Dynamic Deploying Distributed Low-interaction Honeynet

Haifeng Wang[1 2]
1.University of ShangHai for Science and Technology
Shanghai, School of Management, Shanghai, China
2.Lin Yi University, Linyi, China
Email: gadfly7@126.com

Qinkui Chen
University of Shanghai for Science and Technology, School of Optical-Electrical
and Computer Engineering Shanghai, China
Email: chenqingkui@gmail.com

***Abstract*—Distributed virtual honeynet is an important security detection system to Worms, Botnet detection, Spam and Distributed Denial-Of-Service. The honeynet value significantly relies on the disguise capacity. The traditional deploying method is a static scheme that the configuration of honeynet is determined by security experts beforehand and unable to change after the deployment. The hackers or Botnet controllers identify the honeynet and may not trap into the same honeynet again. Therefore, the static deploying honeynet has relatively poor disguise capacity. To improve the disguise capacity, a novel dynamic deploying method is proposed that is capable of redeploying the honeynet in real time. The inducing degree is introduced to measure the disguise capacity by analyzing the inbound and outbound packets of the honeynet. When the inducing degree is less than a specific threshold, the dynamic deploying manager will be activated and to execuate the dynamic deploying algorithms. We have developed three novel dynamic deploying algorithms to solve the problem how to redeploy the honeynet and implemented a prototype for distributed virtual honeynet based on Honeyd. The experimental results of the simulation and real networks datasets demonstrate that the dynamic deploying approach is effective to enhance the disguise capacity of honeynet.**

***Index Terms*—Distributed virtual honeynet, Dynamic Deployment, Honeyd, Cooperative Learning Algorithms, Evolvement Algorithms**

I. INTRODUCTION

A honeynet is an effective information system whose value lies in its unauthorized or illicit use by hackers or worms, which aids the security experts get to known from attacking entities and thus improve existing security architectures and systems. Deploying a physical honeypot is time intensive and expensive process. Provos presents honeyd which is a framework for virtual honeypots that can simulate virtual computer or networks at the network level[1]. This honeypot software makes large-scale honeynet monitoring possible. Since then, there has been much research in the distributed honeynet to warn or analyze the new internet threats like worms or automated attacks. Bailey et al. presented a hybrid and distributed honeynet which combined the low-interaction honeypots and high-interaction honeypots to capture detailed attack traffic [2]. Potemkin is a large-scale honeynet prototype capable of obtaining high-fidelity attack data. And several effective methods were used to handle the tradeoff between the fidelity and the scalability of honeynet. It can emulate over 64,000 honeypots only using a handful of physical servers [3].

Honeynets have been widely used in network security to collect the information of attacks and detect the worms, Botnet, and Spam. Active honeynet was presented to defend against divide-conquer-scanning worms. Especially uniformly distributed active honeynets are more practical and effective to defend against the worms [4]. Tang presented double-honeynet to obtain the position-aware distribution signatures to identify polymorphic worms from the normal background traffic [5]. The honeynet was used to collect attack traffic to automatically generate signatures for zero-day polymorphic worms at their early phase [6]. Since the honeypots can attract Botnet compromises and widely used in Botnet defence [7, 8]. The honeynet as detection sensor was deployed based on Honeyd responders. Using the massive honeynet traffic reveals some interesting and sophisticated scanning behaviors in Botnet [8]. In addition, the honeynet has been used to as a aiding tool to extract intrusion signatures for Intrusion Detection System (IDS) [9 -11].

The Anti-Honeynet technologies appeared in response to the threat posing to the hackers [12]. Zu et al. first systematically focused on the honeypot detection in Botnet detection field and presented some guidelines for defending against general honeypot-aware attacks [7]. Thus, the current honeynet technologies are detectable. When the honeynet is identify by the hackers or the Botnet controllers. It will become worthless because it can not tempt attackers and collect any information about the intrusions. So the disguise capacity of honeynet is important to the applications of honeynet. To address the issue, the research focused on the service design in single honeypot to improve the fidelity of honeypot. Neal

doi:10.4304/jcp.7.3.692-698

argued that it is the effectively way to enhance honeynet disguise capacity by improving the simulation services complexity [13]. Although this approach is useful, the resulting is far from optimal and improves the design complexity of honeypot. Few researcher focus on the way to improve the disguise capacity from the whole honeynet. The honeynet is deployed and no longer change its configuration in traditional deployment scheme. So after the hackers identify the honeynet, they will not come back again. If the honeynet can adjust its configuration in real time. The hackers who has trapped in this honeynet may be induced once more.

