm was presented, based on the Rete algorithm to improve the various programs are also an endless stream [7-8].

In this paper, an improved object-oriented Rete algorithm and its network structure model are proposed based on the traditional Rete algorithm, and on this basis, the rule engine is introduced, on which the network security policy implementation and realization mechanism is proposed. Their performances are analyzed and verified to solve the knowledge match efficiency in a large network information system, and the separate technology of decision-making logic and technical decision-making logic based on security policy is given.

II. Related Definitions

With the development of information safety, varieties of security policies have been appearing, such as BLP, Biba, MAC, DAC, RBAC and Chinese Wall, mostly put forward in answer to how to control the visit to network resources or a particular security demand. Security policy is the core of the whole system already, whose correctness determines the system security directly. This essay reconsiders the general definition of security policy. Security policy is a guide to the network security. In terms of its content, it is a group of restrictions, which administer and control the access to network resources. In terms of its characteristics, security policy is an illustrative, permanent criterion made to regulate behavioral choice in a computer system according to management objectives and security demands.

Definition 1. Security policy P is a set of rules which control the management of man and resources. These rules are the provisions description of what allowed and what prohibition under what circumstance. The policy P can be defined by the quaternion $<D, C, A, G>$. Here D is the security domain identifier of the policy P. C is the triggering of policy which relates to the network behavior events. *A* is the action of the executer, and *G* is the policy guideline. An existent function as Equation: $f_A : D \times C \times G \rightarrow A$.

Security policy is a restrictive description on the security domain entities. Its trigger conditions and actions constitute a formal rule-based policy.

Definition 2. Trigger is used to trigger the rules of conditions, including stimulating condition *cd* and constraint condition *cs*, which describes the stimulating condition by a group of event descriptors, and the constraint condition in the form of a predicate. Only when all the conditions are meeting, policy rules will be triggered.

Definition 3. Executer implements the normal-action *n* when the policy is triggered, and executes the remedial action *e* when there is exception.

Definition 4. Security domain entities $E = < S, O >$, which *S* is said that the information subject (*infoSubject*) collection in the security domain, is the entities can take the initiative to visit the resources, including the man, procedures, hosts, security equipment, etc., and *O* is said that the information security domain object (*infoObject*) collection, which can provide information services , or passively accept the visit request, including systems, files, hosts, equipment, etc., existing: $S \cap O \neq \phi$.

The security policy guidelines (*Guideline*) are the guiding principles of demand on security activities. It can use natural language stratification expression.

According to the above definitions and analysis, the security policy conceptual model can be drawn, as is shown in Fig. 1.

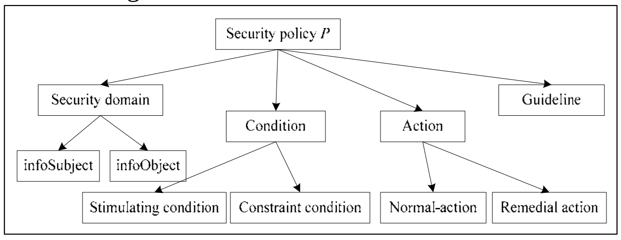


Figure 1. Rete network structure

III. Improved Rete Algorithm

A. Traditional Rete Algorithm

In the reasoning process of rule-based expert system, the matching process is repeated and a dynamic process of change. In each cycle, the rule set and conflict set must be maintained and updated. Rete algorithm is took advantage of the temporal redundancy and structural similarity in the process of rule-based expert system reasoning: (1) In each implementation cycle, a rule's action is only change a very small amount of facts in work memory, therefore, the rules subject to the impact of work memory change are only a small proportion. (2) Many of the rules often contain a similar model structure and model groups. Therefore, Rete preserves the rule status of each cycle of the pattern matching, only considering what the impact of the changed fact on the matching process, which is, using space in return for the implementation efficiency of the system. The core of Rete algorithm is to build Rete matching network structure, which consists of two parts of the pattern network and connection network, as is shown in Fig. 2.

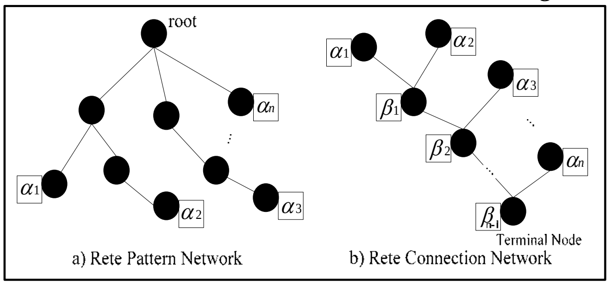


Figure 2. Rete network structure

B. Improved Rete Algorithm

Classic Rete algorithm is used the fact driving (using the fact to drive search rules) approach, thus, the status and numbers of tokens reflecting the fact changes, and the Rete own structural characteristics are important factors affecting the efficiency. Described as follows:

1) Token generation Spending a high time cost: Regardless of pattern network, or connection network, when nodes generate a new token (positive token), the token stack is responsible for the token and control right to the next node, and modify the state stack information. Because each produces a token to carry out a token stack operation and at least one state stack operation, therefore, a large number of tokens will require a large number of stack operations.

