A Network Attack Model based on Colored Petri Net

Xinlei Li 1 and Di Li 2
1. Henan Normal University, Xinxiang, China
2. University of Liverpool, UK
Email: lxli@163.com, carlli2003@msn.com

Abstract—The researches have shown that not all the Petri Net machines can be used to describe attack behavior. When Petri Net machines adapted for attack behavior modeling are detecting the network, for some event of current status, if there is matching event in the model, it has only one corresponding transition; otherwise that may cause errors. Since sharing synthesis and synchronization synthesis of traditional machines cannot ensure synthetic model reserves original detection capability, we propose the novel concept for synthesis operation and colored synthetic operation. By the analysis on the relation among these operations, the ability to reserve original detection is verified. Then an improved colored judgement Petri Net machine is adopted for modeling and renewing the knowledge repository. The inductive learning method is used to extend the attack modes. It creates a four-layered concept space, which actually provides a depth-first search path for matching. To solve the problems in multi-pattern matching and incremental learning, various modes are generalized by colored operation. We also adopt the decomposition and synthesis operation to handle the pattern matching of distributed attack behavior and attack information fusion. Finally the actual cases verify that our algorithm is feasible.

Index Terms—Petri Net Machine; Concept Space; Synthesis; Attack Behavior; Generation

I. INTRODUCTION

Petri Net is a graphical and digital modeling tool and it has rigorous mathematical basis, as well as graphical modeling capability. It is accurately appropriate to describe causality, independence, parallelism and dynamism among events in the system and it can also verify the performance of simulate systematic in advance. Therefore, it is broadly applied in modeling and analysis on discrete dynamic event systems, including flexible manufacturing system, distributed software system, and industrial control system, etc. When complicated system is modeling, more library nodes and transition nodes will appear to increase the complexity of modeling and analysis [1]. Colored Petri net (CPN for short) is used to simplify a high-leveled Petri Net in Petri Net model structure. It merges the library nodes with similar behavior characteristics in the system into one library node and distinguishes different colors of token, to simplify Petri Net structure. Kumar [2] is the earliest one to propose CPN for pattern matching. He establishes a network intrusion detection system prototype IDIOT based on CPN and the research shows that CPN can be used to describe complicated attack. Mark Slagell [3] proposes a distributed CPN implementation of state IDs in MAIDS. At present, some domestic scholars discuss above methods and propose many kinds of systematic schemes [4-6].

Petri Net not only has direct graph and structural hierarchy but it also has a set of mathematical theory methods to support systematic analysis. The research [7] shows that Petri Net has its special advantages, as a tool for analyzing aggressive behavior. Reference [8] studies the relationship between multi-leveled Petri Net and modeling logic. References [11, 12] study the relationship between Petri Net and Horn clause and propose four new inference algorithms to find T-invariant which contains the goal transition. All these algorithms derive from the refinement strategy of resolution refutation system and have high efficiency. Therefore, they all propose effective methods to deduce Petri Net model of logic inference [13, 14]. Thus, under unified Petri Net representation models, some knowledge representation methods can be organically combined to make up knowledge representation in broader field. It is feasible for Petri Net to represent knowledge and it is adapted for many conditions.

This paper aims to study the knowledge representation and intelligent analysis on aggressive behavior. Petri Net theory and artificial intelligence technology are applied to set up related theory and method, which are suitable for aggressive behavior analysis and detection. Some subtype of Petri Net judgment machine is taken as the description model of aggressive behavior. The operation mechanism is used to detect aggression and that of colored judgment Petri Net machine is used to solve the problem in mode relevance. Meanwhile, inductive learning is applied to update and expand knowledge database. The rest of the paper is organized as the follows: Sector 2 proposes a modeling subnet suitable for aggressive behavior, that is, the concept of colored judgment Petri Net machine. Since sharing synthesis and synchronization synthesis process of traditional Petri Net cannot ensure that the synthesis model reserves the detection capability of original model, the definition of sharing synthesis operation and colored synthesis operation on judgment Petri Net machine are proposed to analyze the relationship among these synthesis process. Sector 3 uses colored synthesis...
operation of colored judgment Petri Net to synthesize the aggressive model and make the association of models. Synthesis operation is performed at corresponding level in the concept space. In addition, extended range rule in inductive learning is applied to synthesize security event of transition bindings, to reduce the scale. Sector 4 provides a detection method to describe aggressive behavior using colored Petri Net machine. Our cases only involve searching and matching operations of strings. From colored synthesis process of CPN, if one aggressive model includes just one incident, this incident can be bound to one transition. Then the hierarchical concept space can remove mismatch events as soon as possible, which provides convenience for searching match.