Based on this principle, we present a novel honeynet deployment technique, which is simple but effective to distributed, virtual, low-interaction honeynet. Compared with the traditional static deployment technique, the novel technique is dynamic deploying scheme. When the honeynet redeploys again, it becomes new honeynet and must spend more time to detect it. To our knowledge, this is the first paper to systematically discuss dynamic deployment honeynet that could improve the disguise capacity of honeynet. Although Kuwatly developed a dynamic honeypot to identify attacks [14]. However, the honeypot still needed adjust manually during the deploying process and he concentrated on the single honeypot to dynamic deploying rather than the whole honeynet. In term of the honeynet deploying, Chang et al proposed the virtual honeynet collaboration system to improve the design and concepts of the honeynet architectures. This study presented detailed information about the honeynet deploying invloved the hardware and software techniques. But its deployment scheme was still static scheme [15].

The rest of this paper is organized as follows. Section II presents the dynamic deploying scheme for the honeynet and the dynamic deploying algorithms in details. In section □, we conduct simulation experiments to determine the threshold that controls when to redeploying. And deploying the honeynet in real networks shows the dynamic deploying method is more effective than the static one. In the end we summary our conclusions in section Ⅴ.

II. DYNAMIC DEPLOYING SCHEME

The dynamic deploying honeynet is mainly to solve two issue that are when and how to deploy the honeynet again. The redeploying opportunity should be determine firstly. There are several ways to determine when to deploy the honeynet. It is a simple, effectively way that other security tools inform the honeynet redeploying. Such as, the intrusion detection system can actively send message and control the honeynet deploying. The goal of dynamic deploying algorithms is to solve how to deploy the honeynet. The dynamic deploying algorithms are capable of generating new configuration for the honeynet.

A. Inducing degrees

In this work, we want to design independently deploying honeynet system regardless the other security detection systems. So the redeploying time should be determined by analyzing the collected data captured in the honeynet.

Definition 1 Inducing degree. The inducing degree is introduced to measure the inducing capacity of honeypot and relies on the number of the ICMP packets to and from one honeynet. Assume that the monitoring time $T_i = \{ t_1,t_2,\ldots,t_m\}$, the inducing degree is defined by using the Root Mean Square method.

$$I_i = \sqrt{\frac{(y_{i1} - \overline{y_i})^2 + (y_{i2} - \overline{y_i})^2 + \ldots + (y_{in} - \overline{y_i})^2}{m}} \qquad (1)$$

In Equ.1， where I_i represents the inducing degree, y_{ii} is the number of ICMP packets captured in in t_i ($1 \leqslant i \leqslant m$), and $\overline{y_i}$ is the mean value of the number of ICMP packets captured within the monitoring time T_i.

The inducing degree reflects the change rate of the number of packets captured in one honeynet. It can be used to determine when to launch a new deployment of honeynet. The variation of the number of ICMP packets to and from the honeynet is steady in normal state. However, when the honeynet is detected by the hackers and ignored gradually. The number of ICMP packets reduce abruptly, accordingly the change rate of the number of ICMP packets will be increased. Thus the inducing degree is considered as an indicator to redeploy the honeynet. It is noted that the variation of indcuing degree is greater when the honeynet is just redeployed. So the inducing degree require a sign that is positive or negative to distinguish the changing direction. When the inducing degree is positive, this means it is on the rise. And when the inducing degree is negative, this means it is declining. In brief, when the inducing degree is negative and greater value. The honeynet should be considered to redeploy.

B. Dynamic deploying algorithms

The static deploying scheme need the security managers design the configuration beforehand. On the contrary, the dynamic deploying scheme is capable of adjusting the honeynet configuration in real time without manual intervention. So the dynamic deploying algorithms should have certain intelligence. For this purpose, the following algorithms were designed by using different learning strategies to obtain optimal solution.

1） Cooperative Learning Algorithm

The essential part of the cooperative learning algorithm is to find current optimal solution of honeypot configuration. Then the others honeypots within the honeynet can learn from the current optimal honeypot. However, since the current optimal solution is not global optimal solution. The others honeypots mimic the current optimal honeypot with certain probability.