2) Deleting the token is also expensive: It needs a lot of unnecessary quires and consistency detection operation. From the root node of pattern network, beginning with the breadth-first search algorithm to find a matching pattern, when a leaf node of pattern network receives a negative token, the first in the β register to find the corresponding token, detect its consistency constraints with the token in α register, and with token that pass the tests comined into a new ngative token, then back to deliver, until reach the end node. Obviously, the constraint detecing in this precess is unnecessary, and the retaining the intermediate results of negative token becomes redundant at this point that the heavier the burden on β register, while a large number of β registers can easily result in combinatiorial explosion.

3) Patterns connection order also has an impact: Rete uses a fixed pattern connect order, which greatly affected the efficiency of the connection network [9].

Then ideas for improvement are as follows:

1) Positive token is being passed directly from the node to the subsequent β register. State stack works only when the node search and pattern connection order control in the pattern network. Moreover, in order to be used for expressing the relationship between policy rules classes, the pattern network is extended to "multi-input" structure.

2) In the connection network node, the negative tokens in β register and conflict set are directly deleted. There are no consistency constraint detection and generation portfolio tokens in β register, but the negative tokens are directly into the next connection node. The connection network nodes are only responsible for consistency constraint detection. In order to reduce connection network spatial redundancy, the attribute of pattern network node is extended to "multi-output", and different policy rules can share the same connection node.

3) Rete in real terms can be summarized as first select the connection after the query optimization theory in relational database system. If reach pattern act as a virtual relationship, then the pattern matching is corresponded to the selection operation, while the connection matching corresponds to the multi-link relationship in the database. Therefore, there are the following principle of pattern connection order: Non-variable pattern on the front, Constraints on the other patterns on the front, and When the variable binding phase is the same, the pattern which matching number is smaller on the front of the other patterns.

IV. Improved Rete Network Structure

In order to be able to adapt policy model based on rule engine, integrating the object-oriented paradigm and rule-based paradigm, packaging the Rete network node components into a java class, the system can create instance of the class to achieve the Rete network building, therefore, the original structure of Rete network is expanded. Fig. 3 is an extension describes of the Rete class structure and its relations in UML.

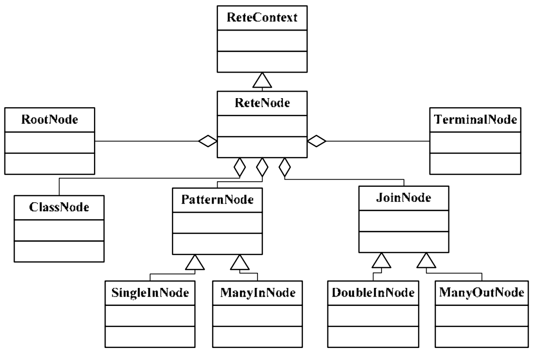


Figure 3. Extended Rete class structure

ReteContext class is the base class of the Rete network structure, but also the parent class of all the others. As the information generated in the context of policy rules, ReteContext class includes general information after its instance, and is used to achieve the initialization of the Rete Network structure.

ReteNode class inherits ReteContext class, with polymerization of the RootNode, ClassNode, PatternNode, JoinNode and TerminalNode class, constituting the basic components of Rete network structure.

RootNode class is the root node in Rete network, which is the first node that the fact token triggered the policy rule flowed into, and is responsible for sending the token into ClassNode.

ClassNode class is used to distinguish different types of objects, which receives only consistent with the types of tokens.

PatternNode class is used to achieve pattern matching, will compare the received token with its own. If the match, the token will continue to be passed to the next node. PatternNode class contains SingleInNode and ManyInNode two kinds of generalization classes; therefore, different patterns can share the same superior class, this time to form a multi-mode balanced search tree (B-Tree) structure.

JoinNode class is used to connect the pattern node and connection node, and test the constraint relations between the tokens that flowed through it. Passed the constraint test tokens can be combined or directly sent to subordinate node. JoinNode class contains DoubleInNode and ManyOutNode two kinds of generalization classed, and different policy rules can share the same JoinNode, so the spatial redundancy is reduced.

TerminalNode class receives the final token, corresponds to a policy rule. TerminalNode generates the corresponding rule instance according to the token, and sends it into Agenda.

Fig.4 shows the improved object-oriented Rete network structure based on the extended Rete class structure, reflecting the flows of tokens in Rete network of two rules. Construction of the network, as well as the optimization of positive and negative tokens, is using the improved methods in the previous section.

