A Novel Network Proactive Defense Model: Anomaly Driven Dynamic Cooperative Defense Model

Li Lixun[1,2], Zhang Bin[1,2], Wu Haoming[1,2]and Li Yingbo[1,2]

[1] Zhengzhou Information Science & Technology Institute, ZhengZhou, China
[2] Henan Province Information Security Key Laboratory, ZhengZhou, China

Corresponding author's e-mail address: lilixun_1994@126.com

Abstract. To defend the more complex and stealthy network attacks, an anomaly driven dynamic cooperative defense model (ADCD) is proposed in this paper. ADCD takes anomalies as defense drive factors to identify network attack types and evaluate system security state. And according to the anomaly processing result, ADCD can combine network cooperative defense (NCD) with moving target defense (MTD) to defend known and unknown network attacks. Firstly, detail explanations of ADCD is given, including its elements, operations, relations and rules. Then, a formal analysis of ADCD is conducted based on finite state machine (FSM) to prove its security. Finally, A typical application scenario where ADCD is realized as a mutiagent system is displayed to verify the effectiveness of ADCD in defending complex and multi-step attacks.

Key words. Network security, Proactive defense, Formal model, Intelligent agent, Multiagent system

1. Introduction

With the development of information technology, network security is becoming increasingly important. Network cooperative defense (NCD) refers to conducting targeted network security policy by collaborating security devices to detect, analyze and deal with various network attacks according to the security mission demands predetermined. [1] NCD is very efficient when attacks are detected, but it does not work well if attacks are too stealthy to be detected. To cope with the more complicated and secretive attacks, researchers proposed moving target defense technology (MTD), which aims at increasing attackers' difficulty and cost by reducing information system's definiteness, stability and isomorphism using dynamic mutations. [2,3] Now, lots of researches proving the advantages of MTD in defending attacks undetectable have been put forward, but MTD still faces two challenges in practice:

- How to collaborate NCD and MTD to realize complementary advantages for the better defense effect? Some of new MTD technologies such as the port hopping are based on dynamic mutations to reduce the success probability of attackers but not to protect information systems by eliminating vulnerabilities[4]. While existing NCD technologies such as the security operation center (SOC) and the unified threat management (UTM) are based on the intrusion detection and vulnerability mining to defend attacks[1,5]. These two different defense methods depend on their different foundations, so it is a challenge to create a cooperative mechanism between them.
- How to determine the key parameters such as the mutation object and the mutation period in

MTD according to the security state of information system protected? Most of MTD technologies focus on conducting dynamic mutations on specific system elements such as softwares or operation systems[6], but not to take flexible driven mutations according to system's security state to choose the best mutation parameter. This non-driven mutation mechanism is lack of flexibility and target when faced with complicated and stealthy attacks, and may increase the risk of been attack for the unintentional exposure of system vulnerabilities, as well as increasing the system's running cost[6]. So, considering the balance of security and stability, it is necessary to figure out how to take driven mutations.

As for the first challenge, both MTD and NCD are to defend network attacks, although they have different foundations. And most of network attacks, whether it is known or unknown, would cause anomalies on network traffic or host behavior. Therefore, it is worth exploring to create a new cooperative mechanism between MTD and NCD based on anomaly processing.

To deal with the second challenge, when the cooperative mechanism of MTD and NCD is constructed, the mutation parameters in MTD such as the mutation object and period could be determined by combining the anomaly processing result in NCD with mutation parameter choosing method in MTD.

Therefore, we propose a novel network proactive defense model called anomaly driven dynamic cooperative defense model (ADCD). ADCD takes anomaly as defense drive factor to identify network attack types and evaluate system's security state, and combines NCD with MTD technologies based on the anomaly processing result in the previous step to realize complementary advantages in defending known and unknown network attacks.

2. ADCD

2.1. Concepts

The basic idea of anomaly driven dynamic cooperative defense model (ADCD) is the combination of anomaly driven and policy response. The anomaly driven refers to that the inputs of ADCD are anomalies such as abnormal file operations or abnormal network communications detected by monitors. Then the anomalies would be processed by the misuse detection, the anomaly detection and the attack effect evaluation, in which the misuse detection and anomaly detection are to identify the attack types while the attack evaluation is to determine the degree of negative impact caused by attack. And these three processes could provide references for the next defense policy together. It is worth addressing that The detection efficiency and accuracy of ADCD can be improved by using the misuse detection and the anomaly detection at the same time.