Definition 2 Maximum Active honeypot. Given a set of honeypots in a honeynet $H = \{h_1,h_2,\ldots, h_n\}$, the maximum active honeypot induces the hackers or worms than any others. Let us assume the monitoring time $T = \{ t_1,t_2,\ldots,t_m\}$, and the active degree denoted as h^d. Then active degree of the ith honeypot is denoted as Equ.2,

$$h^d{}_i = \sum_{i=1}^{m} S(i) \quad (2)$$

S(i) = 1 if packets have captured in the honeypot at t_i, otherwise S(i) = 0.

For example, T = $\{t_1,t_2,\ldots,t_{12}\}$ and two honeypots h_1,h_2. The honeypot h_1 is detected that some hackers or worms scanning it in $\{ t_1,t_5, t_7\}$, and the other h_2 is in $\{ t_3,t_4, t_7 t_8,t_{11}\}$. So $h_1^d = 3$ and $h_2^d = 5$. Thus, the maximum active honeypot has the maximum value of active degree. However, when the specific case appears $h_i^d = h_j^d$, then the number of inbound ICMP packets is considered. In this case, the honeypot has the maximum value of the number of inbound ICMP packets that is selected as the maximum active honeypot.

The maximum active honeypot is the current optimal configuration solution within the honeynet. And the main idea of cooperative learning algorithm is to mimic the configuration of the maximum active honeypot. After finding the maximum active honeypot, the others honeypots will modify theirs configuration following the maximum active honeypot with mimic probability.

Definition 3 mimicry probability. Let the mimicry probability be P, and the mimicry probability allows the cooperative learning algorithm apply different strategies. When the value of P is lower, the honeynet will apply pessimistic strategy to learn the configuration of the maximum active honeypot. otherwise, the optimstic learning strategy.

Cooperative-Learning (H, H', n,p)

INPUT: H is a set of honeypots that contains n honeypots.
INPUT: p is the value of mimicry probability.
OUTPUT: H' is a new set of honeypots.

1. Sort the active degree h^d_i in H such that $h^d_i > h^d_{i+1} > \ldots > h^d_n$
2. Select the maximum active honeypot h^d_x
3. Set the set H'= $\varnothing$
4. For i = 1 to n do
5. IF h_i is satisfied with p then
6. H' = H' $\bigcup\{h_x{}^i\}$; break;
7. Else IF h_i is not satisfied with p then
8. H' = H' $\bigcup\{h_i\}$;
9. return H'

We have developed the cooperative learning algorithm. For convenience, H denotes the current honeynet that contains n honeypots. And H' denotes the new honeypots set deploying next time. The algorithm first sorts n existing honeypots and finds the maximum active honeypot h_x, it then scans the set of H (line 4): for each honeypot h_i, if h_i is satisfied the condition of mimicing the maximum active honeypot (line 5), the configuration of h_i is modified on the basis of h_x and is added to H'. otherwise, the honeypot h_i doesn't change its configuration and is directly added to H' (line 8). The loop stops here (line 8) and H' is returned (line 9).

2） Evolvement Algorithm

The evolvement algorithm is to search the global optimal solution through the evolvement principe borrowed from evolution mechanism. The worst honeypots are eliminated and the best one are added at regular interval.

Definition 4 honeypot life.The lifespan is an important property of honeypot. The honeypot in the evolvement algorithm has three status: alive, death and mature. The honeypot status mainly rely on the life value. When the life value exceeds a specific threshold, the honeypot becomes mature and can generate the same configuration node for the honeynet. When the life value is lower than a specific threshold, then this honeypot will dies and be discarded from the honeynet.

The honeypot life is controlled by this mechanism that works as follows: In the initialization phase, each honeypot within the honeynet is set initial value. Suppose that there are two different deploying time T_1 and T_2, $\Delta T = T_2 - T_1$ and it is divided into k parts, $\Delta T = \{ t_1,t_2,\ldots,t_k\}$. If one honeypot has the minimum inbound packets number in t_i ($1 \leqslant i \leqslant k$), this honeypot life value will be reduced. On the contrary, the life value of the honeypot with the maximum inbound packets number in t_i ($1 \leqslant i \leqslant k$) will be increased.

Evolvement algorithm (H, H', n,)

INPUT: H is a set of honeypots that contains n honeypots.
OUTPUT: H' is a new set of honeypots.