II. RELATED WORKS

A. Analysis on the Nature of Judgment Machine

The judgement machine requires that the event should correspond to the transition one by one. The reason is to acquire easy detection computation and colored synthesis operation with CPN. In the definition of CPN, the concept of token split is introduced. It is different from traditional colored Petri Net and it is to ensure the behavior consistency of colored operation. When describing the Petri Net model of attack, library P is used to represent the condition for transition occurrence; the token of P denotes the attack example, transition T denotes the attack behavior by assigned events, arc F denotes the conditions for transition occurrence. In the IDIOT system, we can directly use CPN for modeling. For convenience, we choose concurrent expression as the transition measure and any sequencing concurrent expression can be transformed into a Petri Net model.

The transition occurred rules of colored judgement machine is: initially, the color of token is weighted by \( I_N \), whose color is the subset of its library. A transition should satisfy the demand of traditional Petri Net, that is, \( \forall p \in \gamma \geq 1 \), and it also requires the intersection, between token color in the library and in the transition is not null. After transition, the generated token color is the intersection of input transition and transition event. The transition will cause inner split of transition input set: if \( p \in \gamma \) and \( N_o(t) \subseteq P(t) \), p is split into \( p_1 \), \( p_2 \), \( p_1 \leftarrow N_o(t) \), \( p_2 = P(t) - N_o(t) \). \( p_1 \) is transited to input set and \( p_2 \) keeps unchanged.

The judgement machine can be seen as the case that colored judgement Petri Net only has one color. If token is output to the next status by transition, it can carry event information of t and the subsequent transitions will acquire this information from token; if multiple token pass the Fuse Operator, the information in token can be fused, which means a union of the information. Kumar uses standard CPN conception in [15]; he defines the needed variables for each transition as global variables in one model, which are called color and can be carried by the token. During the process they can be used by transition, as is different with the method in this paper. In our model, one model can represent multiple attack behaviors and we do not need to define the global variables.

B. Synthesis of CPN Machine

Synthesis of traditional Petri Net can not meet the demand for behavior description. The shared synthesis operation is actually the synthesis of status information in system. This paper proposes the synthesis operation of CPN, to unify the attack behavior model which has the same transition and to make the results keep the detecting ability of original model. First we provide the definition of synthesis process for CPN machine.

\[ PN_i = (P_i, T_i, F_i, C_i, P_0, F_0, H_i, M_i, \Sigma_i, h_i, G_i) \]

are two CPN machines. \( P_0, P_i \) are the input set and output set of \( P_i \), \( h(T_i) \cap h(T_j) \neq F \), \( \Delta_p = h(T_i) \cap h(T_j) \), \( C_i \cap C_j = \phi \). \( PN \) is the colored synthesis result of \( PN_i \) and \( PN_j \). Then:

\[ P = P_1 \cup P_2 - \Delta_p, \]
\[ \Delta_p = \{ p | p \in P_1, \exists t \in \Delta_p, p \in \gamma, p \cap \gamma \mid P_1 \} ; \]
\[ O_p = \{ p | p \in P_2, \exists t \in \Delta_p, p \cap \gamma \mid P_2 \}; \]
\[ p \in \gamma \neq t \neq \gamma \mid P_1 \]
\[ \gamma \mid t \mid P_1 \]

To \( T_1 \cup T_2 \):

\[ M_o(p) = \begin{cases} M_{o1}(p) & p \in P_1 ; \\ M_{o2}(p) & p \in P_2 ; \end{cases} \]
\[ C = C_1 \cap C_2 ; \]
\[ h(t) = \begin{cases} h_i(t) & t \in T_1 ; \\ h_j(t) & t \in T_2 ; \end{cases} \]
\[ G_j = G_{j1} \cup G_{j2} ; \]

When the above conditions are satisfied, \( PN \) is the colored synthesis of \( PN_i \) and \( PN_j \), recorded as

\[ PN = PN_i \cup PN_j \]

CPN can keep \( T \) unchanged during the synthesis and merge the input library of \( \Delta \). The other transitions also keep unchanged in this process.

