International Journal of Networked and Distributed Computing, Vol. 1, No. 2 (April 2013), 108-123

Design and Implementation of Dynamic Hybrid Virtual Honeypot Architecture for Attack Analysis

Yonas Kibret[1]
Dept. of Computer Science and Engineering,
University of Electronic Science and Engineering of China
Chengdu, China
E-mail: yonaskibret72@yahoo.com

Wang Yong
Dept. of Computer Science and Engineering,
University of Electronic Science and Engineering of China
Chengdu, China
E-mail: cla@uestc.edu.cn

Received 12 March 2012
Accepted 7 November 2012

Abstract—Honeypots are dedicated machines whose aim is to delay and divert attackers away from critical resources in order to study new methods and tools used by attackers. However, when looking most of current honeypot systems are statically configured and managed. They are either low interaction honeypot or high interaction honeypot. On this paper, we proposed Dynamic Hybrid Virtual Honeypots Architecture in a single machine. It is capable of adapting in constantly changing network environment using both active and passive scanning. It also mitigates the drawback of low and high interaction honeypots. We use low interaction honeypots as proxy to claim for multiple IP address and to filter uninteresting traffic whereas high interaction honeypots to give optimal level of realism. To capture, analyze and control attack method and tools we used a gateway. Finally, we deploy the proposed architecture and present statically analysis of attacks. The experiment result proves this architecture can claim for multiple IP address, filter uninteresting traffic and gives a realism response for attacker.

Keywords: Low interaction honeypot, High Interaction Honeypot, Dynamic honeypot, Honeyd, Honeywall, VMware

1. Introduction

Internet security has received much focus since the past decades. Network assets are increasingly vulnerable to rapidly moving threats of today's Internet. Any threat to the network security can leads to heavy financial, economic and other loses. Automated attacks have impact beyond the scope of the individual host and enterprise, not only infecting vulnerable hosts with malicious code but also denying service to legitimate users of the network. According to Computer Security Institute (CSI) approximately half of interviewed organization experienced at least one security incident in

2011[1]. To keep up with current and future threats, a large part of corporations has to be considering security as one of their goals in the information technology.

The protection of network against security threats has become a crucial prerequisite for any organization. Since architecture of the Internet was designed with an aim to provide maximum functionality, it has some inherent weaknesses and incapability which result in successful origin and execution of attacks. As attackers becoming smarter, the technology they used becomes

more and more sophisticated. Thus, Instead of using traditional techniques such as firewalls and intrusion detection system (IDS) new methodology and tools have been emerging continuously.

Firewalls, for example, help to protect these organizations and prevent attackers from performing their activities. Intrusion Detection Systems (IDS) are another example of such tools allowing companies to detect and identify attacks, and provide reaction mechanisms against them, or at least reduce their effects. But these tools sometimes lack functionality of detecting new threats and collection of more information about the attacker's activities, methods and skills. In order to better protect our organization and build efficient security systems, the developers should gain knowledge of vulnerabilities, attacks and activities of attackers. Today many non-profit research organizations and educational institutions research and analyze methods and tactics of attackers, which acts against their networks. These organizations uses honeypot based tools to gather data about attackers and malicious software and provide critical additional information, such as their motives in attacking, how they communicate, when they attack systems and their actions after compromising a system.

Most of current honeypots systems are statically configured and managed. They can't interact in change of environment. And also they are either low or high interaction honeypots. High-interaction honeypots has limited view IP address and high cost involved in maintaining a farm of virtual honeypots. It can be compromised. If adversary gains full access to the system, He can use it to launch further attacks. But we can get maximum amount of logs information regarding the hacker's activities. The most common example on this moment is a rapidly spreading worm that attempts to infect new targets from the infected honeypots. On the contrary, real services are not running on low interaction honeypots, hence they can't be compromised, but they can only provide limited information about an attack and also unable to discover previously unknown attacks or vulnerabilities. But it can claim for multiple IP addresses and run different services on each host by emulating the appropriate Operating System with low cost. To have dynamicity and to mitigate the drawback of high and low interaction honeypot in order to study attacker tools and method we proposed new Dynamic Hybrid Virtual Honeypot Architecture.

The remaining parts of this thesis are organized as follows. In Section 2 we start with giving in-depth definition of the term honeypots and discuss different type of honeypots in detail. Section 3 gives a complete overview of the design and implementation of the proposed Dynamic Hybrid Virtual Honeypot architecture. Section 4 will elaborate analysis and evaluation of system. Section 5 presents the result of attacks statistical analysis. Section 6 presents the main technologies our proposed honeypot architecture rely on. Section 7 presents some conclusions and discusses directions on the proposed architecture.

2. Honeypot

A formal definition of a honeypot is a trap set to detect, deflect or in some manner counteract attempts at unauthorized use of information systems. Lance Spitzner, the founder of the Honeynet Project, defines a honeypot as: "A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource"[2]. It is an exciting technology with great potential for the field of network security. It can be understood as a resource used to divert attackers and hackers away from critical resources. It can also be used to study an attacker's methods and tools[3]. Neither any authorized activity runs on these resources nor do they have any production value i.e. no legitimate activity is carried out. It provides a large amount of valuable information for analysis and can detect variety of attacks, working even within encrypted environment[4]. It acts as a cherished observation and early warning tool but on the contrary it should be used with caution as it has risks associated with it. There are several types of honeypot, which can be categorized into five major categories.

2.1. Based on Level of Interaction

2.1.1. *Low Interaction Honeypot*

They are characterized by its minimal interaction with the hacker and emulate fake specific services. There are no real operating systems or services running on them, they are just emulations running above operating system layer. The deployment and maintenance of such systems is fairly simple and does not involve much risk, however, the logs for such

systems have limited amount of information regarding the hacker's activities. The interaction of the attacker with this system is limited and it is for small amount of time thus the attacker can not intrude the system[5]. Best examples for this type of honeypot are Honeyd[6], and KFSensor[7].

2.1.2. *High Interaction Honeypot*

High-interaction honeypot runs real systems and services and offer attackers complete operating systems as well as fully functional applications and services to interact with[8]. It contains more useful information and the highest risk, therefore requires the higher level of knowledge for the deployment and maintenance[9]. It captures extensive amounts of information by allowing attackers to interact with real systems where the full extent of their behavior can be studied and recorded. Hence they are specially used for research purposes.

2.2 Based on Adaptability

2.2.1. *Static Honeypot*

On this type of honeypot the number and location of honeypot are fixed. Hence it provides limited local views of network attacks[10]. Static honeypot have problem to learn dynamically in constantly changing network environments. In addition to this, static honeypot can't intelligently map and respond to our environment and also can't determine what systems are live, types of systems they are, and what kind of services are using.