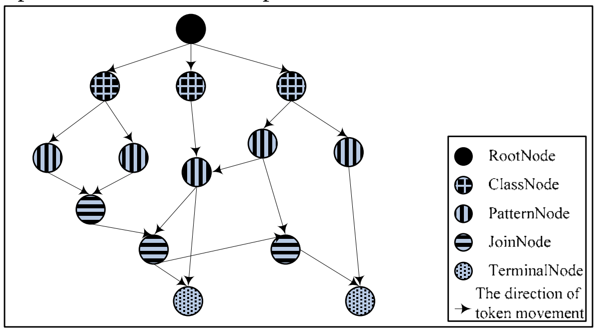


Figure 4. Improved object-oriented Rete network structure

V. Policy Implementation and Realization

A. Security Policy Rule and Rule Engine

According to the definition of security policy, a policy rule is composed of a group of conditions and operations implemented in these conditions, which is expressed as business logic in the application procedures. Policy rules usually by security analysts and system managers to develop and change, but some complex policy rules may be customized by technicians with object-oriented technology language or script [10].

Rule engine is developed from the rule-based expert system reasoning engine, and it is a module embedded in the application. It will achieve the policy decision-making from the application code separated, compile the policy rules using scheduled semantic module to accept data and explain the rules, then make decision in accordance with the rules. The policy rules are the real security service, and their management has nothing to do with their realization.

B. Implementation of Policy based on Rule Engine

Security policy rules can be stored in the relational database or LDAP database in the form of data tables or files. They are security decision-making logic of enterprise managers, which are independent of technical decision-making logic of application developers. Only in designated rules document, the rules could be loaded in application.

Rule engine includes working memory, pattern matcher, agenda, execution engine and rules conflicts processors, as is shown in Fig.5.

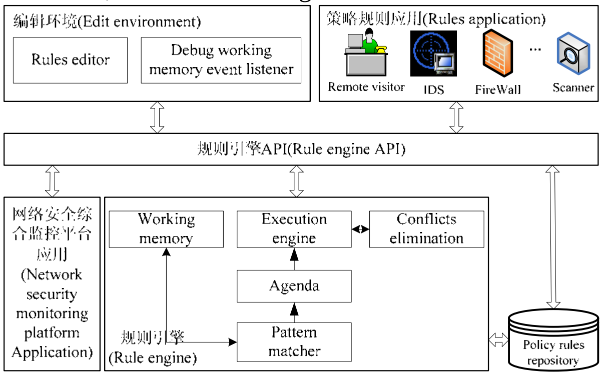

编辑环境(Edit environment)
Rules editor
Debug working memory event listener
策略规则应用(Rules application)
Remote visitor
IDS
FireWall
...
Scanner
规则引擎API(Rule engine API)
网络安全综合监控平台应用 (Network security monitoring platform Application)
Working memory
Execution engine
Conflicts elimination
Agenda
Pattern matcher
规则引擎 (Rule engine)
Policy rules repository

Figure 5. Security policy processing model based on rule engine

Rule engine inferring steps are as follows:

1) Sending the facts objects into working memory.

2) Pattern matcher comparing the facts data with rules in the policy rules repository.

3) Putting the enabled (eligible) rules into agenda orderly.

4) Executing the rules in agenda, detecting and eliminating the potential rules conflicts. Repeating steps 2- 4 above, until all the rules implementing end.

In the agenda of policy rules, potential conflicts may arise, then the specific eliminated methods is need. Aim at the conflicts elimination of policies, jonathan discussed different types policy conflicts in distributed system management [11], Lupu designed four appointed policy implement priority solutions: contrarian policy first, local policy first, near policy first and specified policy first [12]. These solutions can only solve some problems partly.

Modifying the policy conditions, actions and some other properties is the simplest method to avoid the policy conflicts, but this method should be used in designing policies period, namely we must detect it in advance to avoid it. However, it is very difficult in practice. When the conflicts appear, we must apply special method to solve it. According to the policy control rules of production system, we can apply elimination rule in the following:

1) Matching First: choosing the policy which is meet first.

2) Priority First: designing priority for different solutions. There are five solutions at least: contrarian policy first, local policy first, near policy first, newer policy first and specified policy first.

3) Priority-Match First: choosing matching first rule if there is not only one policy based on Priority First rule.

4) Multiple Constraints First: choosing the policy which has the constraints most.

5) "Arbitration": confirming the policy by additional conditions.

In accordance with the priorities, the rules instances will be executed orderly. The process may change working memory data, thus enable certain rules ineffective in agenda as conditions change, thereby be withdrew from the agenda. On the other hand, it may activate the rules which do not satisfy the conditions originally and generate new instances for agenda. Thus a dynamic execution chain of rules come forth, and such rules chain reaction is driven by the data in the working memory entirely.

The conditions matching efficiency of the rules ensure the engine performance. The engine needs to test the data target in the working memory and rules repository rapidly, find the eligible rules from the rule set, and generate the rules instances. Here we will use the improved Rete algorithm, which is a good solution to this problem.

C. Realization of Security Policy based on Rule Engine

An open policy rule engine can be "embedded" in applications to any location. Rule engines of different location can also combine with different sets of rules for handling different data objects. For the sake of concision and clarity, the JAVA language is used to describe how the rule engine realizes the access control policy, whose core is JBoss Rules[13].

