An Approach for Description of Computer Network Defense Scheme and Its Simulation Verification

Zhao Wei, Chunhe Xia, Yang Luo, Xiaochen Liu and Weikang Wu
Beijing Key Laboratory of Network Technology, School of Computer Science and Engineering, Beihang University, Beijing, China
Email: wz@cse.buaa.edu.cn, xch@buaa.edu.cn, veotax@sae.buaa.edu.cn, ann4498@sina.com, wuweikang2012@gmail.com

***Abstract*—In order to solve the problem of which the existing defense policy description languages can only describe some aspects of defense, such as protection or detection, but cannot express relationship among actions and to cope with large-scale network attack, we proposed an approach for description of computer network defense scheme and its simulation verification. A computer network defense-oriented scheme description language (CNDSDL) was designed to describe actions of protection (i.e., access control, encryption communication, backup), detection (i.e., intrusion detection, vulnerability detection), analysis (i.e., log auditing), response (i.e., system rebooting, shutdown), recovery (i.e., rebuild, patch making), and relationship among actions (i.e., sequence-and, sequence-or, concurrent-and, concurrent-or, and xor). The Extend Backus-Naur Form (EBNF) of CNDSDL was provided. At last, we provided an implementation mechanism of CNDSDL. A task deadlock detection algorithm was given for the defense scheme. The simulation was completed in simulation platform of GTNetS. Three simulation experiments verified the description capability and effectiveness of CNDSDL. The results of the experiments show that the defense scheme described by CNDSDL can be transformed to detailed technique rules and realize the defense effect of expression.**

***Index Terms*—defense, deadlock detection, EBNF, scheme description language, simulation verification**

I. Introduction

Researchers have proposed defense technology and mechanisms to protect network security. For example, detection [1] or response [2] mechanism of defense scheme for DDoS attack, a linkage defense framework [3-4] that makes IDS and Honeypot work together, a mechanism [5] that detects the conversion channels through the interaction of IDS and firewalls.

As the scale of network grows, it is a huge challenge for administrator to manage different defense mechanisms and devices in a large-scale network environment. In order to solve these problems, researchers have proposed policy-driven management methods [6] to simplify the management for the complicated and distributed network system, such as cloud framework [7]. Administrator may specify the targets and constraints only in the form of policy. Thus, a variety of policy description languages are proposed [8-9]. However, most of these policy languages [10-12] focus on one aspect of the network defense, such as access control. And these policy languages can't describe the interaction actions of linkage mechanism for defense devices.

Based on the characteristics of computer network defenses (CND), we intend to provide a language description for linkage defense of different defense mechanism to cope with large-scale network attack and make it convenient for the security management of administrator.

Computer network defenses are actions through the use of computer networks to protect, monitor, analyze, detect, and respond to unauthorized activities within Department of Defense information systems and computer networks [13].

Based on these problems, we proposed an approach for description of computer network defense scheme and its simulation verification. We designed a formalized description language of defense scheme and its implementation mechanism and completed simulation verification in simulation platform of GTNetS. This language can describe actions of protection, detection, analysis, response, recovery, and relations among actions. These relations include sequence-and, sequence-or, concurrent-and, concurrent-or, and xor.

The remaining parts of this paper are organized as follows: The relevant literature is discussed in Section II, the description of computer network defense scheme is presented in section III, an implementation mechanism of CNDSDL is provided in Section IV, and the experiment verification and analysis are shown in section V, followed by a conclusion of our work in section VI.

II. Related Work

Many studies on defense scheme focus on defense mechanism. For example, Reference [1] proposes a new defense scheme to develop a flow monitoring scheme to defend against DDoS attacks in mobile ad hoc networks. In this paper, they provide a new defense mechanism that consists of a flow monitoring table at each node. It contains flow id, source id, packet sending rate and destination id. Reference [2] proposes an effective DDoS

Manuscript received June 1, 2013.
Corresponding author: Zhao Wei (wz@cse.buaa.edu.cn)

doi:10.4304/jcp.9.2.388-395

attack defense scheme using web service performance measurement and development of a DDoS attack response system. Reference [14] presents a two-tier coordination defense scheme for detecting and mitigating DDoS attacks. The first tier traffic filter (1st-TF) filters suspicious traffic for possible flooding. The second tier traffic filter (2nd-TF) performs online monitoring of queue length status with RED/Droptail mechanism for any incoming traffic. Reference [15] proposes a robust scheme to defend these routing attacks in MANETs and improve the performance of the networks.

How do we provide a language description for linkage defense of different defense mechanisms to cope with large-scale network attack? This is a problem to be solved. At present, there are a variety of policy description languages to express defense actions.

Reference [16] provides a policy description language-Ponder. This is an oriented-object and illustrative language. It can define access control policy based on role and management policy, such as condition-response rule of event triggering. Then, Ponder2 [17] is applied to miniature embedded system and large-scale complex network for network management. XACML [18] is a general language which is developed by OASIS for accessing control. TPL [19] is used to define authorization policy of Internet services. But its grammar of XML is very tediously long and unreadable. These policies contain many rules but lack succession and reusing. REI [20] is a semantic policy description language in the ubiquitous computing environment including access control policy and management policy. According to the difficulty in the management of distributed network, reference [21] suggests a high-level security configuration description language-FLIP to describe access control policy of firewall.