1.While $|H'| < n$ do
2. CheckStatus(H, α, β)
3. h_i.status = getStatus(h_i);
4. IF h_i.status = mature then
5. H' = H' $\overset{+}{\bigcup}\{h_i\}$;
6. Else IF h_i.status = death
7. H' = H' $-\{h_i\}$;
8. Else
9. H' = H' $\bigcup\{h_i\}$;
10.EndWhile

In the evolvement algorithm, the two important property of honeypot denotes as h.life and h.status respectively. And $|H'|$ represents the number of honeypots in H'. The algorithm first checks the status of all the honeypots in the honeynet by using function CheckStatus (line 2). If the status of h_i is mature, a new honeypot will be generated based on the h_i configuration and be added to H' (line 5). If the status is death, this honeypot will be deleted from H' (line 7). Otherwise the status is alive, this honeypot will be directly added to H' in this case (line 9). The loop stops here (line 10) and the number of honeypots in H' is satisfied. When the algorithm stops, the result is a set of the newest honeypot configuration information.

The function of CheckStatus works as follows: calcuate the life of all the honeypots in the H within a pre-defined time frame (line1-6). The time frame is divided into k part and each part denotes t_i ($1 \leqslant i \leqslant k$). For each t_i in the time frame, the honeypot h_x and h_y with maximum or minimum inbound packets number are finded (line 2-3). Then the life value of the two honeypots are modified (line 4-5). Finally, the loop is to set the

status of all honeypots in the honeynet based on the life value and the threshold α，β (line 7).

CheckStatus(H, α，β)

INPUT: H is the honeypnet to be checked, and α，β is the threshold of mature and death.

1.For $t_i = t_1$ to t_k do
2. Sort the number of inbound packets of H in t_i such that $h_j > h_{j+1} > \ldots > h_n$
3. Select the maximum h_x and the minimum h_y
4. h_x.life ++;
5. h_y.life --;
6. Endfor
7.For i = 1 to n do
8. IF h_i.life > α then
9. h_i.status = mature;
10. else IF h_i.life < β
11. h_i.status = death;
12. else
13. h_i.status = alive;
14. Endfor

3) Hybrid Learning Algorithm

The hybrid learning algorithm is combined with the cooperative learning algorithm and the evolvement algorithm. There exist three status of honeypots in the evolvement algorithm. Most of honeypots are in alive status that have no chance to be optimized in each T_i. So the hybrid learning algorithm mainly provide an opportunity for the honeypots in alive status. When the current honeypot is alive, then the cooperative learning algorithm is used to optimize this honeypot (line 9). The function of cooperative learning is to generate new configuration for the ith honeypot based on the maximum active honeypot.

Hybrid Learning algorithm (H, H', n,)

INPUT: H is a set of honeypots that contains n honeypots.
OUTPUT: H' is a new set of honeypots.

1.While $|H'| < n$ do
2. CheckStatus(H,)
3. h_i.status = getStatus(h_i);
4. IF h_i.status = mature then
5. H' = H' $\overset{+}{\cup}\{h_i\}$;
6. Else IF h_i.status = death
7. H' = H' $-\{h_i\}$;
8. Else
9. $\overline{h_i}$ = cooperative_learning(H, h_i);
10. H' = H' $\cup\{\overline{h_i}\}$;
11.EndWhile
12. return H'

Function cooperative_learning(H, h_i ,p)

INPUT: H is a set of honeypots that contains n honeypots.
INPUT: h_i is the ith honeypot in H,and p is the mimicry probability.
OUTPUT: $\overline{h_i}$ is a new configuration of the ith honeypot in H.

1. Sort the active degree h^d_i in H such that $h^d_i > h^d_{i+1} > \ldots > h^d_n$
2. Select the maximum active honeypot h^d_x
3. IF h_i is satisfied with p then
4. $\overline{h_i}$ = h_x;
5. else IF h_i is not satisfied with p then
6. $\overline{h_i}$ = h_i;
7. return

C. Implementation

To evaluate the dynamic deploying approach, we have developed a prototype dynamic deploying honeynet based on the software Honeyd[1]. Since the virtual honeynet can be deployed by loading a configuration file honeyd.conf in Honeyd. It is a simple but effective way to execute redeploy honeynet using Honeyd. Our prototype system is aim to the distributed virtual low-interaction honeynet that is capable of capturing early worms, Botnet, Spam, etc. This section explains the implementation of the prototype system.

As shown in Fig.1, it is comprised of five major components: a database to store the configuration templates of honeypots, a set of preprocessors to statistic the raw tcpdump logs to extract the features for the dynamic deployment engine, the topology engine to generate the topology for the honeynet, the dynamic deployment engine implementing dynamic deployment algorithms and configuration manager scheduling the others modules to work effectively.