The policy response refers to that ADCD would output defense policies as long as anomalies input into it, no matter whether the attack types causing anomalies are known or unknown. That is to say, ADCD would output precise policies such as the access control and anti-virus responding to known attacks, while it may also output MTD policies, which take the anomaly module such as the IP address and the software as the mutation object in response to unknown attacks.

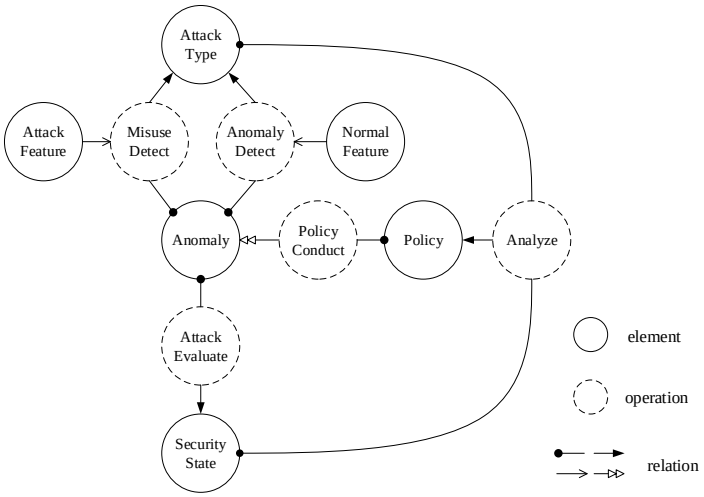

Figure 1 ADCD model

As shown in Figure 1, there are 6 kinds of elements, 5 kinds of operations and 4 kinds of relations in ADCD, and the followings are their explanations.

2.2. Elements

Anomaly (*A*). *A* refers to the system's anomalies detected in a period, $A=\{id,a_1,a_2,\ldots,a_n\}$, $n\in \mathrm{N}$ (N is natural number). *id* is the event number to mark anomalies detected in the same period; a_i $(0\le i\le n)$ is the single anomaly such as the abnormal file operation.

Attack feature (*AF*). *AF* refers to the attack feature which is already known, $AF=\{af_1,af_2,...,af_n\}$, $n\in \mathrm{N}$. af_i $(0\le i\le n)$ is the single attack feature such as a hash code of a known virus or an API calling sequence of a known malware.

Normal feature (*NF*). *NF* refers to the feature of modules of components in a security system, $NF=\{nf_1,nf_2,...,nf_n\}$, $n\in \mathrm{N}$. nf_i $(0\le i\le n)$ is the single normal feature such as an API calling sequence of a known security software or an operation log of a legitimate user.

Attack types (*AT*). $AT=\{id,at_1,at_2,...,at_n\}$, $n\in \mathrm{N}$. *id* is the event number same as in *A*. at_i $(0\le i\le n)$ is the single attack type such as the famous virus WannaCry or Stuxnet.

System security state (*S*). *S* reflects the degree of negative effect caused by attack, $S=\{id,value\}$. *id* is the event number same as in *A*. *value* is a quantitative value of security state calculated by attack effect evaluation.

Defense policy (*Pol*). $Pol=\{id,subject,target,action,constraint\}$. *id* is the event number same as in *A*. *subject* is the policy executor. *target* is the defense object. *action* is executor's action, *constraint* is the limitations of policy. Noting that the *action* in the *Pol* contains both NCD technologies and MTD technologies. In this paper, we choose XML to describe *Pol* for its universality. Here is an example:

```
<?xml version="1.0" standalone="yes"?>
<eventid>0002</eventid>
<policyid>0010</policyid>
<policyname>IP mutation</policyname>
<policybody>
  <subject>Router</subject>
  <target>Communication parameter</target>
  <action>IP Random Mutation</action>
  <constraint>
    <mutationtarget>IP address</mutationtarget>
    <mutationperiod>60s</mutationperiod>
```

```
    <mutationspaee>65536</mutationspaee>
  </constraint>
</policybody>
<decision>true</decision>
```

The meaning of the *Pol* above is: this IP mutation policy should be conducted by routers to take IP random mutation on communication parameters, and the mutation target is IP address, the mutation period is 60 seconds, the mutation space is a network address space of B class (the B class address space contains 65536 addresses).