Security policy rules can be achieved in several levels, which are, in time sequencing, divided into four steps, as is presented in Fig.6.

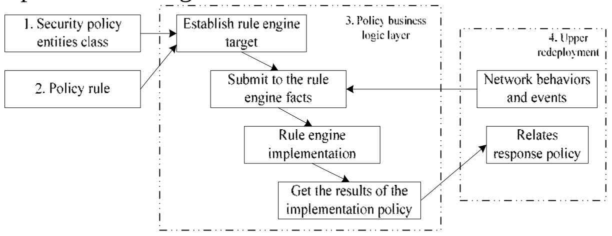

1. Security policy entities class
2. Policy rule
Establish rule engine target
3. Policy business logic layer
Submit to the rule engine facts
Rule engine implementation
Get the results of the implementation policy
4. Upper redeployment
Network behaviors and events
Relates response policy

Figure 6. Policy achieve flow using rule engine

In order to test the improved Rete algorithm, network structure and rule engine, we will take the IP blocking control policy for example.

1) Establishing policy entities class: A more complex IP blocking control policy entity class consists of five members variables: certificateState, linkedTimes, lockState, lockedTime and linkAction. Here certificateState expresses whether the main body state which request to connect is legitimate, linkedTimes is the request number of illegal connections, lockState expresses the current lock status, lockedTime is that has locked the time, and linkAction here describes. The content of the entity class is shown in Fig.7.

```
package accessControlPolicy;
public class IpLinkAccessControl {
        public final static String PERMIT="PERMIT";
        public final static String REJECT="REJECT";

        private boolean certificateState = false;
        private int linkedTimes = 0;
        private boolean lockState = false;
        private int lockedTime = 0;
        private String linkAction=null;
        …
}
```

Figure 7. IP blocking control policy entity class file

2) Editing policy rule file: Fig.8 shows the policy rules file of IP blocking control.

```
package IpLinkAccessControl;
import accessControlPolicy.IpLinkAccessControl;

rule "REJECT LINK AND LOCK, 拒绝并锁定"
    salience 3
    when
     i:IpLinkAccessControl(certificateState==false,lockState==false,linkedTimes>=3)
    then
      System.out.println("执行规则名REJECT LINK AND LOCK, 拒绝并锁定");
      i.setLinkAction(IpLinkAccessControl.REJECT);
      i.setLockState(true);
      i.setLockedTime(4);
End

rule "REJECT LINK, 仅拒绝"
    salience 2  //规则的执行优先级值越大级别越高
    when
     i:IpLinkAccessControl(certificateState==false,lockState==false,linkedTimes<3)
         ||IpLinkAccessControl(certificateState==true,lockedTime<4,lockState==true)
    then
      System.out.println("执行规则名REJECT LINK, 仅拒绝");
      i.setLinkAction(IpLinkAccessControl.REJECT);
      i.setLinkedTimes(i.getLinkedTimes()+1);
end

rule "PERMIT LINK, 仅通过"
    salience 1
    when
      i:IpLinkAccessControl(certificateState==true,lockState==false)
    then
      System.out.println("执行规则名PERMIT LINK, 仅通过");
      i.setLinkAction(IpLinkAccessControl.PERMIT);
      i.setLinkedTimes(0);
end

rule "PERMIT LINK AND UNLOCK, 通过并解锁"
    salience -1  //1-为最低的规则执行优先级
    when
      i:IpLinkAccessControl(certificateState==true,lockedTime>=4,lockState==true)
    then
      System.out.println("执行规则名PERMIT LINK AND UNLOCK, 通过并解锁");
      i.setLinkAction(IpLinkAccessControl.PERMIT);
      i.setLockState(false);
      i.setLinkedTimes(0);
end
```

Figure 8. IP blocking control policy rules file

The policy rule actually described a more complex control of the connection request algorithm:

1. For connecting more than three times the illegal request, directly refuses it and sets the lock status.

2. Under the premise is not locked in the illegal connections, refuses the request and records the number of connection request.

3. Under the premise is already locked and no auto-unlock time, also refuses the legitimate connection request.

4. Under the premise is not locked in, allows the legitimate connection request, and removes it request times.

5. When the locked state has been reached under the premise of unlocking time, allows legitimate connection request, automatically unlocks it, and cleares the number of connection requests.