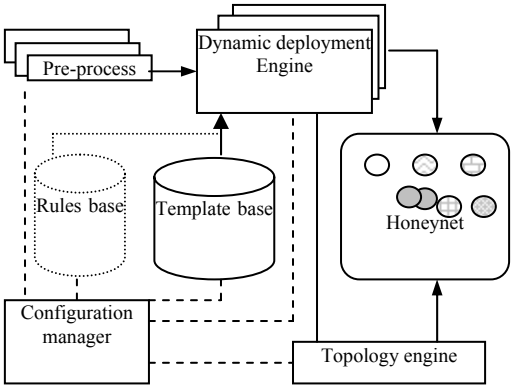

Figure 1. Prototype of Dynamic Deployment Honeynet

Configuration manager is the main procedure in the prototype system that is resposible for scheduling all other modules. It first initializes the template database and the rules set. Then determines to invoke the dynamic deployment algorithm. We have created preprocessors to deal with the alert of intrusion detection system and log of fireworks. The preprocessor is used to record and count the packets captured in the honeynet. Thus the different formats of logs need to be transformed into a uniform. The uniform log is comprised of eight fields: unique id, timestamp, source IP address, source port, destination IP address, destination port, and network protocol, packet size. Dynamic deployment engine analyzes the uniform log of currently honeynet and determines when to redeploy the honeynet. It includes different dynamic deployment algorithms that determine how to build the new configuration for honeynet. Topology engine is to assigned the IP address for

honeypot or generate specific topology for the whole honeynet. The topology engine maintains a IP address queue. This prevents the IP address conflicts between the honeypots and the real computers. In addition, it automatically assigns IP address for different sub-net honeypots. For example, if the honeynet user designates 172.16.0.30-120 as the honeypots IP address range, then the topology engine can generate random IP address within this scope for different honeypots. Finally, the topology engine is capable of building the topology of many virtual honeypots and modifying the topology of the honeynet in run time. Such as adding or removing some honeypots. Template base is a knowledge base storing the configuration template sets of honeypots. It includes configuration template and emulation template. The configuration template contains the inforamtion about the operating systems, ports and the emulation services. The emulation template is the emulation scripts that emulate all kinds of remote service to improve the interaction capacity of honeypot. The template base is built on top of MySQL and consists of 5 tables. The mainly table is configuration template table that defines many configuration properties of honeypot. This table refers to the emulation script table that stores scripts files. Rules base is another expert knowledge database storing rules that can provide heuristic configuration strategies to the prototype and now is very simplest implementation.

The prototype of dynamic deploying honeynet works as follows: the raw input data is stored as in form of Tcpdump and to be handled by the preprocessors. Then the features of status about honeynet is extracted and sent to the dynamic deployment engine. The dynamic deployment engine analyzes these statistic data and determine whether launching a new deployment or not. If the redeployment condition is satisfied, the dynamic deployment engine will automatically design new configuration scheme through the intelligence dynamic deployment algorithms. Then the dynamic deployment engine generate configuration file called honeyd.conf by using the template database and with the help of the topology engine that assign IP address for each honeypots. So the resulting configuration of honeynet is the text file honeyd.conf. The configuration manager make the new deployment by loading the new honeyd.conf file. Finally, the design of our prototype system has a advantage that is the separation of dynamic deployment engine from the whole framework of prototype system and it brings great flexibility to our approach.

III. EXPERIMENTS EVALUATION

A.The threshold of inducing degree

It is an important issue when to launch a new deployment. To solve this issue, we should firstly determine a threshold of inducing degree for the honeynet. When the current inducing degree exceeds this threshold, the configuration manager will redeploy the honeynet. Here we obtain the threshold through the experiment. Some network security testers were organized to emulate attacks. Comparing the inducing degree in the normal and the attack state, the threshold may be set by analyzing the variation of inducing degree in different states.

The attacks in the experiment are all from the DARPA 2000 that includes remote-to-local and local-to-local attacks[16]. Table 1 lists the attacks in the experiments. Since an intrusion incident is often composed of multiple step in attacks. The early attacks provide certain information to the later attacks that require them. For example, a typical network intrusion incident consists of an IP sweep, a port scanning, and a buffer overflow attack. Thus the experimental attacks are composed of multiple step in attacks. Attack #1 and #2 belong to the scanning step. Attack #3 and #4 are in the illegal monitoring phase. And the remainders are in attacking phase. Attack #5 is malicious code attack. Attack #6 belongs to the DDOS attack. Attack #7 is typically buffer overflow attack. The last one is new exploit software attack that is not involved in DARPA 2000.