2.3. Operations

Misuse detection (*MD*). *MD* is to fast detect known attack types by comparing the extracted features in anomalies detected with the *AF* stored in database.

Anomaly detection (*AD*). *AD* is to detect the stealthy attacks missed by the *MD*. In the *AD* process, the feature vector should be constructed according to the anomalies detected, and then running the anomaly detection algorithm based on the extracted feature vector and the *NF* stored in database. *AD* could detect known attacks and part of unknown attacks.

Attack effect evaluation (*AE*). *AE* is to evaluate the negative effect caused by attacks, and to get a quantitative result of attack effect, which is an important reference for the defense policy making.

Association analysis (*A&A*). *A&A* is to gather the results of *MD*, *AD* and *AE* so that we can conduct a further analysis to get the optimal dynamic defense policy. If the attack is detected as a known one, ADCD would output a precise defense policy to defend. While ADCD would output a MTD policy to defend if the attack is detected as an unknown one, and the mutation object should be the module or the component detected as abnormal (e.g. the file operation API should be the mutation object once some abnormal file operations are detected). As for the rest mutation parameters, we can combine the system security state with other MTD researches[7] to determine.

Policy conduct (*PC*). *PC* is to distribute the policies output from *A&A* to related defense subjects such as the firewall and anti-virus software for further implementation.

2.4. Relations

There are 4 basic relations between elements and operations.

Drive relation: An element can drive an operation to work, and the formal definition is: $Drive(\text{element} \rightarrow operation)$. There are 6 drive relations in ADCD: $Drive(A \rightarrow MD)$, $Drive(A \rightarrow AD)$, $Drive(A \rightarrow AE)$, $Drive(S \rightarrow A\&A)$, $Drive(AT \rightarrow A\&A)$, $Drive(Pol \rightarrow PC)$.

Support relation: An element is the support of an operation, in another word, an operation cannot work without an element. The formal definition is: $Support(element \rightarrow operation)$. There are 2 support relations in ADCD: $Support(\text{AF} \rightarrow \text{MD})$ and $Support(NF \rightarrow AD)$.

Output relation: An operation can output an element, and the formal definition is $Output(\text{operation} \rightarrow element)$. There are 4 output relations in ADCD: $Output(MD \rightarrow AT)$, $Output(AD \rightarrow AT)$, $Output(AE \rightarrow S)$, $Output(A\&A \rightarrow Pol)$.

Restrain and eliminate relation: The *PC* operation can restrain and eliminate the element *A*, in another word, ADCD can output and conduct a policy that can restrain and eliminate the *A* input into the model. There is only 1 *restrain and eliminate* relation in ADCD, whose formal definition is $RestrainandEliminate(PC \rightarrow A)$.

2.5. Rules

It is important to ensure the security of ADCD when the operations are running, so there are 6 rules in ADCD. The rules are described by the Datalog language[8], whose grammars are:

(1) RuleName: $\text{RH} \Leftarrow \text{RB}$

(2) $\text{RB}=\text{RE}_1$ op RE_2 op … op RE_n

RH is the header of a rule. RB is the body of a rule, and RB can consist of several rule expressions (RE) and operators (op). There are 3 kinds of operators: and ($\wedge$), or ($\vee$), not ($\neg$).

Rule 1 Attack type output rule. The *MD* and *AD* can determine attack types only with the drive of *A* and the support of *AF* or *NF*. The formal definition is: $Output(MD \to AT) \Leftarrow Drive(A \to MD) \wedge Support(AF \to MD)$; $Output(AD \to AT) \Leftarrow Drive(A \to AD) \wedge Support(NF \to AD)$.

Rule 2 System security state output rule. The *AE* can evaluate system's security state only with the drive of *A*. The formal definition is: $Output(AE \to S) \Leftarrow Drive(A \to AE)$.

Rule 3 Defense policy output rule. The *A&A* can output *Pol* only with the drive of *AT*, while the drive of *S* is optional. The formal definition is: $Output(A\&A \to Pol) \Leftarrow Drive(AT \to A\&A) \vee (Drive(AT \to A\&A) \wedge Drive(S \to A\&A))$.