Nevertheless, most of these policy languages focus on one aspect of the network defense, such as access control. They lack a unified defense scheme description language that can describe defense actions of protection, detection, analysis, response, recovery, and relations among actions to cope with complicated network attacks.

III. Description of The Computer Network Defense Scheme

In this section, we provide a description of computer network defense scheme including a formalized definition of computer network defense scheme, CNDSDL and its EBNF.

A. The formalism Definitions of Defense Scheme

Definition 1. *Defense Scheme.* It is a plan that consists of tasks to achieve defense intention. It is a two-tuple including task set and relation set. It is represented as follows:

$$\begin{cases} Scheme::=(\zeta, R); \\ \zeta ::= \{Task_i | 1 \le i \le n\}; \\ R \subseteq \zeta \times \zeta. \end{cases}$$

Wherein, ζ denotes the set of tasks, R denotes the set of relations between tasks.

Definition 2. *Task.* It is a six-tuple which includes subject, operation, execution time, execution results and constrains. It is represented as follows:

$$\begin{cases} Task ::= (sub, Operation, Time, Effect, Constra\,\text{int}); \\ sub \in Subject; \\ Operation ::= \{ope_i | 1 \le i \le n\}. \end{cases}$$

Wherein, *Subject* are the subjects in the network that can execute tasks; *Operation* is the set of operations of a task; *Time* denotes the starting time when a task runs. *Effect* denotes the executing result of tasks including success and failure. *Constra*int denotes the prerequisites of tasks.

Definition 3. *Subject.* It refers to all the hardware and software resources participating in network defense such as firewall $S_{firewall}$, IPsec VPN $S_{ip\,\text{sec}\_vpn}$, intrusion detection system (IDS) $S_{\text{int}\,rude\,\text{det}\,ect}$, vulnerability library S_{vul_server}, log audit system $S_{\log audit\_system}$, operation system $S_{operation\_system}$, backup server S_{backup_server}.

Definition 4. *Operation.* It is a three-tuple that consists of action, object of action, and input parameters of actions. It can be represented as follows:

$$\begin{cases} ope_i ::= (action, object, InPara); \\ action \in Action; object \in Object. \end{cases}$$

Wherein, *Object* denotes the set of objects including node O_{node}, service $O_{service}$, application program $O_{application\_program}$ and data packet O_{data_packet}. *InPara* denotes the set of input parameters of actions. *Action* was defined as follow.

Definition 5. *Action.* It denotes the set of defense actions including protection action (such as permit action A_{permit} and deny action A_{deny} of firewall, the permission Encryption action A_{permit_crypt} of IPsec VPN, backup action A_{backup} of backup server), detection action(such as alerting action A_{alert} of IDS, scan action $A_{vulscan}$ of vulnerability scan server), responding action(such as rebooting A_{reboot} and shutdown $A_{shutdown}$ action of operation system), analysis action(such as log audit action $A_{\log\_audit}$ of log audit system) and recovery action(making patch action $A_{makepatch}$ of operation system, rebuild action $A_{rebuild}$ of backup server).

Definition 6. *Relation of Task.* It means temporal and logic relation which include sequence and r_{seq_and}, sequence or r_{seq_or}, concurrent and $r_{concurrent\_and}$, concurrent or $r_{concurrent\_or}$, xor r_{xor}. To simplify the discussion, we assume that there are only two tasks in one scheme. $Task ::= \{task_1, task_2\}$; Each relation is explained separately as follows:

r_{seq_and}: If $seq\_and(task_1, task_2)$, it denotes that the $task_1$ is executed firstly. If the executing effect of $task_1$ is true, the $task_2$ is executed as follows. Only when both

tasks are successfully completed can we say the scheme is finished successfully.

r_{seq_or} : If $seq\_or(task_1, task_2)$, it denotes that the $task_1$ is executed firstly. If the executing effect of $task_1$ is true, the $task_2$ does not need to be executed. If the executing effect of $task_1$ is false, the $task_2$ must be executed. Whether the scheme is finished successfully depends on the success of $task_1$ or $task_2$.

$r_{concurrent\_and}$:If $concurrent\_and(task_1, task_2)$, it denotes that both $task_1$ and $task_2$ are executed at the same time. Only if the effects of $task_1$ and $task_2$ are true can we say that the scheme is successfully finished.

$r_{concurrent\_or}$:If $concurrent\_or(task_1, task_2)$, it denotes that both $task_1$ and $task_2$ are executed at the same time. Only if there is a true executed effect between $task_1$ and $task_2$, we can say that the scheme is successfully finished.

r_{xor} :If $xor(task_1, task_2)$, it denotes that there exists one executing task between $task_1$ and $task_2$. Whether the scheme is finished successfully depends on the true effect of $task_1$ or $task_2$.

B. EBNF of CNDSDL

Based on the discussions above about the concepts and relations, we propose a CNDSDL in this section. Its grammar is expressed in EBNF as follows:

The defense scheme described by CNDSDL consists of three main parts: global variables declaration and definition; task description; and the tasks' relation description.