TABLE1.
ATTACKS IN THE EXPERIMENT

Seq. ID	Attack Name or Attack tools
#1	ipsweep,portweep,resetscan
#2	queso,satan,mscan
#3	Xsnoop,illegalsniffer,snmpget,ncftp
#4	Guest,guestftp,guesttelnet,guestpop,dict
#5	Sshtrojan,ppmacro,xlock,sechole,casesen
#6	Smurf,udpstorm,mailbomb,sshprocesstable
#7	Sendmail,imap,ffbconfig,fdformat,loadmodule
#8	Exploits attatcks

The intrusion is a set of attacks, For example, an intrusion is <#1,#2,#5,#6>. The testers were grouped several teams and known nothing about networks. The emulation attacks were carried out for five times in 9:00,11:00,14:00,17:00,and 21:00. As shown in Fig.2(a). The variation of inducing degree appears five peak value. This shows that the inducing degree is an indicator to the attacks. Overall, the figure shows that the threshold of inducing degree can be set 234 due to the inducing degree in normal state less than 100.

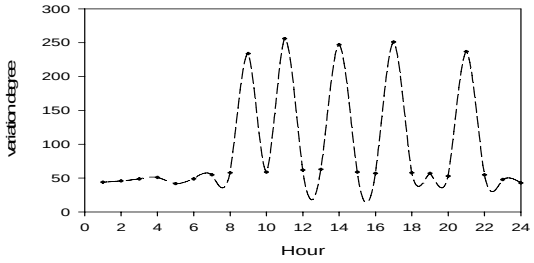

Figure 2. (a) the Variation of inducing degree in one day

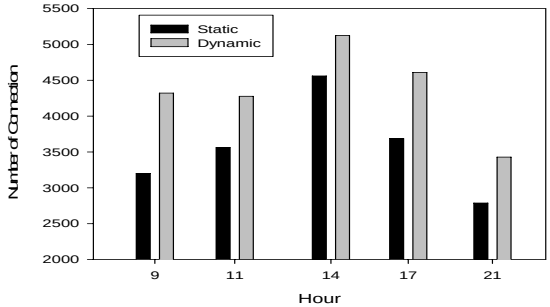

Figure 2. (b) the number of captured packets comparison of the static and dynamic deployment

B. Evaluation with Simulation

As mention in section II(A), all the honeynet traffic with static deploying had been recorded using Tcpdump. Then we had carried out the second pass intrusion with the same way in another day. Here the honeynet with dynamic deploying was tested and the monitoring time was one hour, T_i =1h. Fig. 2 (b) compares the number of connection to the honeypots in the honeynet between the static and dynamic deployment. We can see that the number of connection to the dynamic deployment honeynet is greater than the static one. The reason why the dynamic deployment honeynet outperform the static deployment one is that it has a better disguise capacity than the other one. Thus the testers trapped in the latter honeynet much longer than in the former and generated many more packets. However, the simulation method just shows that the dynamic deployment honeynet is superior to the static deployment one. And it is not enough to prove this fact in practical case. To address this issue, we provide a detailed report of dynamic deployment honeynet deploying in real networks in the next subsection.

C. Evaluation in practice

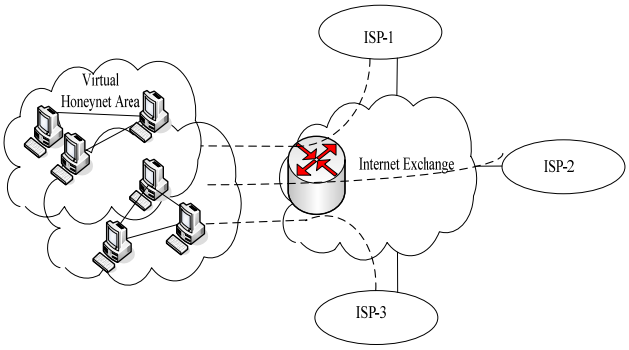

Figure.3 Honeynet deployment scheme in real networks.

In order to evaluate the dynamic honeynet in real network environment, the honeynet has been constructed in the internet exchange point as shown in Fig.3. The honeynet consists of fourty contiguous /24 subnets within one of a large /16 networks and doesn't have any protection appliance like firewall or IDS to temp attack traffic as many as possible. The honeypots were divided seven different configuration according to the number of the emulation sevices that include HTTP, NetBIOS, SMB, WINRPC, MSSQL, MYSQL, SMTP, Telnet, etc. The honeynet have deployed by using static and dynamic scheme respectively. Each scheme has recorded packets for thirty days. The hybrid learning algorithm was used in the dynamic deploying honeynet. The mimic probability p was set 0.2 and the monitoring interval was 12 hours. At the beginning, the two different deploying method had the same configuration. And the setting of deploying parameter is based on the security experts.