Rule 4 Restrain and eliminate rule. The *PC* can restrain and eliminate *A* only when the *Pol* is distributed and conducted and the *id* of *Pol* should be the same as *A*'s. The formal definition is: $RestrainandEliminate(PC \to A) \Leftarrow Drive(Pol \to PC) \wedge Pol.id{=}A.id$.

Rule 5 Event stable rule. The *id* of an attack event causing the anomalies should be stable after the 5 operations of ADCD. The formal definition is: $AT.id{=}A.id \Leftarrow (Drive(A \to MD) \wedge Output(MD \to AT)) \vee (Drive(A \to AD) \wedge Output(AD \to AT))$; $S.id{=}A.id \Leftarrow Drive(A \to AE) \wedge Output(AE \to S)$; $Pol.id{=}AT.id{=}S.id \Leftarrow Drive(AT \to A\&A) \wedge Drive(S \to A\&A) \wedge Output(A\&A \to Pol)$; $A.id{=}Pol.id \Leftarrow Drive(Pol \to PC) \wedge RestrainandEliminate(PC \to A)$.

According to the rules above, we can further derive the following 4 theorems:

Theorem 1. If there is an *AT* and *AT*'s *id* is id', there must be an *A* and *A*'s *id* is also id'. The formal definition is: *if* $\exists AT$ and $AT.id = id'$, so $\exists A$, *s.t.* $A.id = id'$.

Proof. $\because \exists AT.id = id'$, and $\because AT.id{=}A.id \Leftarrow (Drive(A \to MD) \wedge Output(MD \to AT)) \vee (Drive(A \to AD) \wedge Output(AD \to AT))$, $\therefore \exists A$, *s.t.* $A.id = id'$.

Theorem 2. If there is a *S* and *S*'s *id* is id', there must be an *A* and *A*'s *id* is also id'. The formal definition is: *if* $\exists S$ and $S.id = id'$, so $\exists A$ *s.t.* $A.id = id'$.

Theorem 3. If there is a *Pol* and *Pol*'s *id* is id', there must be an *AT* and a *S*, and *AT*'s *id*, *S*'s *id* and *Pol*'s *id* is the same. The formal definition is: *if* $\exists Pol$ and $Pol.id = id'$, so $\exists AT$ and $\exists S$, *s.t.* $AT.id = S.id = id'$.

The proofs of theorem 2 and theorem 3 is similar, we haven't described them in order to avoid repetition in this paper.

Theorem 4. If there is an *A* and *A*'s *id* is id', there must be a *Pol* and *Pol*'s *id* is also id'. The formal definition is: *if* $\exists A$ and $A.id = id'$, so $\exists Pol$, *s.t.* $Pol.id = id'$。Noting that theorem 4 conforms to the ADCD's policy response concept that ADCD would output defense policies as long as anomalies input into it, no matter whether the attack types causing anomalies are known or unknown.

Proof. $\because \exists A.id = id'$, and $AT.id{=}A.id \Leftarrow (Drive(A \to MD) \wedge Output(MD \to AT)) \vee (Drive(A \to AD) \wedge Output(AD \to AT))$, and $S.id{=}A.id \Leftarrow Drive(A \to AE) \wedge Output(AE \to S)$, $\therefore \exists AT, S$, and $AT.id = S.id = id'$. $\because Pol.id{=}AT.id{=}S.id \Leftarrow Drive(AT \to A\&A) \wedge Drive(S \to A\&A) \wedge Generate(A\&A \to Pol)$, $\therefore \exists Pol$, $Pol.id = id'$.

Rule 6 Defense feedback rule. If the policy can't restrain or eliminate the anomalies and the defense feedback factor is still low than the threshold, the anomalies existed would drive the *MD*, *AD* and *AE* and make the defense feedback factor add 1. The formal definition is: $Drive(A \to MD) \wedge Drive(A \to AD) \wedge Drive(A \to AE) \wedge Feedbackfactor + 1 \Leftarrow \neg RestrainorEliminate(Pol \to A) \wedge Feedbackfactor < threshold$. Rule 6 creates a beneficial system feedback and also introduces a security mechanism avoiding a deadlock of ADCD. And if in the situation that the *Feedbackfactor* is larger than the threshold, we could adjust some modules in the model such as the anomaly detection algorithm or eliminate the anomalies manually.