III. CPN MODEL OF ATTACK BEHAVIOR

A. Attack Modeling based of Colored Judgement

We emphasize on the analysis and detection of Petri Net machine, to provide a knowledge representation for intrusion detection. CPN system can define attack behavior by a series of detected basic events. A basic
event means a characteristic behavior which is related to attack. Then these events are represented as a CPN model according to the occurring time or causality. The intrusion detection prototype IDIOT is established by Kumar based on CPN. For example, in IDIOT, the attack mode to identify SendMail attack is described as figure 1. It clearly describes the uncertain relation between cp and touch, to verify that CPN can describe complicated attack. In the implementation of IDIOT, CPN is represented by some description language and it has an explainer in itself. It is in charge of transforming the model to C++ class and turning it to match engine.

![Diagram of CPN model for SendMail attack](image)

**Figure 1.** A model case for SendMail attack

During detection, observed incident is matched with the model in network. Successful match indicates that incident binding transition can be activated. If all input library of transition has a token, the state of Petri Net should be corrected according to Petri Net transition and the token will be output from current library. If one token is acquired on final position, it indicates one attack occurs. In figure 1, if current state in the system is shown as this figure and incident 3 of binding ts occurs, token in s4 will be transferred from s4 to s5. If incident 4 of binding occurs, since there is only s5 which contains token in library s1 and s2 of t1, t4 will not be transited and it must wait until t1 and t4 occur. After s1 acquires the token, if incident 4 of t4 binding occurs, t4 can be activated. Final state pf can get a token to detect attack.

System will be based on attack instance to express concurrent expression and turn it into judgment Petri Net machine model. With machine learning and inductive learning, to get definition space of aggressive behavior, the system can handle unknown attack pattern to some extent so as to update and expand the attack knowledge library. Colored synthesis operation of CPN machine can synthesize many models to handle multi-pattern matching, so system can be efficient in detection when increasing models. Although colored judgment Petri Net machine can be modeled for system, the transition from practical system to CPN machine need more corresponding modeling knowledge. We use sequencing concurrent expression as the transition measures. Since any one sequencing concurrent expression can be converted into a judgment Petri Net machine model, the transition of sequencing concurrent expression description and natural language will be simple.

**B. Syncretic Algorithm for Attack Behavior**

If each attack behavior established a model, each model needs to be matched in order to verify one incident during detection. With increasing model, the detection efficiency will be reduced gradually. This is the most serious disadvantage when applying mode matching algorithms. This paper adopts synthetic operation of Petri Net machine to directly deal with the model. If two attack behaviors have the same incidents, that is, corresponding Petri Net machine model has the same transition. The discussion in last sector shows there are two kinds of synthetically operations of Petri Net machine with the same transition. One type is concurrent operation while the other one is colored synchronous operation. Transitional occurrence after synchronous synthesis operation demands for that these two models can match on this synchronism. Obviously this operation does not satisfy our requirement. Colored synthesis operation can reserve the detecting capability of original model. In addition, it can satisfy the requirements since it cannot influence original behavior. We use colored synthesis operation in colored judgment Petri Net machine to synthesize various models with the same incident description into one model. This can detect many attack behaviors by searching match of this model. Moreover, for the same attack behavior, its attack mode is usually changing. When a new variation behavior appears, this variation behavior can synthesize original system by unification, to realize incremental learning method.

Assume $A, B$ are different attack behavior cases, $\alpha_i, \beta_i, i=1,2,3,4$ are concept space acquired by previous inductive learning algorithm. Now, we will unify the concept space formed by $A, B$. The unification is performed by their concept space with the same hierarchy. We set the result of $A, B$ unification as $C$. Then we can provide the unification algorithm steps of each hierarchy.

1. **Generalization algorithm of concept space in the first layer:** from previous generalization algorithm, there are all variables in incident result area of $\alpha_i, \beta_i$. There may be constant in object domain.

   At first, the incident is unified. If character, type and object area of the incident is the same, these kinds of incidents are similar incidents. During unification, the range of variables of result region variable in $A, B$ only needs to be merged. Extended range rules are applied for generation and different incidents are reserved. Then the event relation is unified: if the relation of two similar event is inconsistent in $\alpha_i, \beta_i$, then we adopt conjunctive rules and inductive rules for generalization. Its generalized sequence is "POWER|"FOLLOW"|"AND"|"OR". There is no relation for the events which are different in $\alpha_i, \beta_i$. The event color after synthesis is a color composite set of $\alpha_i, \beta_i$. Its color will keep original color of $\gamma_i$, which corresponds to the synthetic CPN machine model of $\alpha_i, \beta_i$. The algorithm process is shown as figure 2.

