Improving Deception Capability in Honeynet through Data Manipulation

Akingbola R.A., Dahunsi F.M., Alese B.K., Adewale O.S, Ogundele T.J
Computer Science Department,
Federal University of Technology
Akure, Ondo State, Nigeria

Abstract

Emergence of honeynet, which is a network of honeypots, has changed the phase of internet security and information gathering about intruders. In improving deception of the honeynet which in turn increases the workload of attackers, most honeynet deployments are geared towards improving on the deception deal by using MAC addresses, IP addresses and protocols on host machine. Attackers gain access to networks with different motives, this work focuses on attackers who aim for data theft, data insertion, data deletion and data modification This paper also includes data manipulation, a new component introduced by the users of these hosts. This data content contributes to improving deception by dynamically sorting, shuffling, updating, manipulating and mapping each last modified state of the data to unique MAC/IP addresses of an intruder machine that modified the system.

1. Introduction

The challenges about network security have been on the increase because of the increasing magnitude of electronic commerce occurring over the Internet and the rapidly evolving business trend towards telecommuting. Therefore, more sensitive and critical information is crossing the world than ever before [1]. Over the last few years, network based intrusions have increased rapidly, due to the increase and popularity of various attack tools easily available today. Due to this increase in intrusions, the concept of network Honeypots are being developed, which can be used to trap and decode the attack methods of the malicious attackers. Honeynets are different from Intrusion Detection and Intrusion Prevention Systems technologies as they are not limited to solving a specific problem; instead they are highly flexible and can be used in a number of situations for different purposes.

Some of their capabilities include; firstly detection of attacks, which is similar to Intrusion Detection Systems. Secondly, they can be used to deter an attacker and his attack, a functionality provided by firewalls. Thirdly, they can be used for capturing and analyzing automated attacks, such as worms. Finally, they can also act as indicators and warning sensors.

2. About Honeynet

2.1. Types of Honeynets

Doering mentioned that honeynets can be classified in two broad categories which are Research and Production honeynets. This categorization is based on the purpose of their use [2]. Honeynets can also be categorized in another ways such as on the basis of what they are to be used for, its interaction level with the attackers, etc

2.2. Production Honeynets

Production honeynets are those used by organizations to protect their networks and to help them in mitigating risk. Production honeynets are easier to build, deploy and maintain as they require less functionality than the research honeynets. By using production honeynets, sources of attacks can be detected. However the attackers might not be known, how they are organized and the tools utilized might also not be known. While production honeynets might be helpful, they give less information about the attackers. Production honeynets mirror the production servers or any services for the attackers to work with and to expose any vulnerability present in the network. Honeynets provide alerts and warning about attacks and vulnerabilities of the network to the network administrators. These alerts and warnings are helpful in reducing risks of intrusion [3].

2.3. Research Honeynets

The research honeypots normally do not add any direct value to the organization; instead they are designed to gather information about the blackhat community. The organizations like government agencies, defense institutions, universities and large corporations use research honeynets to collect information about the threats they face. Necessary precautions can be taken on the basis of this information to counter those threats. The focus of research honeynets is to gather intelligence and to understand the ways and means used by the attackers during an attack. This information is helpful to determine the actions, intentions and sometimes the attackers themselves. As mentioned earlier these

types of honeynets do not add any direct value to the organizations' security, whereas the lessons learned through them can prove to be fruitful future use. Research honeynets are both complex to deploy and to maintain.

Research honeynets can be used as a platform for studying cyber threats and for extensive research. The attackers can be constantly monitored and all their actions recorded while compromising any system. Intelligence gathering is one of the most exciting and unique feature of research honeynets.. Sometimes the use of research honeynets can result in the discovery of new worms which might prove useful in the development of forensic skills.

These categorizations of honeynets are simply a guideline to identify their purpose, while the distinction is not absolute. Sometimes the same honeynets may be either a production or a research honeynets. It does not depend as much on how it is built but how it is actually used [2]. For honeypots to add value, the goal to be achieved must first be identified and determined. Only then will the appropriate honeypot be selected and implemented within an organization [3].

If an organization is using the honeynets as a production solution, they will only be interested in detecting the attack, blocking the attacker, and perhaps even prosecuting the individuals involved. Whereas if the organization is using the same honeynets as a research solution, it will be more interested in what tools the attackers are using, where they are coming from, and their activities after they have compromised the honeynets. It's the same honeynets, with the same information captured. The difference is in its purpose, the information can be used as either a production or research solution.