2.2.2. *Dynamic Honeypot*

Dynamic honeypot is an autonomous honeypot capable of adapting in a dynamic and constantly changing network environment. The dynamic honeypot that we can just plug in and leave it to operate without the need to constantly update it[11]. It should be able to automatically identify information about production network and deploy honeypot based on this information. The most critical part of a dynamic honeypot is how the dynamic honeypot learns about our network[12]. There are two approaches active and passive fingerprint to learn the setup of our network.

2.3 Based on Hardware Deployment

2.3.1. *Physical honeypot*

As the name indicates this type of honeypot implemented on single machine running a real OS and services. On this type many security vulnerabilities are deliberately left on the operating system and some personal information is added to enhance authenticity[13]. Where honeypot is connected to a network and is accessible through a single IP address. Physical honeypot are always connected with the concept of high interaction honeypot and they are less practical in real scenarios due to limited view of their single IP address and high cost involved in maintaining a farm of physical honeypot.

2.3.2. *Virtual Honeypot*

They are usually implemented using a single physical machine. The host has several virtual honeypot. Virtual honeypot are more cost effective in monitoring large IP address spaces and emulating large IP addresses at the same time. Virtual honeypot have a number of benefits. A single physical system can emulate more than one virtual honeypot. They can emulate more than one operating system and they can emulate different IP addresses. Virtual honeypot require far less computational and network resources than physical honeypot. They also provide far greater flexibility in emulating various operating systems[13].

3. Design and Implementation of Proposed System

Dynamic honeypot is an autonomous honeypot capable of adapting in a dynamic and constantly changing network environment. One of the biggest challenges when deploying any type of a security system is in maintaining the functionality of the total system as the network topology or technology changes. This issue is especially critical for honeypots. What is sought is a dynamic honeypot that we can just plug in and leave it to operate without the need to constantly update it. It should be able to automatically determine how many honeypots to deploy, how to deploy them, and what they should look like to blend in with our environment. The services should be chosen so as to mimic the real services that a certain OS is able to provide and keep up with new technologies while removing old and obsolete services.

The proposed system faces same challenges to deploy the dynamicity behavior. What types of honeypots is used so as to easily configure and manage, to have high interactivity with backchat's, to claim multiple IP addresses and to minimize risk of detection and compromise from the simple honeyd to advanced

honeynet. There are also different configuration issues in deploying honeypots. One of configuration issue is what the honeypots looks like? Do we want it to appear as windows 7, Linux, Cisco router? We must ensure that the honeypots runs the same operating system as the production network so as the honeypots can blend in our environment. Once we determine the type of the operating system, the honeypots must reflect the same service as production networks operating system.

Another issue is how many honeypots will suffice? If the attack is high, large number of honeypots in the network would make it strong to mitigate the attack. They will provide higher quality of deception. On the other hand, at some point of time in the same network attack becomes low and number of legitimate clients requesting services increase. This leads to inefficient resource utilization. Hence the number of honeypots must be minimized for better resource utilization. The other issue is where to deploy honeypot? Before the firewall, after the firewall or in the demilitarized zone (DMZ). Attackers are restless thus they may use sophisticated tools to investigate the location of the honeypots in the network and if they locate it, the entire network is under risk of attack. Different deployment locations have their own advantage and disadvantage. Hence choosing the best deploring location for our proposed system is big challenge.

The proposed architecture consist three main modules. The first module used to learn the production network technology and topology to deploy honeypot dynamically. The second module is low interaction honeypot that used to view large IP space and filter uninteresting traffic to forward to high interaction honeypot. The last module is high interaction honeypot used to give optimal level of realism for attackers.

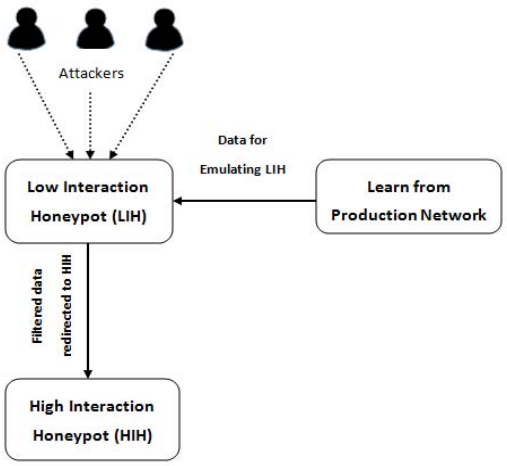

Fig. 1. Abstract structure of the proposed system

3.1 *Data Gathering Algorithm for Deploying of Honeypot*

The first and critical part to achieve dynamicity in our proposed system is selecting appropriate data gathering techniques to learn about the production network. Since it depends on the capability of fingerprinting technique and production network topology our proposed system use the merit of both active and passive fingerprinting. If our production network consists routing network, passive fingerprint unable to cross beyond switched networks unless routers in between are reconfigured. If the production network consists of servers, workstation and other peripheral devices connected with layer two switches active fingerprint not as realizable as passive fingerprint. On the other hand if the production network consists of servers, workstation and other peripheral devices connected through hub, active fingerprint introduce more traffic on a shared medium of our production network, hence packet sniffing is now feasible. After having a complete picture of production network using passive and active fingerprints our system estimates the number, personality, and services of the fake system in order to generate suitable configuration file to deploy on proposed system.

The passive fingerprint sniffs packets silently from the network in order to capture network activity, analyze, and determine IP address, operating system and services to deploy on the proposed architecture. This technique continually gathers mach information necessary to emulate the production network without introducing additional traffic on the network. For this technique we use snort only to gather information about the ports state and remote operating system, since it uses for us as data gathering and intrusion prevention system. To be effectively sniff packet, we place snort on the gateway of our production network.

The active fingerprinting which actively probe the network and determine which systems are live, what type of systems they are, and the services they are using in order to construct the proposed system architecture. For this method we use both Xprobe2 and Nmap. Xprob2 uses ICMP protocol for operating system fingerprinting where as Nmap used TCP/IP suite of protocols. On this technique we use Xprobe2 as default since it is the quietest active scanner to discover the remote operating system. Most of the time Xprobe2

accurately identify hosts which confused by Nmap scan, especially when there are no open ports available on the reporting the state of remote machine in the case of firewall. However, Nmap has provided us with some options that allow us to dynamically alter our scanning packets in a way that may evade firewalls. Hence we use nmap in the case of firewall.

The operations of the proposed system start with active and passive scanning for creation of initial configuration file. We use snort for the passive scanning to extract signature for deducing operating system, the state of ports and other information. After having this signature our system compares this information to a table that contains know parameters of operating system. The result data passed to network scanning parsers and stored in system database for creation initial honeyd configuration file for deployment. The result of target device[14]. But it has problem on correctly

passive scanning compared with data that contains known signature of operating systems. And then it inserts the identified target operating system, IP address, MAC address and other information in a table in the mysql database. Since passive scanning captures only interacting host, ports and services it has limited information. Hence during the passive scanning operation the default active scanning (Xprobe) starts scanning at predefined time slots for small group of IP address to minimize the utilization of extra bandwidth. The active scanning scans all IP address at predefined time slot to update the table with new information. The system only triggers the active scanning Nmap whenever firewall is implemented in the production network.