<scheme>::=[<global_variable_statement>]<tasks>[task_relations':'<task_relations>]

<tasks>=<task>|<tasks><task>

(1) Global variables declaration and definition

Global variable sentence is used to define global variable including statement sentence and variable assignment sentence. Global variable is alive during the entire time of scheme.

<global_variable_definiton>::=globals:<variable_statement>;|<variable_assignment>;|<global_variable_statement><variable_statement>;|<global_variable_statement><variable_statement>;

<variable_statement>::=<variable_type><variable_name>;

<variable_type>::=ip|time|int|float|string

The global variable type can be extended by adding key words. These variables are string.

(2) Task description

• Task

<tasks>::=<task>|<tasks>;<task>

<task>::=task <num> '{' subject: <subject> actions: '(' <actions> ')'[time: <time >] [constrains: '{' <constrains>'}']

• Subject

<subjcet>::=<protection_subject>|<detection_subject>|<analysis_subject>|<response_subject>|<recovery_subject>

<protection_subject>=back_up_server<num>|firewall<num>|gateway <num>|cryptor <num>| host <num>| server <num>

<detection_subject>::=IDS<num>|anti_virus_system<num>|vul_base<num>

<analysis_subject>::=audit_system<num>

<response_subject>::=server <num>|host <num>

<recovery_subject>::=back_up_server <num> | host <num>| server <num>

• Actions

<actions>::=<action>|<actions>,<action>

<action>::=<protect_action>|<detect_action>|<analysis_subject>|<respond_action>|<recover_action>

<protect_action>::=<protect_act><protect_obj>[inPara:'{'<protection_inParas>'}']

Protect action consists of protect act, protect object and parameter. Its EBNF is shown as follows:

<protect_act>::=back_up|permit|deny|crypt|authenticate

<protect_obj>::=<file>|<packet>|ip

<packet>::=<ip_packet>|<tcp_packet>|<udp_packet>|<icmp_packet>

<ip_packet>::=IP <src_ip><dst_ip>

<tcp_packet>::=TCP<src_ip><ports> <dst_ip><ports>

<udp_packet>::=UDP<src_ip><ports><dst_ip><ports>

<icmp_packet>::=ICMP<src_ip><ports><dst_ip><ports>

<src_ip>::=(ip/mask)|any

<dst_ip>::=(ip/mask)|any

<ports>=<port>|<port>:<port>|<port_operator><port>|any

<protection_inPara>::=priority:<num>|type:(full|addition|offset)|crypt:(Y|N)|secure_trans:(Y|N)|interface:<num>

Detect action consists of detect act, detect object and parameter. It contains intrusion detection, virus checking, and vulnerability scanning. Its EBNF is shown as follows:

<detection_action>::=<detect_act><detect_obj>[in_Para:'{'<detection_inParas>'}'

<detect_act>::=ids_detect| check_virus|vul_scan

<detect_obj>::=<IDS_rule>|<virus>|<vul>|<log>

<virus>::=<string>

<vul>::=cve-<cve_year>-<cve_number>

<log>::=<file>

<detection_inPara>::=(host:<num>)|(ip:<ip>)|(service:<service_name>)

<service_name>::=Web|Telnet|Rlogin|Ftp|SMTP

<ids_rule>::=<idsRule_head><idsRule_body>

<idsRule_head>::=<idsRule_action><packet>

<idsRule_action>::=alert|pass|log

<idsRule_body>::='('<options>')'

<options>::=<option>|<options>;<option>

<option>::=(message:<string>)|(content:<bin-str>|<string >)|(refenrence:<vul>)|(fw:<num>|<ip>)|(vbase:<num>|<ip>)|resp:(rst_all|rst_rcv|rst_send|icmp_all|icmp_host|icmp_net|icmp_port)

Response action consists of response act, response object, and parameter. It contains account locking, system shut down, reboot, patch installing, file deleting, process killing, and file access authority. Its EBNF is shown as follows:

<response_action>::=<response_act><response_obj>[inPara:'{'<response_inPara>'}'

<response_act>::=lock|shutdown|reboot|install|patch|delete|kill|set_file_access

<response_obj>::=<account>|<patch>|<file>|<process>|<service_name>

<response_inPara>::=access_authority:<access_authority>|account:<account>

<access_authority>::=R|W|X|RW|RX|WX|RWX

Recovery action consists of recovery act, recovery object, and parameter. It contains rebuild and redundant. Its EBNF is shown as follows:

<recovery_subject>::=back_up_server <num>|host <num> |server <num>

<recovery_action>::=recover<recovery_obj>[inPara:'{'<recovery_inPara>'}']

<recovery_obj>::=<file>|<service_name>

<recovery_inPara>::=host<num>|server<num>|date:<date>

- Time

Time denotes the task's start time.