3) Building policy business logic layer file: The first step is to establish rule engine target and get a new "Workingmemory". The next step is to load facts into Workingmemory. Then the rule engine in accordance with the improved object-oritented Rete network search algorithm to select and implement appropriate policy rules. The results of the execution will be acquired by the upper layer. As is shown in Fig.9.

```
public class IpLinkAccessControlBusiness {
    private static final String RULE_FILE = "IpLinkAccessControl.drl";

    public static void evaluateIpLinkAccessControl(
                IpLinkAccessControl ipLinkAccessControl) throws Exception {
          try{
               final RuleBase ruleBase = readRule();
                  //建立引擎对象并加载策略规则
               final WorkingMemory workingMemory = ruleBase.newWorkingMemory();
                  //为每次策略规则的评估构造新的工作区内存
               workingMemory.assertObject(ipLinkAccessControl);
                  //提交事实数据到工作区内存
               workingMemory.fireAllRules();
                  //引擎执行，评估或更新规则
               List<AclEntry> acls = workingMemory.getObjects(AclEntry.class);
          }catch ( final Throwable t ) {
              t.printStackTrace();
          }
    }

    private static RuleBase readRule() throws Exception {
        final Reader source = new InputStreamReader(
                  IpLinkAccessControl.class.getResourceAsStream(RULE_FILE));
        final PackageBuilder builder = new PackageBuilder();
        builder.addPackageFromDrl( source );
        final Package pkg = builder.getPackage();
        final RuleBase ruleBase = RuleBaseFactory.newRuleBase();
        ruleBase.addPackage( pkg );
        return ruleBase;
    }
}
```

Figure 9. IP blocking control policy business layer file

4) Upper redeployment: The senior network behaviors and events act as facts to trigger rule engine and get the results of implementation policy. Fig.10 shows the trigger mode in a TestCase.

```
IpLinkAccessControl ipLinkAccessControl= new IpLinkAccessControl();
ipLinkAccessControl.setCertificateState(***);
   //这里“***”是当前连接尝试的身份是否合法，值为false或true
ipLinkAccessControl.setLinkedTimes(###);
   //这里“###”是当前连接尝试次数，是一个网络行为事件（事实）
ipLinkAccessControl.setLockedTime($$$);
   //这里“$$$”是当前主体IP已锁定时间，单位为分钟
ipLinkAccessControl.setLockState(&&&);
   //这里“&&&”是当前主体IP的锁定状态，值为false或true

public void testIpLinkAccessControl()throws Exception{
  IpLinkAccessControlBusiness.evaluateIpLinkAccessControl(ipLinkAccessControl);
     //提交将要处理的数据对象
  System.out.println("结果:"+ipLinkAccessControl.getLinkAction());
}
```

Figure 10. Facts trigger the rule engine

D. Application in the Network Security Monitoring Platform

According to the security policy processing model based on rule engine, to establish a security network information system, first under the various entities within the system security requirements, determine the division of security domains, to formulate various security policies, and establish a unified security policy rules database.

Model can be used in the realization of integrated Network Security Monitoring Platform (NSMP). Fig 11 shows the integrated network security monitoring platform architecture is divided into two tiers: Web service layer and basis service layer. Web layer provides access to the system of web services for system content and respond to user input commands, including security monitoring, security policy, integrated audit, systems management and the system announcement; basis service layer provides the background service of the system. it primarily used for equipment communication, lines of communication and other system services (such as topology search, timing services, etc), and from a functional point of view, the basis service layer is mainly includes communication module, the logical address interface module and database read/write module.

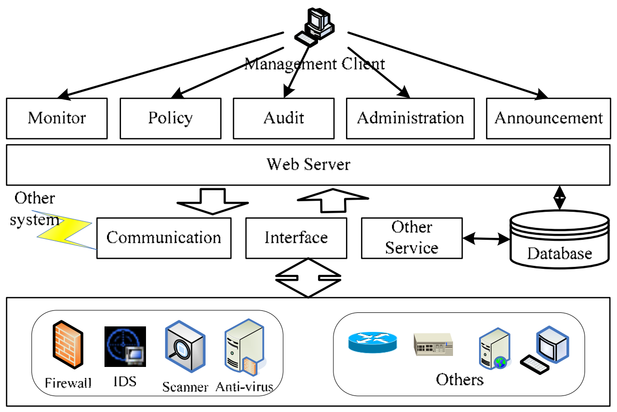
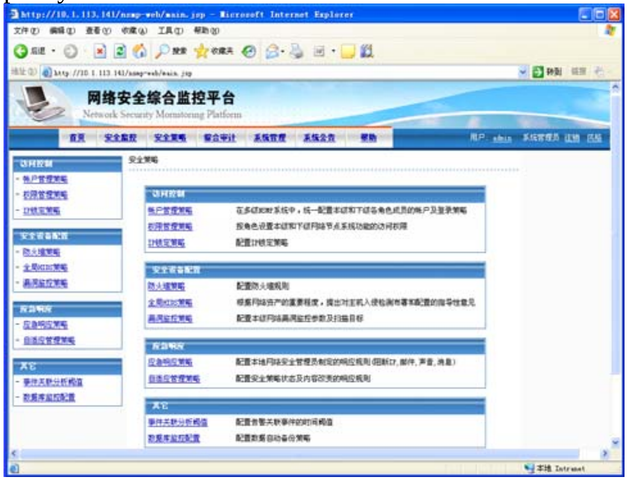


Figure 11. Network Security Monitoring Platform (NSMP) architecture

Security policy builds relationships with the policy rules engine API through Web service interface in NSMP. As a management tool, NSMP help users according to security needs change and the status of network attacks dynamic management policy rules, In the course of administration policy rule base simultaneously the integrity, accuracy and consistency of testing.