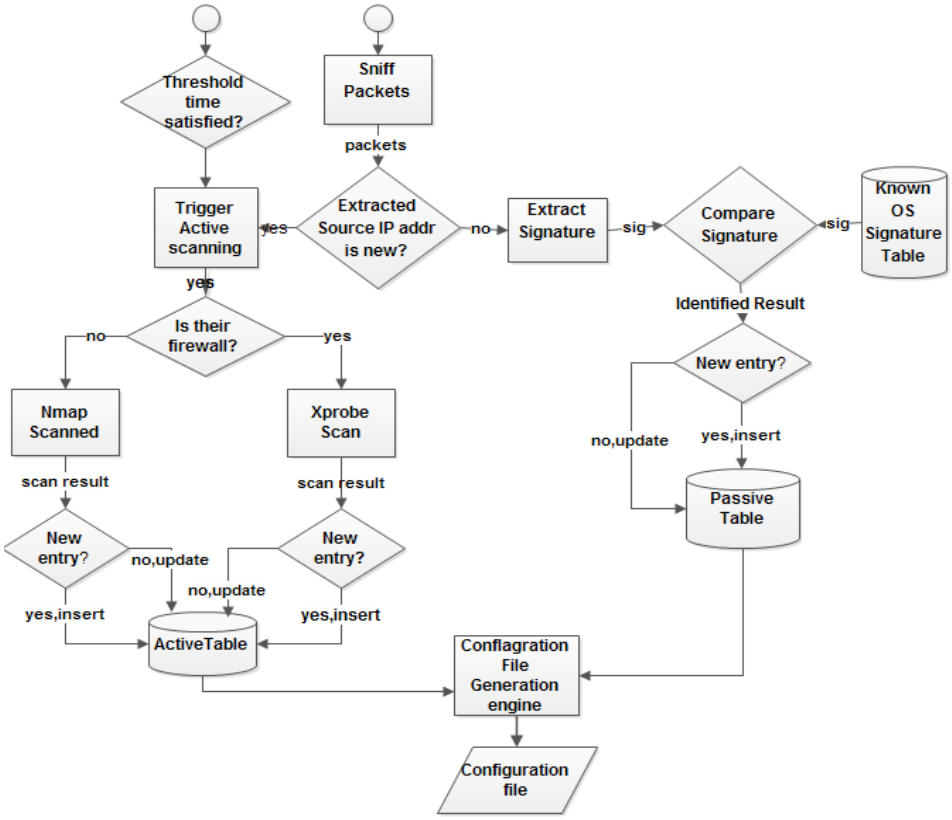

Fig. 2. Data gathering process for honeypot deployment

After initial configuration of the honeypots have been created, the passive scanning continues it works as it is but the default active scanning changes the time slot of scanning to higher time slot than the default scanning time slot which is used to create initial configuration file . It may be on the weekend time to minimize introducing of mach traffic. During passive scanning any change on the production network for example joining and leaving of host if happed on the production network, the system will automatically triggers the default active scanning. But if there is firewall it automatically triggers the active scanning Nmap. If the observed change is new either topological or technological, the system will insert all information associate with production network to a table in mysql database otherwise it will update the existing information which has change from previous status.

Creating of configuration file depends on the identified network image of the production network, and the attacker activity. In same condition the number of packet per unit time from attackers less than predefined threshold and also the identified network image contains hosts less than predefined threshold, the system will create configuration file which reflects exactly the same image as the production network for better deception i.e. if the production network has 10 window XP and 6 Ubuntu, the configuration file will create the same number of windows XP and Ubuntu with their port status. In other condition the attacker traffic would excides the normal condition, at this time the configuration file would create large number of honeypots in the network that will make it strong to mitigate the attack. They will provide higher quality of deception. The proposed system may have more than one honeypots. The total packets captured by all honeypots given as follows.

$$H_T = \sum_{i=1}^{n} h_i \tag{1}$$

Where h_i, is the number of the packet per unit time captured by i^{th} individual honeypot of proposed system and i=1, 2 …n.

The condition for creation a configuration file expressed as follows.

$$\begin{cases} \text{If Ht} < Tt, \textit{The same image as production network} \\ \text{If Ht} > Tt, \textit{Increase the number of honeypots} \end{cases} \tag{2}$$

Where, Tt is the predefined threshold packet per unit time from the attackers.

Each honeypots in the proposed system can induce different attacker which may have different source IP address. Hence the total packet for each individual honeypots can be given as follows.

$$h = \sum_{i=1}^{n} P_i^s \tag{3}$$

Where, P^s is the number of session packet per unit time from specific source IP address. If attacker wants to attack specific target, first the attacker scans the target. If our system deceives the attacker properly the attacker will continue its interacting with the honeypots otherwise it will halt its interaction by leaving scanning packets. Hence the session packet expressed as follows.

$$P^s = P^d + P^{sc} \tag{4}$$

Where P^d, is number of data packet per unit time from specific IP address, and P^{SC} is the number of scanning packet per unit time from specific IP address.

Let's assume there is even distribution of honeypots on the proposed system. The deception rate through honeypots can be given as:

$$D_r = \frac{Nh}{Nh+Nc} \tag{5}$$

Where, Nh is the numbers of honeypots, Nc is the numbers of computers in production network. From the formula we can see that by increasing the number of honeypot of can get grater deception.

Let's take class C network which contains 255 IP addresses from which one IP address used as broadcast address leaving 254 IP address. From those address some of them used by the production honeypots, same used for deploying the honeypots and the remained IP addresses will be free for further implementation of production network hosts and for deploying the honeypots. We can express deception rate interims of free IP address and production network IP address as follows.

$$IP_{total} = IP_{prod} + IP_{free} + IP_{honey} \tag{6}$$

$$D_r = \frac{IP_{total} - IP_{prod} - IP_{free}}{IP_{total} - IP_{prod} - IP_{free} + IP_{honey}} \tag{7}$$

Where, IP_{total} is the total IP address. In the case of class C network IP address is 254. IP_{prod},IP_{honey} and IP_{free} are IP address for production network ,IP address for honeypots and free IP address respectively.

From the figure 3, we can see that by increasing the number of honeypot we can get grater deception but it leads performance degradation. If the numbers of free IP addresses on the network increase, the deception rate will decreases.