<time>::=<num> (s|ms|us)

- Constrains

Constrains denotes some conditions and environment necessary to execute task for task subject. <constrain>::=<conditions>

<conditions>::=<condition>|<conditons>;<condition>

<conditon>::=<state_condition>|<expression_condition>

<state_condition>::=<state_variable><state_operator><state_value>

<state_variable>::=cpu_ratio|mem_ratio|bandwith|disk_pace

<state_value>::=<float>|<int>

(3) Tasks' relation description

<task_relations>::=<task_relation>|<task_relations>;<task_relations>

<task_relation>::=seq_or('<num>,<num>')'|seq_and'('<num>,<num>')'|con_or'('<num>,<num>')'|con_and'('<num>,<num>')'|xor'('<num>,<num>')'

IV. The Implementation Mechanism of CNDSDL

The defense scheme described by CNDSDL needs scheme interpretation and deployment so that the simulation can be executed on the simulation platform. The implementation mechanism of CNDSDL includes three modules: scheme interpretation, scheme deployment, and scheme simulation. In the scheme interpretation module, we executed the lexical analysis, syntactic analysis, and identification of meanings for defense scheme that is described by CNDSDL through lexical and syntactic analyzer lex/yacc in order to check grammar errors and generate the corresponding tasks. In the scheme deployment module, we executed task deadlock detection at first. If a deadlock exists, the scheme is refused to execute. Otherwise, tasks in the scheme will be scheduled and deployed to corresponding simulation node to realize simulation. In the scheme simulation module, we realize simulation of defense tasks of IDS, firewall, vulnerability library, patch making, and system rebooting with network topology information and generated the simulation executing report of defense scheme.

The system architecture of the implementation mechanism of CNDSDL is shown in Fig.1.

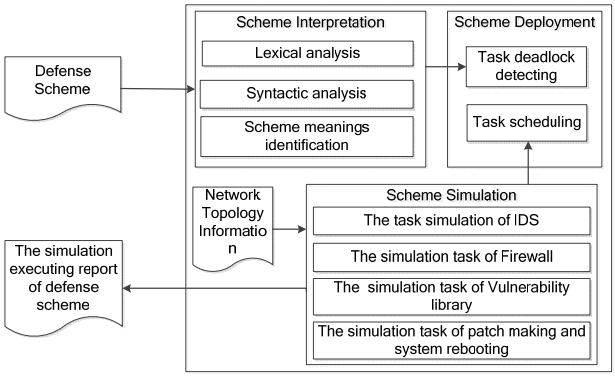

Figure 1. System architecture of implementation mechanism of CNDSDL

In the scheme deployment module, we designed some algorithms to detect deadlock in the defense scheme. At first, task graph and task deadlock are defined as follows:

Definition 7. *TaskGraph.* Suppose the task set in a scheme is ζ and a set of sequential relations is $\prec$ including the relations of "seq_or" and "seq_and" among tasks, the TaskGraph is $TaskGraph ::= < Nodes, Edges >$,wherein $Nodes = \zeta$, $Edges = \prec$, $TaskGraph$ is a diagraph and the directions of the edges indicate the sequential relations.

Definition 8. *Task deadlock.* Suppose the two tasks include t_i and t_j in one scheme: the finish of one task is the premise of the other, or the start of executing one task requires the successful execution of the other. This can be represented by $t_i \prec t_j$. If $t_i \prec t_j$ and $t_j \prec t_i$ exist in one scheme, there is a task deadlock.

The algorithm of task deadlock detection is described as follows. (1) First a graph of task relation is constructed. The task set ζ is derived from scheme. Then, a task is taken as node to be added to task relation graph $TaskGraph$. It means $Nodes = \zeta$. At the same time, we can get $Edges = \varnothing$. Any two tasks t_i and t_j can be analyzed to check whether there is a sequential relation $t_i \prec t_j$ or $t_j \prec t_i$. At last, task relation is taken as edge to be added to the set of edge $Edges$ for task relation graph $TaskGraph$. The number of pairs of tasks is $n(n\text{-}1)/2$. (2) Transitive closure is constructed for ζ which is a binary relation of the set ζ . If the number of element is n for set ζ , the transitive closure for $\prec$, we can get $t(\prec) = \bigcup_{i=1}^{n} \prec^i$. Then, we will construct a graph of transitive closure

TaskGraph' for the graph *TaskGraph*. It means the task of ζ is regarded as node of *TaskGraph*' and relation of $t(\prec)$ as edge of *TaskGraph*'. Because transitive closure $t(\prec)$ is constructed with transitivity of relation $\prec$, the sequential relation among tasks is not changed by *TaskGraph*'. If there is a sequential relation among task node for *TaskGraph*, it will be shown in a directed edge.(3) The edge set *Edges*' is checked for *TaskGraph*'. If there exists $t_i \prec t_j, t_j \prec t_i \in Edges'$, there is a deadlock between task t_i and task t_j. So this scheme will be refused to execute. Otherwise, this scheme will be simulated in corresponding node according to the sequential and logic relations among tasks.