NSMP security policy is the core of the whole system security. From a functional perspective, the system's security policies include access control policy, security device configuration policy, emergency response policy and other policies. Security policy management, Web main page as shown in Fig.12, where access control policy includes account management policy, rights management policy and IP lockout policy, security equipment configuration policy includes firewall policy configuration policy, global HIDS policy and vulnerability scanning policy, emergency response policy includes the general system response policy to the attacks and the state and content of the adaptive management policy, and other policies include event correlation analysis of threshold policy and database monitoring policy.


网络安全综合监控平台
Network Security Monitoring Platform

安全策略

访问控制	
帐户管理策略	在多级NSMP系统中，统一配置本级和下级各角色成员的帐户及登录策略
权限管理策略	按角色设置本级和下级网络节点系统功能的访问权限
IP锁定策略	配置IP锁定策略
安全设备配置	
防火墙策略	配置防火墙规则
全局HIDS策略	根据网络资产的重要程度，提出对主机入侵检测布署和配置的指导性意见
漏洞监控策略	配置本级网络漏洞监控参数及扫描目标
应急响应	
应急响应策略	配置本地网络安全管理员制定的响应规则(阻断IP,邮件,声音,消息)
自适应管理策略	配置安全策略状态及内容改变的响应规则
其它	
事件关联分析阈值	配置告警关联事件的时间阈值
数据库监控配置	配置数据自动备份策略

Figure 12. Network Security Monitoring Platform (NSMP)

VI. Experiments and Results

Our experiments include testing the application and efficiency of improved Rete algorithm and the efficiency of rule-based security policy implementation and realization.

A. Application Testing

The matching efficiency of rule conditions determines the engine performance. Engine need to quickly test data objects in the work memory and policy rule knowledge, find the corresponded rules from the loading rule set, and generate the rule execution instance. Here using the improved object-oriented Rete network structure search algorithm. Application testing environment is Eclipse 3.2 and JUnit test/simulator.

Fig. 13 is the Rete network simulation diagram of the IP blocking control policy rules, showing that it contains four policy rules of the network structure.

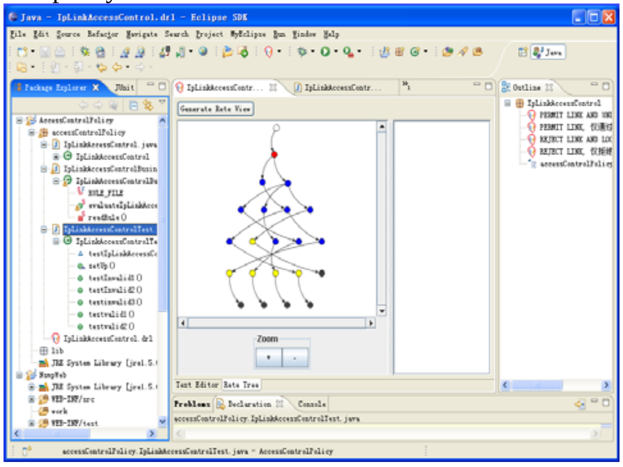
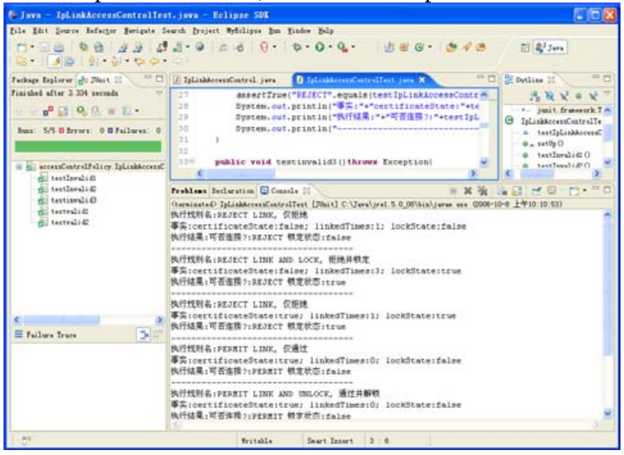