3. Security Analysis

The state of ADCD will change with the conduction of its operations, and because of the finite operations and elements of ADCD, there are also finite states of it. Therefore, we carry out the security analysis of ADCD based on finite-state machine (FSM) as follows.

3.1. FSM of ADCD

3.1.1. State variables. ADCD is defined as a FSM tuple: M={V, I, T, v_0, v_n}. V={v_0, v_1, …, v_n}, which is the set of states that will transfer from one to another under the drive of transfer functions, and ADCD can only be in a certain state at any moment. I is the input set. T is the set of transfer functions, which represent the mapping relation of I and V and can be described as $V \times I \to V$. v_0 is the original state and v_n is the final state.

For any v_i belonging to *V*, v_i={*A*, *AT*, *S*, *Pol*}, and *A*, *AT*, *S*, *Pol* are anomaly, attack type, security state and policy. From the definition we know that v_i can describe the running situation of ADCD, i.e., whether there is anomaly input, whether the attack type is determined and attack effect is evaluated, and whether there is a dynamic defense policy output.

According to the definitions above, and combining the elements and operations of ADCD, we can get all of the states and inputs of ADCD as show in table 1 and table 2.

Table 1. States and inputs of ADCD

state	Explanation	Formalization
v_0	Anomaly input	$\{A, \varnothing, \varnothing, \varnothing\}$
v_1	Known attacks determined	$\{A, AT^k, \varnothing, \varnothing\}$
v_2	Unknown attacks determined	$\{A, AT^u, \varnothing, \varnothing\}$
v_3	Attacks and security state determined	$\{A, AT, S, \varnothing\}$
v_4	Defense policy output	$\{A, AT, S, Pol\}$
v_5	Anomaly eliminate	$\{\varnothing, \varnothing, \varnothing, \varnothing\}$

Note: AT^k refers to known attacks, AT^u refers to unknown attacks, $AT=AT^k \cup AT^u$.

Table 2. States and inputs of ADCD

input	explanation
i_0	Misuse detection
i_1	Anomaly detection
i_2	Attack effect evaluation
i_3	Association analysis
i_4	Policy conduction

3.1.2. Transfer function

T_0: $v_0+i_0 \to v_1/v_2$. Conduct the misuse detection after anomaly is input, which aims to determine attack types quickly based on the attack features.

T_1: $v_2+i_1 \to v_1/v_2$. Conduct a further anomaly detection if attack types can't be determined by misuse detection, and T_1 is based on normal features.

T_2: $v_1+i_2 \to v_3$. Take the attack effect evaluation after the attack being determined as known, to calculate the degree of negative influence caused by anomalies.

T_3: $v_2+i_2 \to v_3$. Take the attack effect evaluation after the attack being determined as unknown too, which is in order to provide references about mutation parameter determination based on the result of evaluation.

T_4: $v_3+i_3 \to v_4$. Conduct the association analysis according to the result of attack type determination and attack effect evaluation, to output precise defense policy.

T_5: $v_4+i_4 \to v_0/v_5$. Conduct the defense policy.

According to the 6 transfer functions above, we can get the state transfer graph as shown in Figure 2.

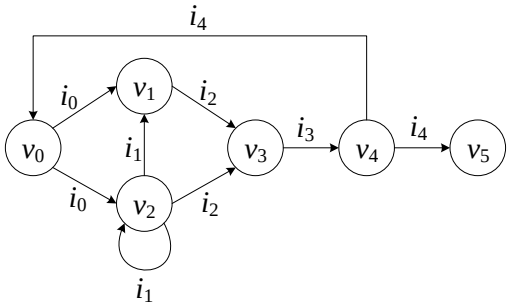

Figure 2. State transfer graph

3.2. Analysis based on FSM

3.2.1. Basic definition. **Secure state.** As for the state v_i={A, AT, S, Pol}, if the drive element (e.g. A is the drive element of AT) exists for any AT, S or Pol, and drive element's id is the same as its output element's id, so we call the state v_i is secure. According to the relevant theorems of ADCD, this secure state definition can be formalized as: $\forall v_i$={A, AT, S, Pol}, v_i is secure when $v_i \mapsto$ Theorem 1 $\wedge$ $v_i \mapsto$ Theorem 2 $\wedge$ $v_i \mapsto$ Theorem 3. The symbol $\mapsto$ means "satisfy".