Transitive closure is constructed with Floyd-Warshall algorithm. The time complexity of task deadlock detection algorithm is $O(n^3)$. wherein, $n = |\zeta|$. The pseudo-code of task deadlock detection algorithm is shown as follows:

```
ALGORITHM TASK_DEADLOCK_DETECTION
  INPUT: Task Set : ζ; Task Relation Set : ≺.
  OUTPUT: BOOL : ISDeadLock.
  PROCEDURE DeadLockDetection()
    TaskGraph :< Nodes, Edges >;
    TaskGraph' :< Nodes', Edges' >;
    Nodes ← ∅; Nodes' ← ∅;
    Edges ← ∅; Edges' ← ∅;
    // adding task set ζ to node set Nodes and Nodes' of task graph //
    AddNodeSet(TaskGraph, Nodes, ζ)
    AddNodeSet(TaskGraph', Nodes', ζ)
    // adding task relation to edge set Edges of TaskGraph //
    FOR i = 0 TO |ζ| DO
      FOR j = 0 TO |ζ| DO
        IF(n_i, n_j ∈ Nodes AND n_i ≠ n_j AND < n_i, n_j > ∈ ≺)
          AddEdge(TaskGraph, Edges, < n_i, n_j >);
        ENDIF
      REPEAT
    REPEAT
  // constructing a transitive closure graph TaskGraph' for TaskGraph //
    t(≺) = ⋃_{i=1}^{n} ≺^i ;
    FOR i = 0 TO |ζ| DO
      FOR j = 0 TO |ζ| DO
        IF(n_i', n_j' ∈ Nodes' AND n_i' ≠ n_j' AND < n_i', n_j' > ∈ t(≺))
          AddEdge(TaskGraph', Edges', < n_i', n_j' >);
        ENDIF
      REPEAT
    REPEAT
    // checking dead lock in TaskGraph' //
    FOR each e in Edges' DO
      FOR each e' ≠ e in Edges' DO
        IF(e' = e^{-1}) THEN
          ISDeadLock = TRUE;
          RETURN ISDeadLock;
        ENDIF
      REPEAT
    REPEAT
  END DeadLockDetection
```

V. The Experiments

In this section, we provide some defense scheme instances to illustrate the usage of CNDSDL. These defense schemes can be executed automatically in the simulation platform through our implementation mechanism of CNDSDL and the simulation effect can be observed.

A. Experiment Environment

The experiment environment is the network security simulation platform based on GTNetS. Network topology environment is shown in Fig.2. The whole network is divided into three main parts: external network, DMZ, and internal network. DMZ includes Web server, DNS server, FTP server and SMTP server (The corresponding IP addresses are 192.168.1.4/24, 192.168.1.5/24, 192.168.1.2/24 and 192.168.1.3/24.). The internal network is partitioned into two segments by switcher, i.e. Net 1 and Net 2. There are three hosts and one backup Server (IP:192.168.2.2/24)in Net 1; one host and one Database Server (IP :192.168.3.2/24) in Net 2.

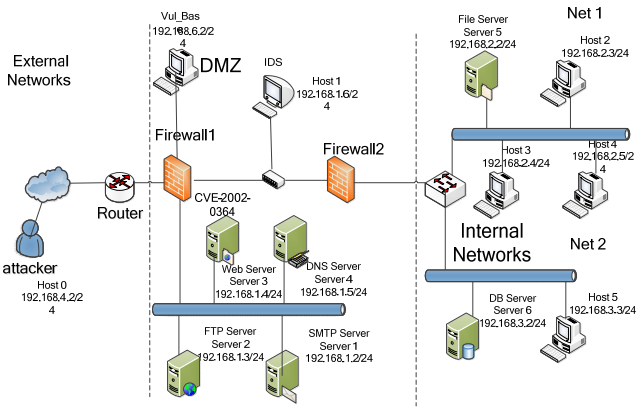

Figure 2. Network topology

The IP address, operation system, service information of hosts and servers are shown in Table I.

TABLE I.
HOSTS AND SERVERS INFORMATION

Outside Hosts			
IP Address	**Hostname**	**OS**	**Notes**
192.168.4.2	Host0	Redhat 5.0	kernel 2.0.32
DMZ Hosts			
192.168.1.2	Server1	Windows Server 2003	MS Exchange 2003 Mail server
192.168.1.3	Server2	Windows Server 2003	Windows IIS 5.1 FTP server
192.168.1.4	Server3	Red hat 5.0	Linux Apache 2.8.19 Web server
192.168.1.5	Server4	Windows 2000 server	DNS server
192.168.1.6	Host1	FreeBSD 4.0	IDS
Inside Hosts			
192.168.2.2	Server5	Windows Server 2003	System management software
192.168.2.3	Host2	Ubuntu 8.04	host in domain B
192.168.2.4	Host3	CentOS	host in domain B

192.168.2.5	Host4	Windows XP Home Edition sp1	host in domain B
192.168.3.2	Server6	Solaris 2.6	Oracle 11i Database server
192.168.3.3	Host5	Ubuntu 8.04	Domin SunPRC(linux) Database server

B. Verification and Analysis of the Experiments

(1)Defense scheme including one simulation task of access control of firewall

Scenario: We assume that the attacker can access FTP and SMTP servers at the beginning. Then this attacker is detected. So we must give a defense scheme to deny this attacker.