Figure 13. Rete simulation diagram of IP blocking control policy rules

The test case (TestCase) results of policy processing model based on rule engine as shown in Fig. 14. TestCase tests IP-blocking control policy for the rules of the five kinds of possible results, and the test is passed.


```
        assertTrue("REJECT".equals(testIpLinkAccessContr
        System.out.println("事实:"+"certificateState:"+te
        System.out.println("执行结果:"+"可否连接?:"+testIpL
        System.out.println("-----------------------------
    }

    public void testinvalid3()throws Exception{
```

```
<terminated> IpLinkAccessControlTest [JUnit] C:\Java\jre1.5.0_08\bin\javaw.exe (2008-10-8 上午10:10:53)
执行规则名:REJECT LINK, 仅拒绝
事实:certificateState:false; linkedTimes:1; lockState:false
执行结果:可否连接?:REJECT 锁定状态:false
--------------------------------------
执行规则名:REJECT LINK AND LOCK, 拒绝并锁定
事实:certificateState:false; linkedTimes:3; lockState:true
执行结果:可否连接?:REJECT 锁定状态:true
--------------------------------------
执行规则名:REJECT LINK, 仅拒绝
事实:certificateState:true; linkedTimes:1; lockState:true
执行结果:可否连接?:REJECT 锁定状态:true
--------------------------------------
执行规则名:PERMIT LINK, 仅通过
事实:certificateState:true; linkedTimes:0; lockState:false
执行结果:可否连接?:PERMIT 锁定状态:false
--------------------------------------
执行规则名:PERMIT LINK AND UNLOCK, 通过并解锁
事实:certificateState:true; linkedTimes:0; lockState:false
执行结果:可否连接?:PERMIT 锁定状态:false
```
Java - IpLinkAccessControlTest.java - Eclipse SDK
JUnit
Finished after 3.304 seconds
Runs: 5/5 Errors: 0 Failures: 0
Failure Trace
Writable
Smart Insert
3 : 6

Figure 14. TestCase of IP blocking control policy rules

B. Efficiency Testing

Our external test environment is the local network within the laboratory installation of network security monitoring platform. In the same condition, we have tested two ways of system efficiency using the improved Rete algorithm or not, they also reflect the rules matching efficiency under the different ways. The attacking event is the simulation of network data packets based on the ISS's reference documents about the attacking characteristics. As is shown in Fig. 15, the system is running start, initialization environment, attacks on the network to response more slowly, but after the initialization is completed, the improved Rete algorithm than the classic way in the system average response time can be quickly nearly doubled.

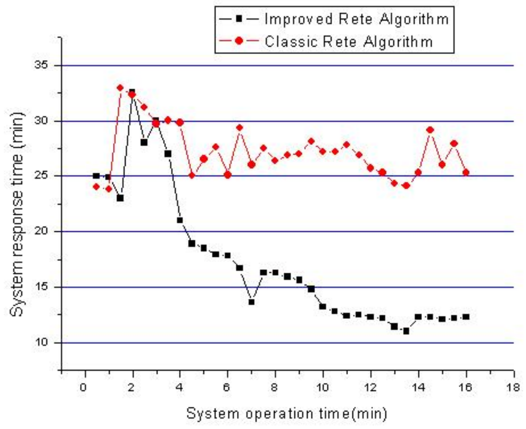


Figure 15. System operation efficiency of improved Rete algorithm

In the also same conditions, we have tested two ways of system operation efficiency as shown by Fig.16. They also reflect different ways of matching the efficiency rules. Testing targets are system average response time of access control policy and events response policy. The results show that application rule engine approach reflects slow response to attacks in the system during initialization, but at the completion of initialization process, application rule engine is more efficient than no application it.

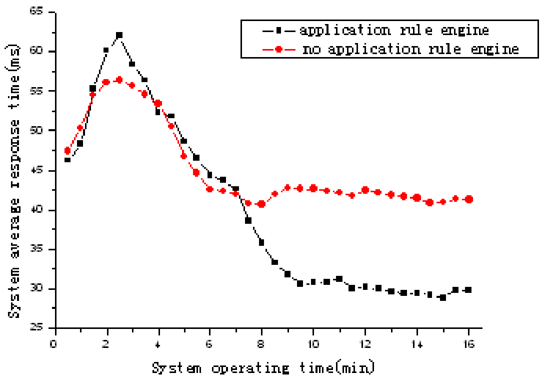


Figure 16. System operation efficiency using rule engine

VII. Specific Application

According to the above security policy implementation and realization model, establishing a security network information system, need to determine the security domain entity classification for all entities under the system security requirements firstly, and then create security services management policy and a unified security policy database. Fig.17 gives the application of security policies in NSMP (Network Security Monitoring Platform) under the circumstances of single-class system. The dotted line is the process of security policy data flow. Through various network policy equipment and security software centralized management and control, the original separation of network security equipment into an organic whole collaboration. Real-time dynamic monitor, comprehensive security audit, dynamic policy management and the timely response to the threat provide reliable basis for the overall security policy formulation and implementation. It is through security policy to achieve unified management of the system itself and network security equipment, including allowing or refusing access to the system or security equipment under what conditions, demanding the system or safety equipment to make what responses in what incident, such as interdiction IP, mail or message, voice warning, IDS adjustment, Scanner start-up, etc.