2. **The generalization algorithm of concept space in the second layer:** from previous generalization algorithms, we can know that $\alpha_i, \beta_i$ are all variables in object region and result region. If character, type, range’s composite set
of object or result domain of incident is not null, this incident is called similar incident. During unification, range in \( \alpha, \beta \) can be synthesized in object domain and result domain with expented range rule. Different events are reserved for relationship unification. If two similar incident relationship in the same sub-determinant is different in \( \alpha, \beta \), there is not relationship between different incidents in \( \alpha, \beta \). CPN machine model of \( \gamma \) equals to the colored synthesis of CPN machine model of \( \alpha, \beta \).

(3) Generalization algorithm of concept space in the third layer: from previous generalization algorithms, \( \alpha, \beta \) are all variables in object region and result region without considering the range. At first, the events are unified. If event character and operation type are the same, this incident is a similar incident. They can be synthesized as one during unification. Different events are reserved and the relationship will be unified. If the relationship between two similar events is different in \( \alpha, \beta \), conjunctive rules and inductive rules will be adopted for generalization. Default relationship in \( \alpha, \beta \) is synchronous. Their CPN models correspond to colored synthesis of colored judgment Petri Net machine. That is, the CPN machine model equals to colored synthesis of colored judgment Petri Net machine model in \( \alpha, \beta \). The algorithm is similar to the unification algorithm in the second layer. The target and range of result region in the third layer event are entire space, so the unification function is relatively simple.

(4) Generalization algorithm of concept space in the fourth layer: from previous generalization algorithms, there is only incident type in \( \alpha, \beta \). As long as event type is the same, the events will be considered to be one. When they are unified, they can be unified as only one and different events can be reserved. If the relation between two events is different in \( \alpha, \beta \), conjunctive rules and inductive rules will be adopted for generalization. Colored judgment Petri Net machine model of \( \gamma \) equals to \( \alpha \), that of \( \beta \) equals to colored synthesis of CPN machine model. Since unification algorithm corresponds to colored synthesis operation of CPN machine, it can be known from colored synthesis operation quality that the model after synthesis has reserved the detecting capability of original model.

C. Decomposition and Synthesis of Attack Model

On the basis of complicated attack behavior, many targets are usually involved and the detection needs multi-target cooperation. It needs decomposition of attack description model. Detected information of each target is a sub-part of attack model. During detection, these sub-parts needs to be synthesized. There are two cases: one is the attack behavior is distributed attack. The single target of distributed attack detection can not be completed and it needs combination of many targets; the other one is attack behavior is not distributed and one target can
detect its attack behavior. If multiple targets are combined, the detection will be more efficient. For distributed attack, attack model can be decomposed according to different targets. The joint operation and synthesis operation of judgment Petri Net machine is mainly implemented. Total attack model is made up of parallel connection, series connection or synthesis of each target sub-model. During detection it usually follows reversion of information flow from target node of attack, to gradually accumulate the attack root node. Root node fuses all the information to determine the attack behavior.

[17] Provides a case of springboard attack. As is shown in figure 3, by external access of Linux RHEL6.0 machine as the springboard, the attacker attacks port 39 of an internal Windows2003 server to obtain control rights. In principle, this machine cannot be accessed by external machine. However, due to the springboard, the attack is successful. Table 1 shows a general description and its attack model of this springboard attack.

![Figure 3. Case of springboard attack](image)

If Attack target wants to detect this kind of attack, it will test whether the sending packet and receiving packet are similar or not, which is unrealistic for servers. Usually, it needs to wait for information from Victim target and inquire the data package based on information of Victim target.

**TABLE I. DESCRIPTION OF SPRINGBOARD ATTACK**

<table>
<thead>
<tr>
<th>Attack &quot;Springboard&quot; [Netmon]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netmon</td>
</tr>
<tr>
<td>Server: listen= $LISTEN_PORT</td>
</tr>
<tr>
<td>#define N1</td>
</tr>
<tr>
<td>Recv: ip=$IPADD LocalPort=$LISTEN_PORT IPADD IN [EXTERNAL] message =$ MESSAGE</td>
</tr>
<tr>
<td>#define N2</td>
</tr>
<tr>
<td>Send: ip=$IPADD port=139 IPADD IN [INTERNAL] messages=MESSAGE</td>
</tr>
<tr>
<td>#define N3</td>
</tr>
<tr>
<td>Victim</td>
</tr>
<tr>
<td>#define N4</td>
</tr>
<tr>
<td>Recv: sourceip=$SOURCE_IP localport=139 SOURCE_IP_IN [INTERNAL]</td>
</tr>
<tr>
<td>#define N5</td>
</tr>
<tr>
<td>Relations=FOLLOW(AND(N1, FOLLOW(N2, N3 , AND(N4, N5)))</td>
</tr>
</tbody>
</table>