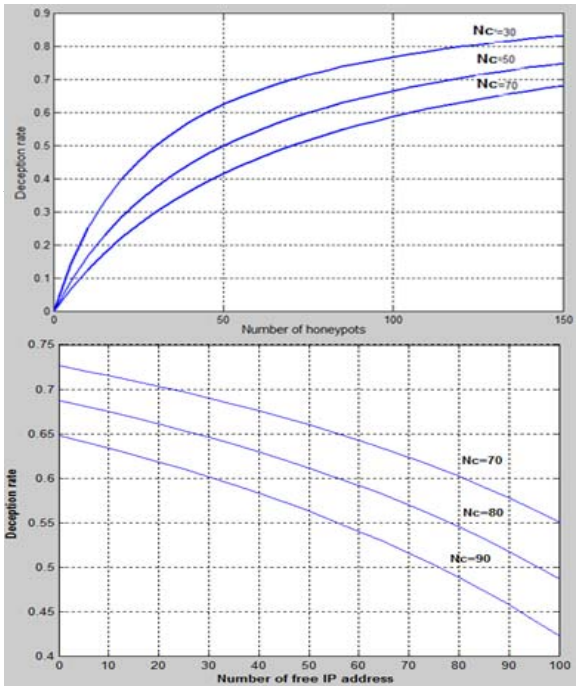

Fig 3 Deception rate of system versus number of honeypots and free IP address

3.2 *Hybrid System Architecture*

The proposed system uses Hybrid architecture of low interaction and high interaction honeypot. The low interaction honeypots act as light weighted proxies[15]. It captures all traffic from attacker and filter out uninteresting traffic that cannot be process and direct to high interaction honeypots for realistic response back to attackers. Honeyd is low interaction honeypot which has proxy mode of operation to play the role of front end for capturing and filtering. If the packet received by honeyd does not comply as standard three way handshakes the connection automatically dropped. If three way handshakes between attacker and low interaction honeypots completed, the connection directed to appropriate high interaction honeypots. Hence, at this time the low interaction honeypot honeyd becomes a relay proxy to direct all application level data to high interaction honeypots and vice versa. This process will continue until the attacker terminates the connection.

Using honeyd we can emulate several virtual machines which run the same operating systems and services as the high interaction honeypots. Hence without writing a complex script we can easily emulate specific service for low interaction honeypots with high interaction honeypots services. It is expected only to install the service what we need in high interaction honeypots. The proxies mode of honeyd listen traffic to specific ports based on particular configuration. The operating system and the services running on this light weighted proxies receives this traffic and directs to the real operating system and service running high interaction honeypots. To illustrate this, if low interaction honeypot honeyd emulate a windows XP SP2 machine with web server running, then the low interaction honeypots port 80 direct the received traffic to specific high interaction honeypots that run windows XP SP2 with web server.

Let's explain connection redirection from low interaction honeypots to high interaction honeypots, the attacker sends a TCP/SYN packet to the low interaction honeypot. If the low interaction honeypots configured to listen to specific ports, then it sends back a SYN/ACK packet and waits for the next packet to receive. If the next packet not ACK packet then the low interaction honeypots assumes as a port scan. Hence depending to the configuration of honeyd it will drop or ignore. If the third packet received from attacker is ACK then it is valid TCP connection and this point referred as zero point. Thus low interaction honeypot direct the packet to high interaction honeypots running the requested services. Then after, the connection establishment and the low interaction honeypot continue to work as a proxy.

The big problem is when a high interaction honeypots becomes compromised, an adversary gains full access to the system and can use it to launch further attacks. The numbers of outbound connections we allow from high interaction honeypots depend on how much risk we are willing to assume[16]. Hence limiting the number of outbound connections is important to prevent attackers from using the high interaction honeypots to scan or attack large numbers of production systems, or to

launch Denial of Service attacks. Hence we use honeywall for limiting the number of outbound that separates high interacting honeypots from low interaction honeypots. In addition to this, it performs access control of outbound connections from the high interaction honeypots. honeywall also uses for capturing attack activity, analyzing , controlling and visualizing analysis result. Honeywall includes many security tools such as Sebek, Snort, Snort_inline, Argus and walleye to perform its operation. We can see the location of honeywall in physical and logical structure in figure 4 and 5 respectively.

connections to prevent attackers from using high interaction honeypots. Honeywell is a gateway device

From figure 4, we can see the implementation of our proposed system architecture in an organization. This implementation shows when the traffic per unit time from attackers less than predefined threshold. Hence the honeypots has the same image as the production network. The production network used the IP address from 4 to 104 whereas the honeypots used from 152 to 252 with the same operating system and topology with production network.

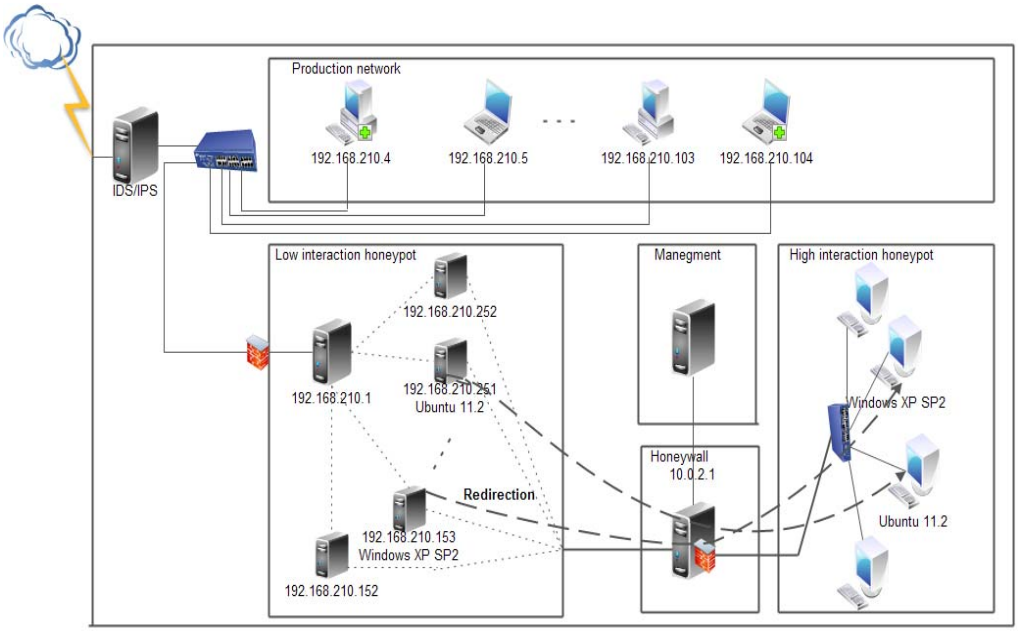

Fig 4 The proposed system deployment architecture

The proposed architecture induced attacks or worms. The Inducing Degree of honeypots can be expressed in termers of spam area and interacting degree. The interacting degree used to describe the interaction of the attackers with honeypots. When attackers interact with honeypots they may know they are interacting with honeypot. If they know they are interacting with honeypot using same honeypot detection method, they will halt automatically interacting by leaving same scanning packet. If they don't know they will interact until finish their session. When completing the session the honeypots will have much more packet than when attacker halts its interaction.