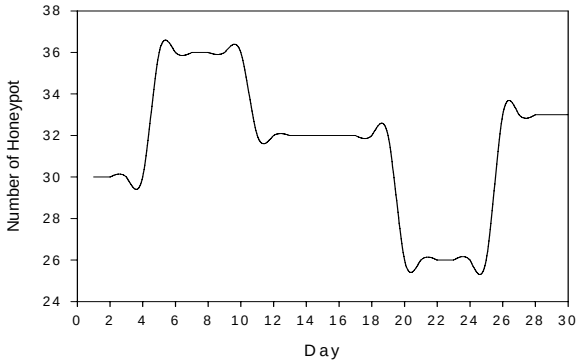

Figure 4. (a) Variation of the number of Honeypot

In the dynamic deploying honeynet, we can obtain the honeypot type number by analyzing the honeyd.conf file. The Fig. 4(a) describes the variation of the honeypot type number. there are five stages in the curve and it means the honeypot type number has changed four time. Thus, the four changes prove that the dynamic deployment honeynet has redeployed four times in thirty days and emerges on Aprial 5th, 11th, 20 th, 26 th, respectively.

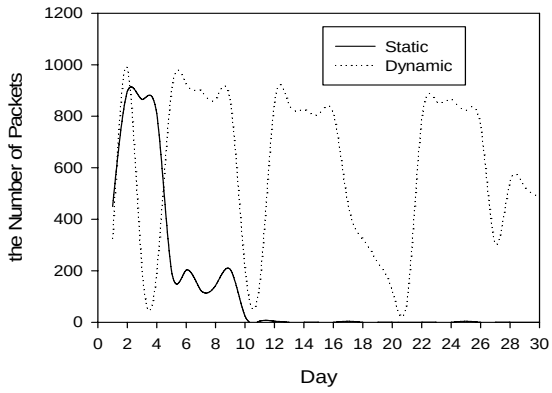

Figure 4. (b) Comparison of the number packets of the maximum active IP

We can further demonstrate that the dynamic deploying method improves the disguise capacity through analyzing the packets number of the maximum active IP. In generally, the maximum active IP address is currently the most active attack source. Its inbound or outbound packets number indicates the hackers from this IP address has launched a intrusion or not. As shown in Fig.4.(b), the packets number of the maximum active IP in the static deployment falls abruptly on 5th and disappears gradually. This means that the hacker had identified the honeynet and withdraw from it. However, there are several fluctuations in the curve of the dynamic deploying honeynet. It seems that the dynamic honeynet is capable of inducing the same hacker several times by using the dynamic deployment method. When deploying again, the hacker mistake this honeynet for a new attack target and scan it again. They will speed more time to identify the honeynet. So the dyanamic honeynet can deplete the hackers' energy.

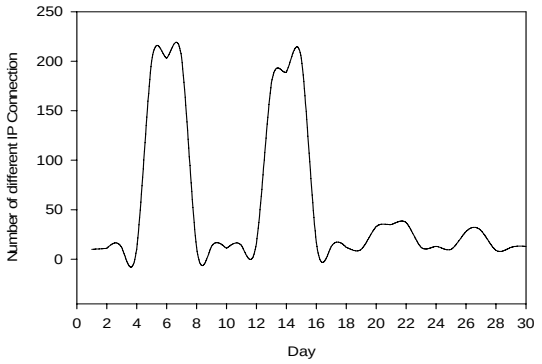

Figure 4. (c) Variation of the different IP connection number

Subsequently, we analyze the intrusions in the dyanamic honeynet to prove that it can tempt some important attacks. Fig. 4 (c) shows the variation of the number of different IP addresses connecting the same honeypot. The curve occurs two rising segments on 5th, 6th, 7th and 13th, 14th, 15th respectively. This results indicate that there are two distribution denial-of-sevice(DDOS) attacks in those days. In addition, from the number of packets connecting to the specific ports 901 and 4455 in the honeynet logs. The number of connections to port 901 on 20th and to port 4455 on 26 rise quickly. This means that two worms attacks appear during the monitoring days. So we conclude that the dynmaic deploying is capable of inducing the real attacks and has high value in practice.