The defense scheme that is described by CNDSDL is shown as follows:

```
task 1 {  subject : Firewall 1;
          actions : (
          deny TCP 192.168.4.2 / 24  any 192.168.1.2 / 32 25
           inPara : {int erface :  4 },
          deny TCP 192.168.4.2 / 24  any 192.168.1.3 / 32 21
          inPara : {int erface :  4 },)
          time :  1; }
```

After executing this scheme, we find there are many denial rules in the firewall. This firewall ACL is shown in Fig.3.

```
task1 finished at 1sec
ACLRulse of EXNET Direction IN:
action     srcIP      destIP      sMask   dMask   sPort   dPort   protocol
deny   192.168.4.2  192.168.1.2    24      32       *       25       6
deny   192.168.4.2  192.168.1.3    24      32       *       21       6
```

Figure 3. The control platform results of the ACL in firewall1

In the simulation platform, we can find that the attacker cannot access FTP server. The simulation effect is shown in Fig.4.

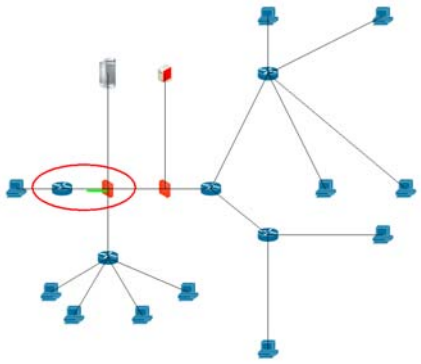

Figure 4. Denying packet from attacker in firewall1

(2) Defense scheme including simulation task of access control of firewall, patch making, system rebooting, and temporal-logic relations among these tasks.

Scenario: There is a presumption that the network exits some vulnerability shown in Server3 (CVE-2003-0542), Server4 (CVE-2007-0939) and Host2 (CVE-2005-0753). By utilizing these vulnerabilities, the attacker can gain root access and bring about DoS attack. The attacker can access Net1 and Net2 through the DMZ area to form some attacking paths which are found with the tool in reference [22], such as Host0->Sever3->Host2, Host0->Server4->Host2.

To cope with the situation above, we have designed a defense scheme: At first, we must deny the attacker accessing the Server3 and Server4. Then, we must install some patch on Server3 and Server4, and reboot system subsequently. The defense scheme description using CNDSDL is shown as follows:

```
globals : ip attIP = 192.168.4.2; int  t1 = 1; float t2 = 1.5;
task 1{subject :  Firewall 1;
       actions :  (deny  IP attIP / 32 192.168.1.4 / 32  inPara : {int erface :  4 },
       deny  IP attIP / 32 192.168.1.5 / 32 inPara : {int erface :  4 })
       time :  0.5;},
task 2{subject : Server 3;
       actions : ( patch  httpd - 2.0.46 - 26)
       time :  t1;},
task 3{subject : Server 3;
       actions : (reboot)
       time :  t2;},
task 4{subject : Server 4;
       actions : ( patch KB924430)
       time :  t1;},
task 5{subject : Server 4;
       actions : (reboot)
       time :  t2;},
task 6{subject : Host 2;
       actions : ( patch cvs - 1.11.2 - 27)
       time :  t1;},
task 7{subject : Host 2;
       actions : (reboot)
       time : t2;},
task _ relations : seq _ and (2,3); seq _ and (4,5); seq _ and (6,7).
```

In this scheme, task1 gives a description which denotes firewall preventing attackers from access Server3 and Server4. Then Server3, Server4, and Host2 are installed patch and reboot subsequently in task2, task3, task4, task5, task6, task7. The expression “seq_and” denotes that patch installing is finished before system rebooting. It is shown in Fig.5.

```
the packet from 192.168.4.2 to 192.168.1.4 has been blocked
the packet from 192.168.4.2 to 192.168.1.5 has been blocked
Got mouse move event
There is a patch in the Server3
httpd-2.0.46-26
There is a patch in the Server4
KB924430
There is a patch in the Host2
cvs-1.11.2-27
```

Figure 5. The control platform results of the patch installing and rebooting

(3) Defense scheme including simulation tasks of access control of firewall, intrusion detection of IDS, vulnerability library, and temporal-logic relations among these tasks.

Scenario: There is a presumption that the attacker can bypass the Firewall2 and access DB server in Net2 and file server in Net1 according to the configuration venerability of firewall. The attacker send a one byte packet to server which runs the service of Oracle 9i 9.0.1. These results in triggering the vulnerability of daemon process TNS Listener. So, it will conduct the dos attack. In addition, attacker can also conduct the buffer overflow attack for the file server.

To cope with the situation above, we have designed a defense scheme: At first, IDS can detect the dos or buffer overflow attack. Then, IDS inquires the vulnerability library to affirm vulnerability information. At last, the firewall denies the packet from attacker. Three tasks and task relation “seq_and” are used to describe the situation

mentioned above. This defense scheme description using CNDSDL is shown as follows:

```
task 1{
        subject : IDS 1;
        actions :
          (alert IP 192.168.4.2/32 192.168.2.0/24
          (message :"bufferoverflow";content :"01000110";)
           alert TCP 192.168.4.2/24 any 192.168.3.0/24 21
          (message :"dos";content :"00110010";))
         time : 0;
},
task 2{
        subject : vbase :192.168.101.2;
        actions : (vulcheck cve - 2007 - 5398;cve - 2002 - 0509;)
       time : 0.5;
},
task 3{
        subject : Firewall 2;
          actions :
          (deny IP 192.168.4.2/24 192.168.2.0/24 inPara : {int erface : 4},
           deny TCP 192.168.4.2/24 192.168.3.0/24 inPara : {int erface : 4}
          )
          time :1.2;
}
task _relations : seq _and(1,2);seq _and(2,3).
```

This defense scheme is deployed in the simulation platform. The simulation effect is shown as follows:

When the dos attack is detected by IDS, IDS inquires the vulnerability library to affirm this vulnerability information. In the Fig.6, the yellow packet in the circle denotes enquiring packet from IDS to vulnerability library.