Interaction Degree (ℸ) expressed as number of session packets from the same internet address per unit time and can be obtained as follows.

$$\text{ℸ} = P_i^d / P_i^s \quad (8)$$

Max indicates the size of the data set. The second parameter is Spam Area (S) .We can get by counting different source IP addresses per unit time. It can be obtained as follows.

$$S = \sqrt{\frac{IP_{adds}}{P_h}} \quad (9)$$

Where, IP_{adds} is the number of different IP address per

$$\alpha + \beta - 1 \quad (11)$$

unit time captured by honeypot, P_h the number of packets captured by honeypot per unit time.

Interaction Degree (ℐ) of high interaction honeypots mach greater than low interaction honeypot but we can claim multiple IP address with low interaction honeypots. Spam Area (S) of low interaction honeypots mach greater than high interaction honeypots. Hence, our proposed system uses large spam area of low interaction honeypots and high interactive behavior of high interaction honeypots as we have seen previously. We can express Inducing Degree interns of interaction degree and Span Area using Cobb-Douglas production function.

$$HI_t = \gamma \mathcal{I}_t^{\alpha} S_t^{\beta} \qquad (10)$$

Where, γ is the inducing characteristic attack or worm of the honeypots, α and β are inducing elasticity of Interaction Degree and Spam Area. Inducing elasticity measures the responsiveness of inducing to the change in level of either Interaction Degree or Spam Area in inducing attack or worm.

To make more clear, for instance if α=0.35, 1% increase in interaction degree will results to approximately 0.35 % increase in inducing degree. Inducing degree of the proposed system directly proportional to Span area and Interactive degree. We can increase Inducing degree by either increasing Span area or Interactive degree.

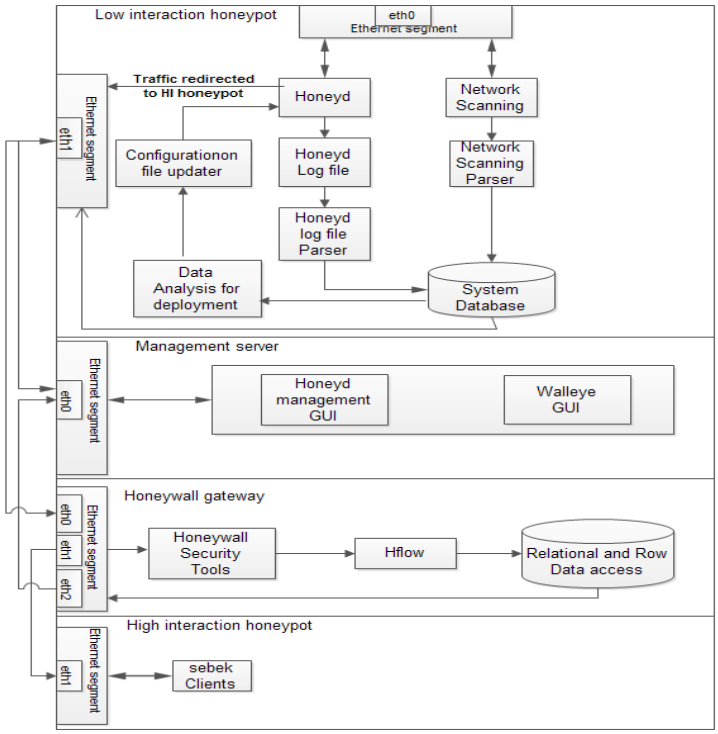

Fig. 5. Logical structure of proposed system

The interaction between attackers and honeypots depends on the how the honeypots gives a realistic response to the attacker and the spam area of the honeypots depends on how the honeypots can claim for multiple addresses. Low interaction honeypots such as honeyd has the capability of calming for multiple IP address and High interaction honeypots such as VMware have high interaction capability with attackers.

The cost of our proposed system is its initial cost in most cases. If there is change on the production network

due to either topology change or addition or removal of device, then the only thing that our proposed system can do is changing the configuration file. That is, if we add new computer that run windows XP SP2 on the network, then we will create the low interaction honeypot and direct traffic to already deployed high interaction honeypot that run windows XP SP2.

3.3 Location of Deployment

Selecting the deployment location for honeypot is a big challenge. Since we must sure that the honeypot blend into the real computer network and appear as any other computer node on the network[17]. When we deploy the honeypots before firewall or IDS, we can collect a lot of information about intrusion. But it will increase the publicity of our organization to hackers. When we deploy honeypots in demilitarized zone (DMZ), which is inside of the security system. The honeypot cannot be easily detected by attackers and also it may gather a large number of valuable information compare to the first position. When we deploy honeypots behind firewall we may introduce security risk to the internal network, if the internal network not secured by additional firewall against the honeypot. The main reasons for placing a honeypot behind a firewall are to detect internal attackers. It is also possible to detect a miss configured firewall which forwards unwanted traffic from the Internet to the internal network. Each deployment location has its own merit and demerit. Thus, in our proposed system we have different deploying option according to our need. We can deploy honeypots after firewall, before firewall, on DMZ or combination of them. We can see figure 6 which shows deploying on the three locations. If we already deploy our system on one of the three locations, to add on another location, no need to deploy other high interaction honeypots. We will use already deployed high interaction honeypots. Hence the only cost that we have is the low interaction honeypot.

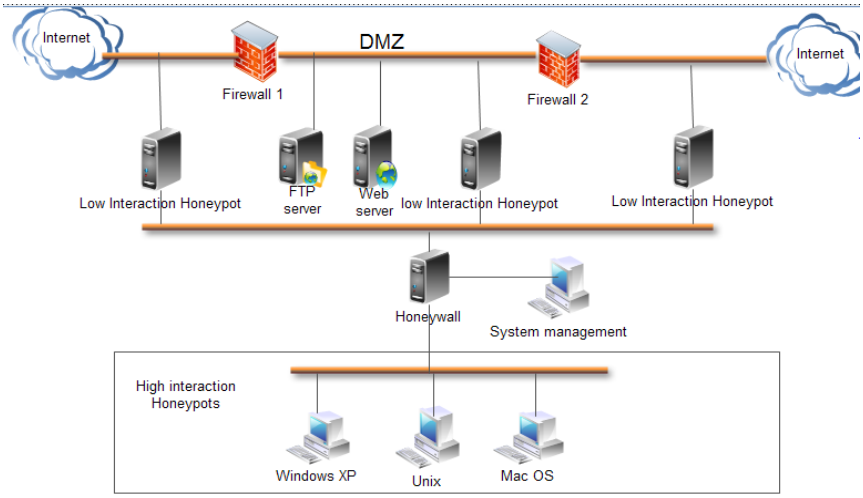

Fig. 6. Proposed System Implementation on the three locations

4. System Analysis and Evaluation

After we set up the architecture according to figure 4.3. We performed the experiment on a network that contains 16 windows XP SP2, 16 windows server 2003 SP2 and 16 Linux 2.2 honeypots. On each low interaction honeypots we opened the common ports to direct traffic destined to this port to high interaction honeypots

For example for windows sever 2003 SP2 we opened ports such as port http (80), DNS (53), telnet (23), FTP (22), SSH (21) and etc. All the other ports are reset. We used 2.2 GHz Pentium(R) dual-core CPU with 3GB RAM computer. We use honeyd for low interaction and VMware for high interaction.