IV. CONCLUSION

In this paper, we proposed a dynamic deploying approach to the distributed virtual low-interaction honeynet. We showed that this dynamic deploying honeynet is prior to the traditional static deploying honeynet in the disguise capacity. The dynamic deploying honeynet can be applied into many security applications, such as worms detection and monitoring, intrusion detection system, Botnet or Spam detection, etc. Several research problems remain unsolved. The important parameters in the dynamic deploying algorithms is not optimized. The dynamic deploying algorithms can handle different situations. Therefore, we will compare the performance of the different algorithms and obtain what case each algorithm is suitable for.

ACKNOWLEDGMENT

We would like to thank the support of National Nature Science Foundation of China (No.60970012), the Innovation Program of Shanghai Science and Technology Commission (No.09511501000, 09220502800), and Shanghai leading academic discipline project (S30501).

REFERENCES

[1] Niels Provos. A virtual honeypot framework. In proceeding of the 13th conference on USENIX security symposium. USENIX，Berkeley, USA 2004.

[2] Bailey,M., Cooke, E., Watson, D., Jahanian, F. and Provos,N. A hybrid honeypot architecture for scalable network monitoring, Tech. Rep. CSE-TR-499-04, U. Michigan.

[3] Vrable,M., Ma, J., Chen, J., Moore, D.,Vandekieft, E., Snoeren, A., Voelker, G. Savage,S. Scalability, fidelity and containment in the potemkin virtual honeyfarm, Proceedings of the ACM Symposium on Operating System Principles. ACM, New York, NY, USA,2005

[4] Chao Chen, Zeshen Chen, Yubin Li. Characterizing and defending against divide-conquer-scanning worms. Computer Networks, 54,3210-3222, 2010.

[5] Tang Yong, Chen,S. Defending against internet worms: a signature-base approach, Proceedings of the IEEE INFOCOM. 2005.

[6] Lan. Wang, Zhi.Li,Y. Chen.et, al. Thwarting zero-day polymorphic worms with network-level length-based signature generation. IEEE. Trans.on Networking. 18(1)53-66, 2010.

[7] Ping W., Lei Wu, Ryan Cunningham, Cliff C. Zou. Honeypot detection in advanced botnet attacks. Int. J. Information and Computer Security. Vol. 4， no. 1， 2010.

[8] Zhichun Li, Anup, G. ,Yan chen, Vern P. Towards Situational Awareness of large-Scale Botnet Probing Events. Trans. on Inforamtion Forensics and Security. Vol. 6, No. 1, 2011.

[9] Christian Kreibich and John Crowcroft.“Honeycomb creating intrusion detection signatures using honeypots.”In 2nd Workshop on Hot Topics in Networks(Hotnets-II),Cambridge,Massachusetts,November 2003.

[10] Hyang-Ah Kim and Brad Karp. Autograph:Toward automated,distributed worm signature detection, In 13th USENIX Security Symposium,San Diego, California, August 2004.

[11] J.Newsome,B.Karp, and D.Song. Polygraph:Automatically generating signatures for polymorphic worms, In IEEE S&P, 2005.

[12] N Krawetz. Anti-honeypot technology. IEEE Security&Privacy,2004,2(1):76-79

[13] Niels Provos. Open Source honeyd.http://wwww.citi.umich.edu/u/provos/honeyd/

[14] Z Kuwatly, Masri M Sraj, H Artail. A dynamic honeypot design for intrusion detection.ACS/IEEE Int’1 Conf on Pervasive Services(ICPS 2004),Beirut,Lebanon,2004

[15] Jason Chih-Hun Chang, Yi-Lang Tsai. Design of Virtual Honeynet Collaboration System in Existing Security Research Networks. ISCIT2010，IEEE,2010.

[16] MIT LINCOLN LAB.2000. DARPA 2000 intrusion detection evaluation datasets. http://ideval.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html.

Haifeng Wang. He received his diploma in computer science from the Shandong University in 1995, China. He is currently a PHD in the University of Shanghai for Science and Technology. His current research interests include the network security, network computing, GPGPU.

Qinkui Chen. Received the MS degree in computer science from the JILIN university, and PhD degree in University of Shanghai for Science and Technology. He is a full professor of computer science at the University of Shanghai for Science and Technology, China, where he is the head of the Network Computing Group and the Wireless Sensor Network Research Center. His research interests include WSN, network computing, and GPGPU.