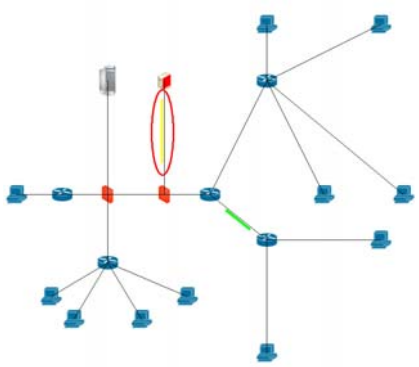

Figure 6. Inquiring message sending from IDS to vulnerability library

The vulnerability library queries the database and affirms this attacking. Then the vulnerability library sends affirmed information to IDS. In the Fig.7, the gray packet in the circle denotes affirmed packet from vulnerability library to IDS.

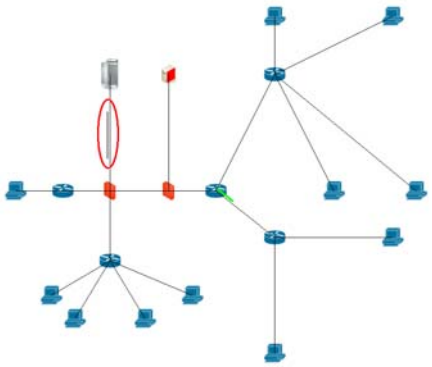

Figure 7. Affirming message sending from vulnerability library to IDS

In the Fig.8, the red packet in the circle denotes that IDS informs the firewall1 to forbid the unlawful access after receiving the vulnerability affirmed information.

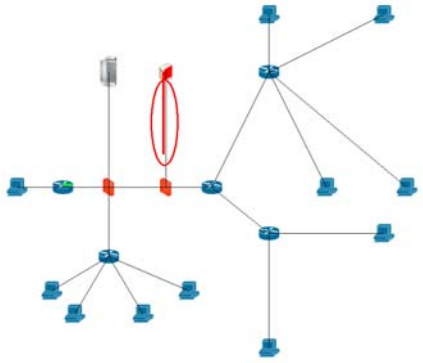

Figure 8. Denying packet message sent from IDS to firewall

Now, the packet of attacker cannot bypass the firewall1. It is shown in Fig.9. The control platform results of the packet denying from attacker are shown in Fig.10.

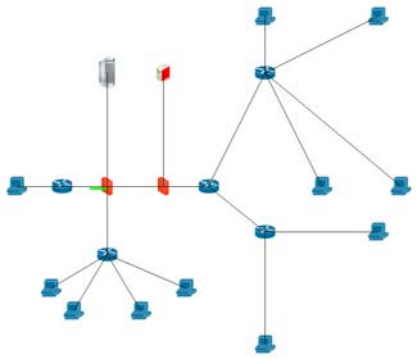

Figure 9. The packet is denied from attacker

```
IP access list 4
the packet from 192.168.4.2 has been blocked
```

Figure 10. The control platform results of the packet denying from attacker

VI. CONCLUSION

In this paper, we proposed an approach for description of computer network defense scheme and its simulation verification. The formalized definition of computer network defense scheme was provided and a novel computer network defense-oriented scheme description language (CNDSDL) was designed to describe the actions of protection (i.e., access control, encryption communication, backup), detection (i.e., intrusion detection, vulnerability detection), analysis (i.e., log auditing), response (i.e., system rebooting, shutdown), recovery (i.e., rebuild, patch making), and relations among actions. These relations include sequence-and, sequence-or, concurrent-and, concurrent-or, and xor. This language provides a language interface of linkage defense for the different security devices and its EBNF was given. On the other hand, we provided the implementation mechanism of CNDSDL. A task deadlock detection algorithm was designed for the defense scheme. At last, we conducted three simulation experiments of defense scheme: simulation task of access control of firewall; simulation task of access control of firewall, patch

making, system rebooting, and temporal-logic relations among these tasks; simulation tasks of access control of firewall, intrusion, detection of IDS, vulnerability library, and temporal-logic relations among these tasks. The results of these experiments verified the description capability and effectiveness of CNDSDL. In our future work, we will describe a variety of defense schemes in CNDSDL in order to further verify our language's description capability.

ACKNOWLEDGMENT

This work is supported by the following funding sources: the National Nature Science Foundation of China under Grant No. 61170295, the Project of National ministry under Grant No.A2120110006, the Co-Funding Project of Beijing Municipal education Commission under Grant No.JD100060630 and the Project of BUAA Basal Research Fund under Grant No.YWF-11-03-Q-001.