2.4. Level of Interaction

Honeynets can be classified according to their level of interaction, low-interaction honeynets and high-interaction honeynets [1], [2]. Low interaction honeypots does not emulate all aspects of the operating system, it emulates only some part of the operating system and a set of services/protocols [14], [16]. It does not provide operating system access to the intruder instead it makes available only services [15]. They are easier to deploy and maintain, and due to their limited abilities, they are safer [15],[17]. Low interaction honeypots are easier to detect by the attacker and they are only able to collect limited information about the attack and therefore they have limited reactions to attacks [14],

High interaction honeypots are more sophisticated that the low interaction honeypot, therefore they are more difficult to design, deploy and maintain. The systems are often made up of real services and operating systems. Deploying them is time-consuming and they have higher risks and therefore considered the most dangerous. They are more reliable and intruders rarely detect them [15], [14]. They are able to capture and log information about attacker for more informed decision making.

2.5. Method of Implementation

They can also be classification based on their method of implementation: physical and virtual honeypots. Physical honeypots are hosted on another system and do not have unique IP address. They are difficult to deploy and maintain, they are often time intensive and expensive. There is also a lot of administrative work to be carried out to ensure proper securing and monitoring [14], [16].

The Virtual honeypotsdo not require additional computer systems for their implementation; they can be hosted on another machine. It is easier to populate the network, inexpensive and less time consuming. Virtual honeypots have similar advantages and disadvantages with low interaction honeypots [13],[14],[16].

2.6. Deployment

Honeynets is simply the network of Honeypot. The classical Honeypot deployment consists of one Honeypot placed within a production network. It is possible to deploy more than one Honeypot, but each of these is a stand-alone solution and according to the concept, it is still a single machine. Deploying a Honeynet requires at least two devices:

- Honeypot
- Honeywall

In that scenario the attacker is given a Honeypot with a real operating system. This means he can fully access and mangle it. Through that possibility an attacker could easily attack other systems or launch a denial-of-service attack [2]. To reduce this risk a firewall is configured on the Honeywall, which limits the outbound connections. Access to the production network is completely restricted. The Honeywall also maintains an Intrusion Detection System which monitors and records every packet going to and from the Honeynet.

3. Related work

Cohen proposed that in order to improve the situation for the defender, there is the need to increase the intelligence workload by increasing the size of the search space [4]. He configured one Ethernet card as the host for numerous IP. Each of these can optionally have their own MAC address as well. This technique can be applied for deception by filling a large address space that would normally be

sparsely populated to dazzle the attacker. Cohen combined the use of the honeyd system developed by Provos to further enhance the deception by providing bogus TCP/IP fingerprints to the attacker. This aid in the deception by making the system appears at a TCP/IP stack level to also correspond to the deceptive system. This approach of offering a wider range of systems will increase the workload of the intelligence effort in determining which of these systems are legitimate and which are not. Cohen used Deception Tool Kit (DTK) to populate more than 40,000 IP addresses with false services. This was highly effective in increasing the intelligence workload, in increasing the time to attack and decreasing the odds of certain classes of intelligence probes going undetected. The effect should be that of an increased attack detection window for the honeynet system [4].

In Cai, et al the objective of the honeynet was to collect data as long as possible without being mapped by an Attacker [5]. However, once identified, a honeynet will have to be assigned to a new set of addresses. This remapping or shuffling was said to be a costly process. An additional objective of the Attacker is to identify the honeynet with a minimum number of probes. The Defender has two objectives. The first is to prevent the honeynet from being identified, which is done by periodically shuffling its location within the address space. The second objective is to extend the duration of shuffling epochs in order to minimize demands on the system responsible for shuffling. The Defender can use its protocol mimicking capability as a means for delaying the reshuffling.

Garg and Grosu considered the attacker and the honeynet system as players in a strategic non-cooperative game [6]. They denote the attacker by A and the defender, which is the honeynet system, by D. The honeynet is assumed to be composed of k honeypot hosts out of n possible hosts within a block of IP addresses. The goal of the attacker is to attack a host that is not a honeypot while the goal of the defender is to have ensure that the honeypot was attacked. Nandan and Daniel called the game that model the situation, a (n,k)-honeynet game, denoted as H,G (n,k)

Definition 1: A (n,k)-honeynet game consists of:
(i) the set of players N {A,D}.
(ii) the set of actions for each player:

(a) AD = {(x1,x2,…….xn)|k vector components are 1 and n-k components are 0}. If xi=1 then D places a honeypot at position (address) i If xi=0 then D places a regular host at position i.

(b) AA = { {j|j=1,…….n}. A chooses to attack host j.