In this section, we discuss the metrics in design our proposed honeypot system. Particularly, we focus on the tradeoffs associated with ability to provide span of coverage, the behavioral fidelity of a system, and other related concepts.

4.1 Span Area

Span Area refers to the ability of the system to effectively capture diverse scoped threat events using multiple IP address. Increasing the monitoring size yields increased visibility of threat. Our proposed architecture has the property of low interaction, hence it is easy to increase the number of IP address by changing the configuration file of honeyd. Honeyd enables on a single host to claim thousands of virtual honeypots at the same time. The test result shows up to 65536 hosts on a LAN network simulation. This improves cyber security and information infrastructure protection by making our information infrastructure more resilient against attacks.

The proposed architecture can also increase then coverage by deploying the low interaction honeypot before firewall, after firewall and in DMZ. It uses already deployed high interaction honeypot in one of the three locations. Thos locations can see different views of attacks. It is certainly increase the size of monitored space, this leads to increase visibility. Increased monitoring size decreases the time to detect diverse scope of threat. Because of those, we believe that effective coverage is achieved through the proposed architecture.

4.2 Behavioral fidelity

Behavioral fidelity refers to the degree to which our architecture is able to emulate the interactions and behavior of an end host node. Higher fidelity implies a closer approximation of end host behavior. This gives an increase in inducing degree of attacks and worms of the proposed system. In the context of honeypot system, a good behavioral fidelity can be obtained by decoying attackers scanning tools such as Nmap, Nessus and etc.

For our experiment we use active fingerprinting Nmap normal mode for operating system detection (-O) and version detection (sV). The version detection scan runs in conjunction with another scan type which identifies open ports. If another scan type is not specified on the command line for Nmap, it will select the default scanning technique. In our case TCP SYN scan runs prior to the version detection scan. We use the whole class C network for performing this experiment.

TCP SYN scan uses common method for port identification to gather information about open ports without TCP handshake process. It is commonly referred as half open scanning. It sends SYN to remote honeypot. If the remote honeypot port is open, it replays SYN/ACK otherwise RST. The IO traffic for this process showed in figure 7 between from 0 second to 170 second. The IO traffic until 170 seconds large than after 170 second since it is generated to scan all class C network hosts with their port state. After 170 second on identified open ports only the version detection started and also low interaction honeypots honeyd direct this traffic to high interaction honeypot (see figure 8). The IO traffic will minimize. It shows only interaction between attacker and the services on open ports of remote honeypots.

We use different color to show IO traffic for the interaction of services installed in high interaction honeypots on the graph. Nmap scanning result shows effectively detect the service and version installed in high interaction honeypots for low interaction honeypots without writing complex script. We detect the virtual honeypots using Nmap with their unique IP address and the version of the services for each low interaction honeypots through high interaction honeypot (see figure 9).

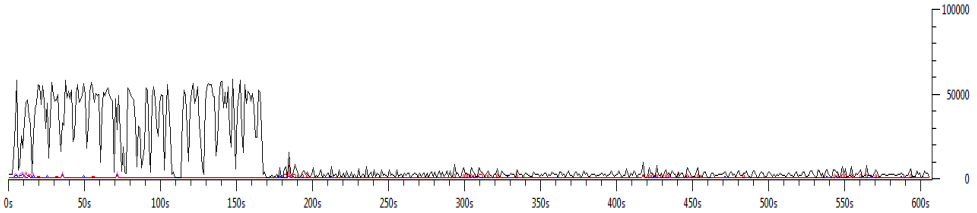

Fig.7. The interaction between attacker and low interaction honeypot honeyd

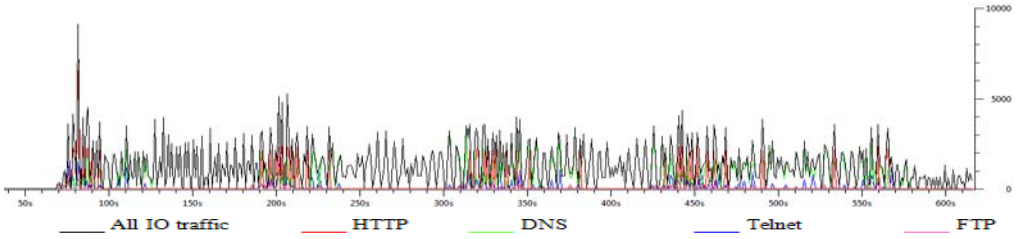

Fig.8. Detected services and version of low interaction honeypots through high interaction honeypot

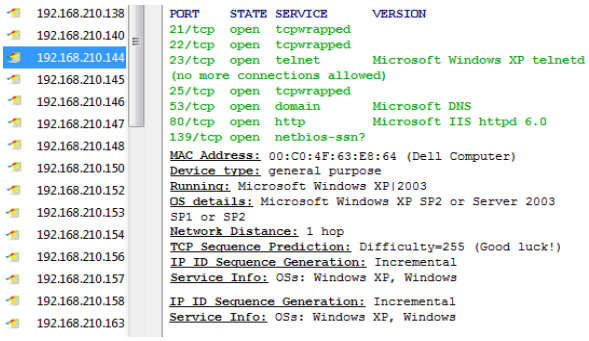

Fig.9. Detected services and version of low interaction honeypot through high interaction honeypot

5. Results

5.1 Statistical Analysis of Honeypot Data

On this section we present an overall statistical analysis of attacks gathered from our proposed honeypot architecture. We deploy our system to monitor attacker activities for a period of one week from April 2 to 7 2012. The experiment performed at the University network infrastructure. During this period our system received over 61,119 identified attack connections such as denial of service, user privilege gain, and Network Trojan. From the identified attack connection approximate average of 1273 attacks suffered per honeypot. To generate summaries and graphics from the logs file of honeyd and Snort we use trial version of sawmill 8 and Honeysum v0.3. Honeysum is free software and it is available when we installed honeyd.

These experiment results provide the better insight to the readers what was observed in our experiment. The overall statistical analysis of data collected from our proposed system during one week period shows our proposed honeypot system suffered from different kind of attacks. Table 1, Shows the number of attack connections per protocol during the observation period.

Table.1. Attack connection per protocol

No	Protocol	Connections	Percentage
1	TCP	31566	51.57%
2	UDP	27891	45.57%
3	ICMP	1742	2.84%
	Total	61199	

From Table 1, we can see that TCP is the most used protocol by attackers. This can prove the fact that multiple service and applications uses TCP compared to other protocol. UDP also has a considerable impact on our overall results.