It is obvious that the original state $v_0=\{A,\varnothing,\varnothing,\varnothing\}$ is secure according to the definition.

Secure transfer function. For any transfer function T_i, it is secure only when its state output is secure for any secure state input.

3.2.2. Analysis of ADCD. There are 6 transfer functions, T_0~T_5, in the FSM of ADCD. According to the operations and rules of ADCD, we know that T_0, $T_1 \mapsto$ Rule1, $T_2 \mapsto$ Rule2, $T_3 \mapsto$ Rule2, $T_4 \mapsto$ Rule3, $T_5 \mapsto$ Rule4 $\wedge$ Rule6. Based on the 6 rules, we can prove the security of these transfer functions. The security of T_0 and T_5 are proven in this paper for instance, the rest transfer functions' security proof is similar.

Proof of T_0's security. Assuming that the FSM of ADCD is in the original state $v_0=\{A,\varnothing,\varnothing,\varnothing\}$, and the A in v_0 is any kind of anomalies. Now we conduct the misuse operation, i.e., input i_0 into the FSM. According to Rule 1, AT will be output. So the FSM will go into state $v_1=\{A, AT^k,\varnothing,\varnothing\}$ or $v_2=\{A, AT^u,\varnothing,\varnothing\}$. And according to Rule 5, there is $AT.id=A.id$. Furthermore, we can draw the conclusion: as for v_1 and v_2, $v_1,v_2 \mapsto$ Theorem 1 $\wedge$ $v_1,v_2 \mapsto$ Theorem 2 $\wedge$ $v_1,v_2 \mapsto$ Theorem 3. Therefore, the state v_1 and v_2 are secure. According to the Secure transfer function definition, T_0 is secure.

Proof of T_5's security. Assuming that the FSM of ADCD is in the secure state $v_4=\{A, AT, S, Pol\}$. According to the secure state definition, there is an A existed, and $A.id = Pol.id$. Based on Rule 4 and Rule 6, the FSM will go from v_4 into $v_5=\{\varnothing,\varnothing,\varnothing,\varnothing\}$ or $v_0=\{A,\varnothing,\varnothing,\varnothing\}$. It is obvious that $v_0,v_5 \mapsto$ Theorem 1 $\wedge$ $v_0,v_5 \mapsto$ Theorem 2 $\wedge$ $v_0,v_5 \mapsto$ Theorem 3. Therefore, according to the Secure transfer function definition, T_5 is secure.

Based on the above analyses, the original state of ADCD is secure and every transfer function is secure too. According to the theory of FSM, it is proven that ADCD is secure.

4. Scenario

4.1. Implementation

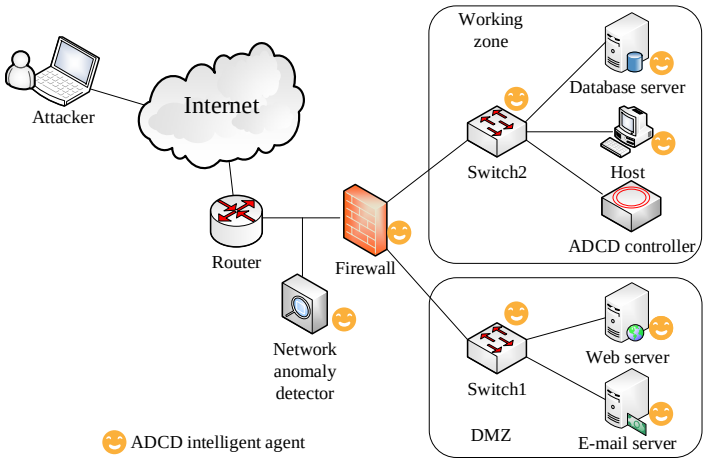

Figure 3 Implementation environment

As shown in Figure 3, it is a simulative enterprise network which is divided into demilitarized zone (DMZ) and working zone by the firewall security rules. There are 1 Web server and 1 E-mail server in DMZ, and 1 database server and 1 host in working zone.