(iii) the payoff functions of each player:

$$UA((x1,x2,\ldots\ldots xn), j) = \begin{cases} -C1 & \text{if } Xj=1 \\ C2 & \text{if } Xj=0 \end{cases} \quad (1)$$

$$UD((x1,x2,\ldots\ldots xn), j) = \begin{cases} C1 & \text{if } Xj=1 \\ -C2 & \text{if } Xj=0 \end{cases} \quad (2)$$

where Ci>0,for i ∈{1,2}

The parameter c1 represents the payoff gained by the defender and also the loss incurred by the attacker if the attacker attacks a honeypot. The parameter c2 represents the payoff gained by the attacker and also the loss incurred by the defender if the attacker attacks a regular host.

According to Rammidi, the concept of virtualization cannot be overemphasized when it comes to the design of honeynet [7]. Virtual honeynet is a technology that virtually implements many different operating systems in one hardware computer, and hence instead of having a honeynet of different physically separate honeypots, all the honeypots will be virtually housed in one machine and still appear to the attacker like its different separate machines. The main advantage of using a virtual honeynet over a classic honeynet is the cost of equipment; a virtual honeynet needs about one machine compared to buying many physical machines and their connecting equipment. It is also easy to manage since all the honeypots are configured in one machine [8].

4. Motivation

Honeynet works by providing an illusion that appears to be real and desirable to an attacker. The attacker, in searching for the honey of interest, comes across the honeynet, and starts to have a taste of its wares. If they are appealing enough, the attacker spends significant time and effort getting at the honey provided. If the attacker has limited resources, the time spent going after the honeynet is the time not spent going after other things the honeynet is intended to protect. If the attacker uses tools and techniques in attacking the honeynet, some aspects of those tools and techniques are revealed to the defender in the attack on the honeynet.

Deception has several advantages has it increases the workload of the attackers. The attackers cannot tell which attack attempt works and which fails. It exhausts attacker's resources; increase the sophistication for an attacker and attacker's uncertainty; all of which is an added advantage to the security of our production network. According to Jai et al (2010), the objective of a honeynet is to collect data as long as possible without being mapped by an attacker. However, once identified, a honeynet will have to be assigned to a new set of addresses. Also an attacker aims at getting into a victims system with

minimum probe. In the case where an attacker gains access into a honeypot, administrators are advised to ensure restriction to the production server from the honeynet by the outbound settings to avoid the attacker using the compromised honeypot as a lunch pad to attack other non-honepot system. No doubt, one detail to be looked into is that attackers should not be allowed to use the honeypot as a platform to lunch further attack. However, the attacker should be contained and occupied with interesting and seemingly genuine information in order to keep him busy on the honeypot while information about him is being gathered. One of such interesting and seemingly genuine information or data that this work tends to look into is the data/information stored in the production system. This data are mirrored and stored on the honeynet. This data/information could be, password files, personal profile, grade files of students, classified information, Close Circuit Television files, etc, depending on which organization owns the production server to be mirrored. The presentation of the mirrored data on the honeynet would be modified to render them useless to an intruder.

5. Methodology/conceptual framework

In honeynet deployment, the Honeynet Project defined three types of architecture Abbasi and Harris (). These architectures are:

1) Generation I
2) Generation II
3) Generation III

Gen I Honeynet were developed in 1999 by the Honeynet Project. Its purpose was to capture attackers' activity and give them the feeling of a real network. The architecture is simple with a firewall aided by an IDS at front and honeypots placed behind it. This makes it detectable by attackers.

Gen II honeynets were first introduced in 2001 and Gen III honeynets was released in the end of 2004. Gen II honeynets were made in order to address the issues of Gen I honeynets. Gen II and Gen III honeynets have the same architecture. The only difference is in the improvements in deployment and management. In Gen III honeynets, Sebek server was built into the honeywall. Sebek is a stealthy capturing tool installed on honeypots that capture and log all requests (read[] and write []) sent to the system read and write system call. This is very helpful in providing an insight on the attacker.
Virtualization would be employed in the design of the honeynet. As mentioned earlier, it saves the cost of setting up the honeynet as multiple physical machines would not have to be bought. Rather, other needed machines would be virtually configured on a physical machine.

Below is the honeynet setup with two physical computers. System (1) is used for virtualization while system (2) is used for managing the honeynet; it collects all the information gathered from an attacker for analysis.

The diagram in Figure 1, is the proposed architecture for the conceptual framework of the system. It is divided into sections for ease of analysis, Sections A, B and C.