Actually, springboard attack model can be formed by series between attack model of Attack target and the attack model of Victim target. As is shown in figure 4. This attack detection is caused by occurrence from Victim target detection to N4 as N5. Then detected event is transmitted by SOURCE_IP of N5 marked target. Attack target will inquire whether N1, N2, N3 will appear from local monitoring information based on received information. If it is, complete model of springboard attack can be established for attack detection.

For undistributed attack behavior, sometimes the detection needs to obtain other targets’ support. As for Blaster attack, in W32.Blaster@mm, the sub-net N1, N2, N3, N4 describe the transmission behavior of broadcast. Infected object is also attacker Attack. When it attacks other targets, Victim might find out and stop its attack behavior, so that incident N4 cannot be detected. This sub-net cannot reach the final state and Blaster will not be detected. If Victim sends its attack behavior incidents N5, N6, N7 to the attacker, the attacker can infer the occurrence of local N4, and detect the Blaster further.

**IV. CASE STUDY**

We have used generalization algorithm to obtain the conception space for colored judgment Petri Net machine description. From up to bottom, this space is a process from general to special. Then, unification operation will diffuse all correlated attack behaviors. When some attack behavior occurs, whether it is actual attack or not, the essence is seeking the most approximate pattern of this behavior. When there are many known attack patterns, this operation and conception space provides a deep priority path for search and it can find out appropriate mode more rapidly for searching module. This sector presents systematic composition characteristics with CPN machine intrusion detection system and attack detection algorithm. In addition, the worm attack is taken as an example to illustrate the algorithm application under Windows system.

**A. Monitoring Example**

The follows provide a case for system detection of W32.netsky.P@mm registry. The process is `document.txt` and `Id = 1700`. Figure 5 provides the monitor record in registry monitoring system. We use “HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion” and “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion” to denote “HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion”. Each row is separated by blanks. The first row denotes the serial number of monitor records; the second row shows the time of event occurrence; the third shows the process ID; the fourth row denotes operation; the fifth denotes operation object; the sixth shows if operation is success and the seventh shows the results.

Each record in operation record corresponds to one security incident. All the event types of registry tables monitoring are RegMon. Its detecting process is:

1. Execute the first step of matching algorithm, that is, searching the first layer of concept space. Judge whether there is a case for the main event of “document.txt:1700”, to establish a model case first. The status of this case is
the initial status of model and the type RegMon get successful match. Then the second step is executed. If event operation Open fails in match, return to the first step and execute the sub-program of transition. At that time only the first layer of model case is changed. The state of the first-layered Petri Net is shown as figure 5.

(2) Event “14673 489.58189798 document.txt: 1700 SetValue HKLM\SOFTWARE\WinNT\Protect.exe” executes the matching algorithm to the fourth layer. When matching the result area, the range of model is ["service=c:\winnt\services.exe-serv", ICQ Net=c:\winnt \"winlogon.exe -stealth", “windows auto update=mbslast. exe"]. While the result of this event is “Norton Antivirus AV=C:\WINNT\Protect.exe”, so the system may think it is a distortion of attack and the character of case modification is generalization. Then the transition sub-program will be executed by Petri Net machine. Figure 6 shows the local graph of case model in the fourth conception space after match, the rest part is left out.

(3) The following two events will be eliminated due to failed event match. Its subsequent event "146377 489.58246816 document.txt: 1700 DeleteValueKey HKLM\SOFTWARE\Run\Explorer" will be executed to the fourth layer and gets successful match. But the character of the case is generation so system will stop to execute match in the fifth layer.

(4) The following two events still will be eliminated due to failed event match. Similarly, its subsequent event "146380489.58280983 document.txt: 1700 DeleteValueKey HKCU\SOFTWARE\Explorer\Run\Reg swinger" will be executed to the fourth layer and the system will stop to execute match in the fifth layer.