5.1.1 *Geographical distribution of attacks*

We observed different attack originating from 75 countries across the globe. The top ten sources of attack countries was China, Taiwan, USA, France, Russian, Federation, Netherlands, Korea, Japan, and Iran. Among those countries about 45654 connections wear

received from china. This is 74.69% of the total attempt. Figure 10, shows top ten attacker belonging to countries other than china.

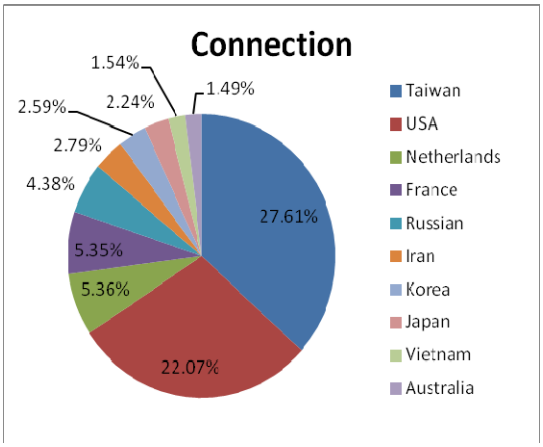

Fig.10. Graphical representation of Top 10 source countries

As we can see in figure 10, the implemented system attacked by different attackers from several geographical locations in the world. It is difficult to know in advance from where attacks launched to our proposed system, the vulnerabilities exploited by attacker and the scenarios they used to attack. On the other hand, the attack attempt by attacker might depend on the country they belong to. All this factors are uncertain and must be taken in account for statistical analysis and modeling. The data collected from our system can be processed using different statistical and probabilistic models. We can predict attack activities in different ways by considering geographic location, IP address of attacking machine, vulnerabilities exploited by attacks and etc

The evolution of the number of attacks per unit of time observed on all platforms considering specific geographical location described as follows:

Let's denote the random variable Y and dependent variable X

- Y (t) –is a function which describes the evolution of attack per unit time in all honeypots during the experiment.
- Xi (t) – is a function describing the evolution of attack per unit time in all honeypots from specific attacker belong country during experiment.

We have seen the empirical representation of Y (t) and Xi (t) corresponds to specific country i. when we inspect visually the empirical representation Y (t) and Xi (t) there is similarity between them for attackers that belong to china. To be sure we decide to analyze using linear regression models.

Using linear regression model we can estimate Y (t) from Xi (t) taking in consideration limited number of countries i. α and β are the intercept (the value of Y where x=0) and the slope of the line respectively. The linear regression estimated model given as follows.

$$Y^* = \sum \alpha_i x_i + \beta \qquad t = 1,2,3 \ldots k \qquad (12)$$

$$R^2 = \sum (Y^*(T) - Y_{av})^2 / \sum (Y(T) - Y_{av})^2 \qquad (13)$$

Y (T) and Y*(T) are the observed and estimated number of attacks per unit time T respectively. Y_{av} is the average number of attacks per unit time, taking into account the whole observation period. R^2 is measure the strength and direction of a linear relationship between the observed values and estimated model. If the estimated and the observed value have a strong linear correlation, then R^2 will close to +1. An R2 value of exactly +1 indicates a perfect positive fit.

We applied this model for one, two and more countries taking in account the total number of attacks. But we get the best fit only for one country especially for attacks that originate from china. This is true when we consider each honeypot platform .The R^2 value for attacks originated from china by considering all platforms is 0.936 which is best fit but for other countries is less than 0.5.

The estimated model by considering attacks that originated from china given as follow

$$Y^* = 33.404 + 1.224 * Xc \qquad (14)$$

Xc represents the evolution of number of attacks from china. We can see in figure11, the evolution of the estimated and observed attack per unit time. We used one hour unit of time in the graph. We can change unit of time depend on the collected data more than one hours.

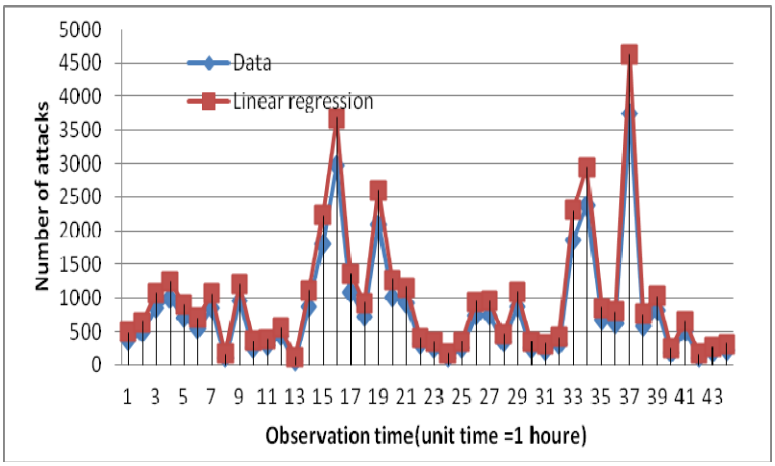

Fig.11. The observed and estimated number attack evolution per unit time by considering china.

The next table deals with the sources IP address of the attacker. Most of the IP addresses are from china. Hence to have picture on other country source IP address, we deal the source IP address that originate from china separately to other country source IP address. Only top 10 of IP addresses that attacked more our honeypot were listed in table 2, together with the country they belong to. The total number of sources IP address registered during observation period was 2312 IP addresses. This gives an average of 48 attacks on each honeypot. From the 2312 IP address 1396 IP address are originated from china, this gives an average of attack about 29 on each honeypot.

Table.2.Top 10 source IP address with country they belong

No	SrcIP	Country	Connections
1	101.15.xx.xxx	Taiwan	1711
2	111.242.xxx.xxx	Taiwan	880
3	93.27.xx.xxx	France	333
4	60.196.xx.x	Korea	330
5	118.90.xxx.xxx	Netherland s	298
6	78.194.xx.xxx	France	258
7	212.92.xxx.xxx	Russian	256
8	122.152.xxx.xxx	Japan	209
9	88.190.xx.xx	France	200
10	85.220.x.xxx	Iceland	199

5.2 Statistical analysis of Snort IDS Detected Events

In this section, we describe the analysis results of honeypot data according to IDS alerts. We used Snort for the detection and IDS alerts. During the analysis, we first counted the number of sessions detected by Snort IDS system, and we observed that among the total of 61,119 attempts of attack connection, 741 sessions triggered Snort IDS alerts. Within this period 74.8% of the attack was denial of service. Also, the average number of IDS detected sessions in each day was 123. In Table 3, we can see statistical information of IDS detected alert events and their distribution.

Before starting attack an attacker gather information about their target. The experiment result confirms this. After they gather information about the target, they determine what vulnerabilities exist before starting to exploit. As in our experiment, most attacks initially involved collecting the information. They usually use different port and vulnerability scanners, such as Nmap, Nessus and etc. to find open ports and vulnerabilities of victims. From Table 6.6 Snort alert events distribution we can see that the attacker uses Nmap to determine the vulnerabilities before launching an attack. After the scanning by Nmap, the attacker reviews the identified vulnerabilities and open ports, and then starts to exploit the existing vulnerabilities.