5.1. Section A

Section A consists of the internet, a network router and the physical machine on which the VMware Workstation is used to emulate four virtual machines; one honeywall and three honeypots. Whenever a connection is made to the router from the internet, it will either be passed to the production server or to the honeynet (Figure 2). This decision by the router is based on some predefined settings, and data controls, setup on it in order to determine if a traffic is legitimate or it is an attacker. The Virtual Honeynet is set up on this physical machine with a Linux Operating System installed on it the machine then serve as the host machine on which the honeynet is setup. The traffic from this section is then passed to the virtual honeywall in section B

5.2. Section B

Section B houses the second physical machine where all captured data and information gathered from the attackers are collected. This system would be configured to receive data strictly from the honeywall in the same section. This data movement is stealthily done, so as not to appear to the attacker. It is from this system that all logged events are later pulled out for analysis.

The section also houses the transparent virtual honeywall, a key component to the design of this honeynet. A honeywall is a computer system that separates honeypots on a honeynet from the production network the honeynet is deployed on. All network traffic that enters or leaves the honeynet passes through the honeywall. All traffic that passes through the honeywall are monitored and recorded for later analysis. The honeywall is a transparent honeywall in bridge mode with no IP for inbound and outbound traffic; hence is not visible to an attacker. A Honeywall software called Roo is installed on the Honeywall. Roo version 2.0 was developed by the Honeynet Project Community with Generation III architecture of Honeynet. The honeywall typically allows all connections into the honeynet, but carefully controls connections originating from the honeypots inside the honeynet. The honeywall uses a combination of outbound connection-limiting and a network intrusion prevention system to limit the chances that a

compromised honeypot can be used to launch attacks against non-honeynet systems.

This section is further divided into data control, data capture and data analysis.

Data Control: The Roo includes two primary methods for data control. They are rate-limiting of outbound connections based on Iptables and the Snort-Inline intrusion prevention system. The rate-limiting firewall script can be configured to set upper limits on the amount of data that can be sent from each honeypot per unit time. The limits are based on outbound connections in the case of TCP and the number of packets for UDP, ICMP, etc. Outbound packets that are allowed through the firewall are passed to Snort-Inline for further inspection.

Data Capture: The honeywall records a variety of different types of information that is gathered on the honeynet. This includes all of the network traffic that enters and leaves the honeynet as well as data that is sent to the honeywall by the honeypots themselves. The data that is collected by the capture integrated on the honeywall. The tools are:

- Snort: This is a sniffer Intrusion Detection System. Generates IDS alerts of traffic into and out of the honeynet.
- Snort_inline: This is a sniffer Intrusion Prevention System. It either drop or mitigate an attack by screening packets from the IPTable.
- Argus: This records information on network traffic and identify flow IPs, ports and packet numbers.
- Hflow2: This is a data coalescing tool for honeynet data analysis.
- Pof: This is Passive Operating System fingerprinting tool.
- Tcpdump: This records all packets that enter and leave the honeynet.
- Sebek: This is a kernel module that captures data and keystokes.
- Swatch: The Swatch Monitors honeynet logs for outbound connections from the honeynet.

5.3. Data Analysis

The Roo also contains the Walleye web interface the principal tool for honeynet data analysis. Walleye is designed to provide analysts with a flow-oriented view of honeynet activity. This is done by associating the logs created by the honeywall's data capture tools with a network connection between two hosts. This can be viewed on the management system. The honeywall captures the details of the traffic going into the honeypots.

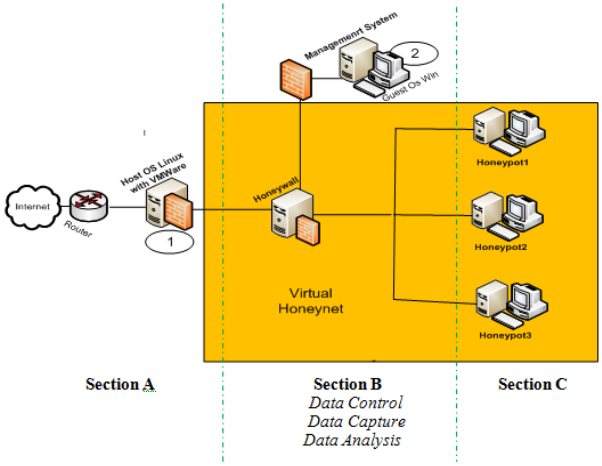

Figure 1. Architecture of the Virtual Honeynet

5.4. Section C

The honeypots are located in this section. The honeypots inside the honeynet do not have normal users who make outbound connections. Any connections that originate from inside the honeypots are most likely the result of a hacker or worm who has compromised a honeypot system. Sebek is installed on the honeypots providing information on malicious activities performed on the Honeypot. A hacker might use an encrypted connection to the Honeypot. In this case the network dump is not revealing its activities. Sebek is secretly logging keystrokes and event messages. It sends gathered data to a specified MAC address and hides this traffic to the intruder [10].