(5) The following three events are eliminated due to failed event match. The process of event “46386 489.58433991 document.txt: 1700 DeleteKey HKCR\ICLSID\E6FBSE20- DE35-IICF-9C87-OOAA005127ED\InProcServer32” is the same with the above.

Since the next transition for concept space in the fourth layer is Fuse operator, system will make execution automatically and acquires a token at Final. Once the match is successful the system will send alarm to recognize the Netsky worm.

B. Attack Detection Results

We test 20 trojan or worm virus emerged recently with the method in this paper, including W32.Netsky.C@m, Worm_Mimail.C, Win32.Mydoom.AD, BackDoor-AZV_gen, Backdoor/Beast.207Trojan/PSW.QQjb, Worm_Bbeagle.J, TrojanSpy.Banker.s, Worm/DocKiller, Worm/Mabutu, Worm/Pikis.b, I-Worm.Goner, BackdoorBouncer, TrojanSpy.Elelist, Worm/QQmsg.Lee, Win32.Troj.QQmsgSupnot, W32.Cervivec.A@mm, W32.Beagle.AX@mm, W32/Spybot.BAT, Backdoor.IRC.Bot. These programs contain operation of files or registry and 95% programs will execute similar behaviors, such as adding files to “C:\WinNT\System32” or “C:\WinNT”, modifying registry whose key is “HKLM\Software\Microsoft\Windows\Policies\Explorer\Run” or “HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices”. For multimode test we use two modes for matching one by one and there is no relation among these modes; the other adopts mode synthesis algorithm to test the unified model.

The test time includes the moment to read a monitoring record and describe the event. Through the match of 2400 Registry monitoring records and files monitoring records, figure 8 shows the results of multi-mode match, single-mode match and syncretic-mode match. We can see that with the number increasing the time needed by single-mode matching is increasing obviously; while the time of syncretic-mode is relatively small.

Then we test the different numbers of events, including single-mode test, one-by-one matching test of 20 modes, syncretic-mode matching of 20 modes. The results are shown in figure 9. With the development of event number, the time for single-mode matching increases rapidly, which is the opposite case for the other two methods. Time consumption of syncretic-mode matching is almost consistent with the single-mode, and the time difference is less than 0.01 second.

Kumar uses the sentences of compiler theory in IDIOT to optimize the mode. Reference [2] has provided the testing results in multi-mode matching. We make match
Figure 6. Monitoring records of the registry

Figure 7. Matching graph of the fourth layered space
behavior in Petri Net language, which equals to PN machine identification formed by aggressive behavior. First, we analyze and prove the behavior relationship theorem of colored synthesis operation. Colored synthesis operation of CPN machine can operate many aggressive models in unification, but it will not affect detecting capability of original model. Then, the colored judgment Petri Net machine is used to represent aggressive behavior and inductive learning method is also applied to realize automatic updating and extension. The colored synthesis operation of CPN machine can solve the problems in multi-mode matching and incremental learning. The joint operation of CPN machine can acquire the specialization, decomposition and synthesis of the model. Finally, specific cases are implemented to verify our scheme in describing detection methods.

V. CONCLUSION

Traditional CPN has problems in that aggressive modeling is not efficient to perform multi-model matching and it cannot detect aggressive transformation. Thus, this paper studies modeling match of aggressive behavior in the logs in RHEL 6.0 and it is described in the figure 10. The time needed to transform the log into event description mode is 5.17s, single-mode matching needs 5.45s for 2514 events. For 19 modes the test needs 5.91s. The time span from single-mode to multi-mode is bigger, about 0.45s. The algorithm has relatively large change, as seen from the curve trend.

Limited by samples, we only perform test for the generated results of W32.Netsky.B@mm. By the model generation we can detect the deformation of Netsky such as W32.Netsky.C@mm, W32.Netsky.D@mm, W32.Netsky.P@mm and W32.Netsky.X@mm. So our strategy is verified to be successful.

REFERENCES


Xinlei Li, Network Engineer of Henan Normal University. Xinlei Li born in Henan China, he received Master of Engineering degree in Information network of laboratories at Zhengzhou University in 2011. His current major field of study includes network security and computer application technology.

Di Li was born in Chengdu China. He received his PhD. degree and M.Sc degree in Electrical Engineering at the University of Liverpool, the UK in 2010 and 2005, respectively. His current major field of study includes wireless networks, radio propagation and antennas.