Table.3. Top 10 Summary of events alerts

No	SrcIP	Event frequency	
1	COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	488	65.6%
2	ORIGINAL DATAGRAM DUMP	76	10.2%
3	ICMP Large ICMP Packet	69	9.3%
4	ICMP PING NMAP	36	4.8%
5	[http_inspect] OVERSIZE REQUEST-URI DIRECTORY	31	4.2%
6	[http_inspect] DOUBLE DECODING ATTACK	11	1.5%
7	MISC MS Terminal server request	11	1.5%
8	MS-SQL probe response overflow attempt	9	1.2%
9	[portscan] TCP Portsweep	6	0.8%
10	BACKDOOR typot trojan traffic	4	0.5%

6. Related works

Based on honeypot technology researchers have developed many methods and tools for the collection and analysis attack activities. As main source of our work we use different papers implementation of honeypots architectures and practical tools they used for collection, analysis and controlling of attack.

Jiqiang et al. in[18] proposed dynamic honeypot which is capable of adapting in a dynamic and constantly changing network environment using low interaction honeypot honeyd. But it lost the realism response of high interaction honeypots. Kuwatly et al. in [11] also proposed dynamic honeypots for intrusion detection using low interaction honeypots honeyd and network scanning tools Snort. It uses only passive scanning technique and they only focus on the dynamicity of the honeypots.

The other references which is used often for our research is[15]. It is a distributed set of honeyfarms that can collaborate. On this system low interaction honeypots filter uninteresting traffic and forwarded to high interaction honeypots for better reliable response. On this implementation, if high interaction honeypots compromised, Attackers can use it for attacking production network and it hasn't the capability to interact with change network environments.

Cláudia J et al. in[9] describes the deployment of a honeypot by configuring a unique machine as part of the Distributed Honeypots Project. And presents all the tools needed to implement the honeypot environment, as well as the collection and analysis attacks. But it uses only low interactin honeypots honeyd for collection of attacks.

The Honeynet project[19] is a non-profit organization that is devoted to the research concerning honeypots and their underlying architecture. The main objective of this project is the in-depth analysis of attacks and the capture of malware. The Honeynet project deploys an architecture that consists of a central gateway, "Honeywall", and the honeypot network. Honeywall separates the network in which the honeypots are deployed from the rest of the network. In addition, Honeywall performs access control by limiting the outbound connections from the honeypots and captures network data. But it needs more hardware whether it is implemented with virtual[20] or physical to claim for multiple IP address.

7. Conclusions

This paper provides a detailed overview of the Dynamic Hybrid Virtual Honeypot Architecture for capturing and analyzing network attacks based on standard network technologies. It provides a complete framework for the construction of honeypot networks. Moreover, this architecture is flexible and easily expandable. This architecture consists of both low and high interaction honeypots. The low interaction honeypots act as lightweight proxies that aim to reduce the traffic that arrives to high interaction honeypots by filtering out uninteresting traffic such as port scanning.

The experiment result shows the implemented honeypots architecture can effectively emulate specific service for low interaction honeypots with high interaction honeypots services without writing a complex script on low interaction honeypots. It can claim for multiple IP address using the low interaction honeypots and filter uninteresting traffics according to our needs. It can also give reliable response for the attackers.

The installed architecture has a great utility for both

national and organization security. With this architecture it is possible to identify frequently accessed variability, tools and methods used by attackers, internal and external attacks. It helps to an organization to identify new threats and make action on the organization IDS to avoid worse consequences. It also contributes for detection of internal attack sources and infected host because during collection of data for analysis, we have seen a great volume of internal attacks traffic. In general, after we deploy the appropriate number, type and services on each honeypot host, our system induces attacks or worms. It makes attackers and hackers delay their activity and make them away from critical resources to study an attacker's methods and tool.

8. References

1. CSI. *Computer Crime and Security Survey 2010/2011*. 2011 [cited 2011; Available from: http://gocsi.com/survey.
2. Spitzner, L., *Definitions and Value of Honeypots.* 2002.
3. Ghourabi, A., T. Abbes, and A. Bouhoula. *Design and implementation of Web service honeypot*. in *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*. 2011.
4. Project, h. *Definitions and Value of Honeypots*. 2003 [cited 2011; Available from: http://www.tracking-hackers.com/papers/honeypots.html.
5. Abhishek Mairh, D.B., Kanchan Verma, Debasish Jena, *Honeypot in Network Security: A Survey.* Proceedings of the 2011 International Conference on Communication, Computing & Security, 2011.
6. Provos, N., *Honeyd: A Virtual Honeypot Daemon.* 10th DFNCERT Workshop, 2003.
7. KeyFocus.Ltd. *Advanced Windows Honeypot System:KFSensor* 2011.
8. Provos, N.H., Thorsten:, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. 2007: Addison Wesley Professional.
9. Pei-Sheng Huang, C.-H.Y., Tae-Nam Ahn, *Design and implementation of a distributed early warning system combined with intrusion detection system and honeypot.* ICHIT '09 Proceedings of the 2009 International Conference on Hybrid Information Technology, 2009.
10. R.C. Joshi, A.S., *A New Paradigm to Information Security*. 2011.
11. Kuwatly, I., et al. *A dynamic honeypot design for intrusion detection*. in *Pervasive Services, 2004. ICPS 2004. Proceedings. The IEEE/ACS International Conference on*. 2004.
12. Budiarto, R., et al. *Honeypots: why we need a dynamics honeypots?* in *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on*. 2004.
13. Li, L., S. Hua, and Z. Zhenyu. *The research and design of honeypot system applied in the LAN security*. in *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*. 2011.
14. Allen, J.M., *OS and Application Fingerprinting Techniques.* 2007.
15. *Honeypot Node Architecture.* European network affined honeypots 2006. **SIXTH FRAMEWORK PROGRAMME**.
16. Project, H. *Gen II Honeynet*. [cited 2011; Available from: http://www.linuxvoodoo.com/resources/security/gen2.
17. Wang, H. and Q. Chen. *Design of cooperative deployment in distributed Honeynet system*. in *Computer Supported Cooperative Work in Design (CSCWD), 2010 14th International Conference on*. 2010.
18. Jiqiang, Z. and W. Keqi. *Design and implementation of dynamic virtual network*. in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*. 2011.
19. Project, h. *Know Your Enemy:Honeynets*. 2006; 31 May, 2006:[Available from: http://old.honeynet.org/papers/honeynet/.
20. Abbasi, F.H. and R.J. Harris. *Experiences with a Generation III virtual Honeynet*. in *Telecommunication Networks and Applications Conference (ATNAC), 2009 Australasian*. 2009.