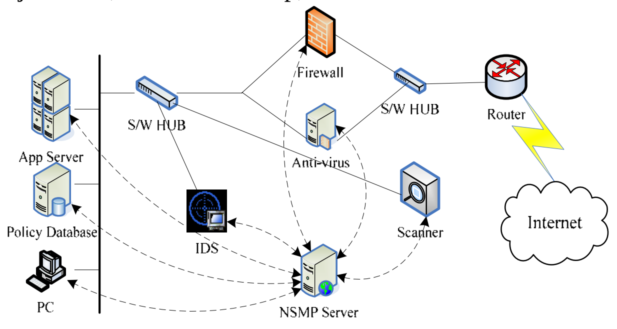


Figure 17. The application of security policies in NSMP

VIII. Conclusion

Rete is currently the most efficiency forward chaining algorithm of the production system, is the only decision-making support algorithm which efficiency is independent of rules number, and its core idea is to use the separate matches to construct matching tree dynamically based on content, in order to achieve significantly reduce the computational results. The improve idea is the expansion of attributes of the pattern network and connection network nodes, and the connection order of the connection network nodes that is appropriate to policy class description, thereby establishing the extended Rete class structure and network structure, and using this high-performance pattern matching algorithm and network as a guarantee to improve system implementation efficiency.

From the steps of policy rules engine can be seen, the establishment and storage of policy rules can be independent of the application system development process, and can according to market security needs changes and timely changes dynamically, so that security policy management has high adaptability and robustness, this processing mechanism of security policy has been applied in a number of practical projects currently, and experiments have proved that the policy rule-based engine model has a higher processing efficiency. As an important technique of solving security problem of current system, making a deep research into security policy will have great effort on the security management of already-existing and further system.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant No.60970146, the China Postdoctoral Science Foundation under Grant No.20070420793, the Doctoral Fund of Ministry of Education of China under Grant No. 20090171120001, and the Department of Education research project in Guangxi, P.R. China. The supports from electronic information security laboratory of Sun Yat-sen University are highly appreciated. The helpful comments from anonymous reviewers are also gratefully acknowledged.

REFERENCES

[1] Song Zhen and Li Lian zhi, "COMP: an efficient multi-pattern/multi-object match algorithm", Journal of Harbin Institute of Technology, 2001, vol.33, No.5, pp.668-670.

[2] CL. Forge, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattren Match Problem", Artifical Intelligence, 1982, vol.19, No.1, pp. 17-37.

[3] Neves P and Barata J, "Evolvable production systems", Conference on IEEE International Symposium on Assemble and Manufacturing, IEEE press, Suwon, 2009, pp.189-195.

[4] Blumel E, "Virtual Development, Testing and Learning Platforms for the Integrated Development of Products and Production Systems", Conference on IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE press, Rome, 2008, pp.xxvi-xxvii.

[5] B. L. Cohen, "A Powerful and Efficient Structural Pattern Recognition System", Artifical Intelligence, 1977, No.9, pp.223-255.

[6] J McDermott, A Newell and J More, "The Efficiency of Certain Production System Implementation", Waterman D. A. & Hayes-Roth F., (Eds), Pattern - Directed Inference System, Academic Press, NY, 1978.

[7] Dongdai Zhou, Yifan Fu, Shaochun Zhong and Ruiqing zhao, "The Rete Algorithm Improvement and Implementation", International Conference on Information Management, Innovation Management and Industrial Engineering, IEEE press, Taipei, 2008, vol.1, pp.426-429.

[8] Minsoo Kim, "Fast Serive Selection Using rete Network in Dynamic Environment", 2009 World Conference on Services-I, IEEE Press, Los Angeles, CA, 2009, pp.85-92.

[9] Wu Zhiuing, Lu Hanrong, "An Improved Implementation of RETE Algorithm in OCLIPS", Computer Engineering and Design, 1996, Vol.17, No.3, pp.22-27. (in Chiness)

[10] M. Kohli, J. Lobo, "Realizing Network Control Policies Using Distributed Action Plans", Journal of Network and Systems Management, 2003, vol. 11, No.3, pp.305-327.

[11] Jonathan D. Moffett, Morris S. Sloman, "Policy Conflict Analysis in Distributed System Management", http://www.moffett.me.uk/jdm/pubs/polconfl.pdf, 1993.

[12] E Lupu, M Sloman, "Conflict Analysis for Management Policies", Conference on Fifth IFIP/IEEE International Symposium on Integrated Network Management IM'97, Chapman & Hall Publishers, San-Diego, May 1997, pp 430-443.

[13] M. Proctor, M.Neale, P.Lin, and M.Frandsen, "JBoss Rules User Guide 3.0", http://labs.jboss.com/file-access/default/members/jbossrules/freezone/docs/3.0.1/html_single/index.html ,2006.

Chenghua Tang, born in 1974. He is a research post doctor with the electronic information security laboratory of Sun Yat-sen University, China. He had received the Ph.D. from the Beijing Institute of Technology. His main research areas include information security, Data Fusion, and policy management, etc.

Yi Xie, born in 1973. He received the B.S. degree and M.S. degree both from Sun Yat-Sen University, Guangzhou, China. From July 1996 to July 2004, he was working in Civil Aviation Administration of China (CAAC). He returned to academe in 2004. He was a visiting scholar at George Mason University during 2007 and 2008. He is currently a teacher at School of Information Science and Technology in Sun Yat-Sen University, China. His research interests are in network security and Web user behavior model algorithm.