REFERENCES

[1] S.A.Arunmozhi and Y.Venkataramani, "A New Defense Scheme against DDoS Attack in Mobile Ad Hoc Networks," *International Conference on Computer Science and Information Technology, Bangalore, India, p p.210-216, 2011.*

[2] N.Baik, A.Sungsoo and K.Namhi, "Effective DDoS attack defense scheme using web service performance measurement," *International Conference on Ubiquitous and Future Networks, Phuket, Thailand, pp.428-433, 2012.*

[3] B.Khosravifar and J. Bentahar, "An experience improving intrusion detection systems false alarm ratio by using honeypot," *International Conference on Advanced Information Networking and Applications, Okinawa, Japan, pp.997-1004, 2008.*

[4] B.Khosravifar, M. Gomrokchi and J. Bentahar, "A multi-agent-based approach to improve intrusion detection systems false alarm ratio by Using Honeypot," *International Conference on Advanced Information Networking and Applications, Bradford, United Kingdom, pp.97-102, 2009.*

[5] S.Hammouda, L.Maalej and Z.Trabelsi, "Towards optimized TCP/IP covert channels detection, IDS and firewall integration," *International Conference on New Technologies, Tangier, Morocco, pp.1-5, 2008.*

[6] J.P.Loyall, M.Gillen, A.Paulos et al, "Dynamic policy-driven quality of service in service-oriented information management systems," *SOFTWARE-PRACTICE & EXPERIENCE*, Vol.41, No.12, pp.1459-1489, 2010.

[7] X.LUO,M.SONG and J.SONG, "Research on service-oriented policy-driven IAAS management," *The Journal of China Universities of Posts and Telecommunications*, Vol.18, pp.64-70, 2011.

[8] M.D.Amicoa, G.Sermeb, M.S.Idreesa, A.S.de.Oliveirab, Y.Roudiera, "HiPoLDS: A Hierarchical Security Policy Language for Distributed Systems," *Information Security Technical Report*, Vol.17, No.3, pp.81-92, 2013.

[9] J.Poroora, B.Jayaramanb, "C2L:A Formal Policy Language for Secure Cloud Configurations," *Procedia Computer Science*, Vol.10, pp.499-506, 2012.

[10] Q.Ni, E.Bertino, "xfACL: An extensible functional language for access control," *ACM Symposium on Access Control Models and Technologies, Innsbruck, Austria, pp.61-72, 2011.*

[11] P.W.L. Fong, "Relationship-based access control: protection model and policy language," *ACM Conference on Data and Application Security and Privacy, San Antonio, USA, pp.191-202, 2011.*

[12] P.Fong, I.Siahaan, "Relationship-based access control policies and their policy languages," *ACM Symposium on Access Control Models and Technologies, Innsbruck, Austria, pp.51-60, 2011.*

[13] Department of Defense, "JP3-13: Information Operations," Washington DC: US Government printing, 2006.

[14] C.Chin-Ling, and C.Chih-Yu, "A two-tier coordinated defense scheme against DDoS attacks," *International Conference on Computer Science and Service System, Nanjing, China, pp.148-151, 2011.*

[15] H.M.Sun, et al, "A Robust Defense Scheme to Resist Routing Attacks in Mobile Ad Hoc Networks," *World Congress in Applied Computing, Computer Science, and Computer Engineering, Kota Kinabalu, MALAYSIA, pp.58-65, 2011.*

[16] N.Damianou, N.Dulay, E.Lupu, et al, "The ponder policy specification language," *Proceedings of the International Workshop on Policies for Distributed Systems and Networks, Bristol, UK, pp.18-38, 2001.*

[17] K.Twidle, N.Dulay, E.Lupu, et al. "Ponder2: A policy system for autonomous pervasive environments," *Fifth International Conference on Autonomic and Autonomous Systems, Valencia, Spain, pp.330-335, 2009.*

[18] OASIS, "eXtensible access control markup language(XACML) Version 3.0," http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf , 2013.

[19] A.Herzberg,Y.Mass,J.Mihaeli,D.Naor, et al, "Access control meets public key infrastructure, or: assigning roles to strangers," *IEEE Symposium on Security and Privacy, Berkeley, America, pp.2-14,2000.*

[20] L.Kagal, "Rei: A Policy Language for the Me-Centric Project," http://ebiquity.umbc.edu/_file_directory_/papers/57.pdf, 2005.

[21] Z.Bin, A.S.Ehab, J.Radha, R.James, P.Corin, "Specifications of a high-level conflict-free firewall policy language for multi-domain networks," *Proceedings of the 12th ACM symposium on Access control models and technologies, Sophia Antipolis, France, pp.185 – 194, 2007.*

[22] L.P.Swiler, C.Phillips, D.Ellis, S.Chakerian, "Computer-attack graph generation tool," *Proceeding of DARPA Information Survivability Conference & Exposition II, Vol.2, pp.307-321, 2001.*

Zhao Wei, born in 1984, Ph. D. candidate, His main research interests include computer network and information security.

Chunhe Xia, born in 1965, professor, Ph. D. supervisor. His main research interests include computer network and information security, information operations and cloud computing.

Yang Luo, born in 1989, master candidate, His main research interests include network security and virtualization technology.

Xiaochen Liu, born in 1988, Ph. D. candidate. Her main research interests include network security and cloud computing.

Weikang Wu, born in 1987, Master. His research interest is network security technology.