According to the concept of ADCD, we make several implementations in the network as follows: 1) Install security softwares such as host firewall and host intrusion detectors on every device to detect anomalies and conduct defense actions; 2) Implement a network anomaly detector to detect abnormal traffics; 3) Deploy an ADCD controller in working zone; 4) Deploy ADCD intelligent agent on every device. Based on the implementations above, we create a multiagent system to realize ADCD. And the multiagent system is intelligent and ordered because of the rules defined in ADCD. What follows is the explanation for functions of ADCD controller and intelligent agent.

4.1.1. Functions of ADCD controller. The functions of ADCD controller are realizations of operations in ADCD, which including:

- Gathering all of the anomalies detected by devices, including abnormal file operations, abnormal network traffics, abnormal API callings and so on.
- Conducting misuse detection based on signatures of anomalies. This function is supported by some anti-virus softwares such as Kaspersky and Norton.
- Conducting anomaly detection based on anomalies and with the support of normal features. In this paper we choose the extended attack tree model[9] as the anomaly detection rules.
- Evaluating the effect of attacks to get the system security state. And we choose the multidimensional attack effect evaluation method[10] in this paper.
- Combining the attack type determined and system security state evaluated to conduct association analysis, and get the defense policy. For example, when the attack type is “Stuxnet virus”, the policy will be “block communications and delete the malware”. When the attack type is “unknown virus” and the anomaly is “abnormal network communications”, the policy will be “mutate device’s IP every 60 seconds in the B class address space”.
- Distributing defense policy to relevant ADCD intelligent agent to conduct.

4.1.2. Functions of ADCD intelligent agent. The functions of ADCD intelligent agent focus on assistance to ADCD controller such as follows:

- Detecting or collecting anomalies detected in devices and sending them to ADCD controller.
- Receiving policies distributed by the ADCD controller, and resolving the policies into the standard instructions of a certain device and then send them to devices to conduct.

Note that all the functions of ADCD intelligent agent should be automatic without administrator’s interference.

4.2. Defense process

We conduct a remote and multi-step attack as background to explain the defense process of ADCD. Firstly, we set a MS10_046 vulnerability on the working host, and a MS08_067 vulnerability on the database server. Then we use *Metasploit* (an attack testing platform) to attack as follow steps:

1) Use MS10_046 and DNS cheating to attack the working host and get a cmd shell, and then promote the privilege to system.
2) Take the working host as a springboard to conduct further attack to the database server using MS08_067 and get the administrator privilege.
3) Search classified files on the database server and send them back to attacker's computer, and also upload a Trojan to the database server as the backdoor.

In the above background, ADCD conducts following defense steps:

1) Firstly, the detector on working host captures abnormal communications launched by the device itself, and locates 2 processes, svchost.exe and rundll32.exe, by the process id. Then the detector on database server also detects abnormal communications and Windows register modification conducted by the device, and locates a process svchost.exe by the process id and a suspicious software Gh0st.exe by the record of Windows register. According to the FSM of ADCD, now the defense system is in state v_0. After the detection, the ADCD intelligent agents send these anomalies to ADCD controller.
2) ADCD controller conduct misuse detection on the anomalies received firstly, and find that the suspicious Gh0st.exe is a kind of Trojan. So the defense system goes into state v_1.
3) To determine what kind of types the attack causing the abnormal network communication is, the anomaly detection is conducted in ADCD controller. According to the detection rules proposed by [9], it can only be judged that some files are stolen by a remote computer, but the precise attack type is still unknown. So the defense system goes into state v_2.
4) Then, ADCD controller evaluate system's security state using the method of [10], and get a quantitative value. In the scenario of this paper, the quantitative value is 42.42. The defense system goes into state v_3.
5) Based on the result of attack detection and evaluation, ADCD controller conducts association analysis and outputs 3 defense policies, and the defense system goes into state v_4. The first policy is "delete Trojan":

```
<?xml version="1.0" standalone="yes"?>
<eventid>0010</eventid>
<policyid>0001</policyid>
<policyname>delete Trojan</policyname>
<policybody>
  <subject>databaseserver</subject>
  <target>Gh0st.exe</target>
  <action>delete</action>
</policybody>
<decision>true</decision>
```

The svchost.exe and rundll32.exe are system processes, so the attack can not be eliminated by deleting a malware. Therefore, the ADCD controller outputs the rest 2 policies: "IP block" and "IP mutation". The "IP block" policy is to cut off the current communications between attacker and victim host.

```
<?xml version="1.0" standalone="yes"?>
<eventid>0010</eventid>
<policyid>0002</policyid>
<policyname>IP block</policyname>
<policybody>
  <subject>firewall</subject>
  <target>10.100.200.11</target>
  <action>block</action>
</policybody>
```

```
<decision>true</decision>
```

The "IP mutation" policy is to decrease the probability of success for the attacks that may happen in the future. And the mutation parameters are determined according to the method of [7].

```
<?xml version="1.0" standalone="yes"?>
<eventid>0010</eventid>
<policyid>0003</policyid>
<policyname>IP mutation</policyname>
<policybody>
 <subject>Router</subject>
 <target>communicationparameter</target>
 <action>IP Random Mutation</action>
 <constraint>
  <mutationtarget>IP address</mutationtarget>
  <mutationperiod>60s</mutationperiod>
  <mutationspaee>65536</mutationspaee>
 </constraint>
</policybody>
<decision>true</decision>
```

6) ADCD controller sends the 3 policies above to the intelligent agents on Firewall and Switch2 to conduct. After these defense steps, anomalies are eliminated and the defense system goes into state v_5.

From the scenario above, we can find that there are two significant advantages of ADCD when attacks happen:

- ADCD can combine the traditional defense techniques such as anti-virus with the new moving target defense (MTD) techniques such as IP mutation cooperatively, to defense complex and multi-step attacks, and without changing the original network topology.
- Based on anomaly processing, ADCD can output targeted defense policies according to different attack types, no matter whether the attack is known or unknown. For example, ADCD can output a "delete Trojan" policy for known Trojans, or an "IP mutation" policy for unknown attacks that cause the communication anomalies.

5. Conclusion

A novel network proactive defense model called anomaly driven dynamic cooperative defense model (ADCD) is proposed and very detail explanations of ADCD's elements, operations, relations and rules are given. Furthermore, based on FSM, a formal analysis of ADCD is conducted to prove its security. Finally, A typical application scenario is displayed to verify the effectiveness of ADCD in defending complex and multistep attacks. We believe that ADCD can provide a new solution in defending network attacks.

Acknowledgments

This paper is supported by Henan Province Foundation and Frontier Technology Research Project (No. 142300413201), Fund for Zhengzhou Information Science & Technology Institute (No. 2016604703)

References

[1] Zhang T N, Yun X C, Zhang Y Z, Men C G, Sun J L. 2011 *J. Chinese Journal of Computers*, 34(2):216-228.
[2] Jajodia S, et al. 2011 *Moving Target Defense* (New York: Springer) 2011.
[3] Okhravi H, Rabe M A, Mayberry T J, Maybery T, Leonard W G, Hobson T, Bigelow D, Streilein,W W. 2013 *Survey of Cyber Moving Target Techniques (Electronic Materials)*.
[4] Lee H C J, Thing V L L. 2004 *Proc. Int. Conf.Vehicular Technology* (New York: IEEE), 5:3291-3295.
[5] Dwivedi S, Angeri H R, Arora V J. 2008 *Architecture for Unified Threat Management (Electronic Materials)*.

[6] CAI G L, WANG B S, WANG T Z, Luo Y B, Wang X F, Cui X W. 2016 *J. Journal of Computer Research and Development*, 53(5):968-987.
[7] Li N, Mitchell J C. 2003. *Lecture Notes in Computer Science*, 2562:58-73.
[8] Lei C, Ma D H, Zhang H Q, Yang Y J, Wang L M, 2017 *J. Chinese Journal of Computers*, Online Publishing 40(130).
[9] Wang L N, Tan C, Yu R W, Yin Z G. 2017 *J. Journal of Computer Research and Development*, 54(7):1537-1548.
[10] Zhan B F, Yin X C. 2011 *J. Computer Engineering and Design*, 32(8):2596-2599.