Sebek is a loadable kernel module for Linux that hooks into the stock read() system call in order to capture data sent to that function and then send it over a network link to another machine, such as a honeywall on a GenII honeynet deployment. The module can be loaded into the kernel in the same way as normal loadable modules, however once it is loaded, it can be “cleaned” and unloaded while still remaining active. This allows the Sebek module to run with a smaller chance of being detected by the attacker. The data logged from the read() call is sent to the external system via forged UDP packets with a specific (and nonsensical) source, destination, and port number. The external system listens for these specially crafted packets and logs them for future analysis. To prevent attackers who install network sniffers on the honeypot from noticing these packets coming from the system they are on, the network protocol stack of the kernel is modified to identify the specially crafted packets and drop them rather than pass them along to the sniffing software. This module allows a GenII honeynet to intercept encrypted communications as well as activities local to the honeypot that do not utilize the network [11], [12].

6. Case study

For the purpose of clarity, we looked at an instance of the database of students in a University. A typical example of students' record in the database in such a University will be in the form of Table 1. An attacker for instance may be motivated to hack into the network system of the university to modify the CGPA of a student named Bobola Kudirat Mary from 3.49 (Second Class Lower Credit) to 3.51 (Second Class Upper Credit). The format for retrieving data from the honeypot is predetermined to be in a table format. The intruder would be allowed to modify, delete, insert, print and copy but will not be allowed to query the database (This will further increase the workload and elongate the time spent on the honeynet as the intruder will have to transverse the record). Table 1 shows the record of students has it appears on the production server. It is copied from the production server to one of the honeypots for the intruder to access. Data may be presented as copied directly from the production server or may first be shuffled or modified before placing on the honeynet depending on how sensitive or important the data or information is. Figure 2 shows the activity flowchart of the honeynet responses to an intruder.

In this case, the copied record is directly placed as it is on the production system because the intruder is gunning for a specified candidate from the record of over thirty thousand students. Whenever the intruder performs a modification, deletion or insertion on Table 1, the record changes from the initial state zero to state one and the present state is saved after the intruder is done with the record. The intruder's MAC and IP address will then be assigned to the saved state one record last modified by the intruder. If another intruder accesses the same honeypot, the honeypot is designed to present the initial state zero of the record in the honeypot to the new intruder. If however an earlier intruder revisits, the last modified record in state 1 is presented to him. The intruder then gets the feel from the pot that the record is exactly the way he left it. This implies that every intruder is given a false monopoly of the supposedly hacked system. Furthermore, if the intruder wants to either print or copy, the copy or printed record would have been modified in such a way that what is displayed on the screen will be different from what was copied or printed as shown in Table 2. The four field that were reshuffled are: Surname, First name, Level and Semester 2.

This is done by modifying the print and copy ommand API with the addition of a shuffling algorithm. A motivated attacker would want to probe further by trying to come back into the honeypot. As we have earlier on said, the aim of this is to get the intruder further occupied by increasing his workload. If the intruder comes back into the honeypot, the honeywall checks if the MAC address has once been captured. If yes, the honeynet will presents the latest state of the modified record to that MAC address because the modified record was last assigned to it.

Table 1. Mirrored record of students from the production system to the honeynet

Mat. No	Surname	First Name	M. Name	Faculty	Department	Sex	Level	Semester 1	Semester 2	CGPA
CSC/08/112	Charles	Boye	David	Science	Computer	M	300	4.22	3.15	3.69
ENG/07/425	Bobola	Kudirat	Bunmi	Art	English	F	400	3.67	3.31	3.49
MEE/05/092	Oise	Ohikilome	Desmond	Engineering	Mechanical	M	200	4.57	4.85	4.71

Table 2. Re-shuffled record output for print and copy operations

Mat. No	Surname	First Name	M. Name	Faculty	Department	Sex	Level	Semester 1	Semester 2	CGPA
CSC/08/112	Boye	Bobola	David	Science	Computer	M	400	4.22	4.85	3.69
ENG/07/425	Kudirat	Oise	Bunmi	Art	English	F	200	3.67	3.15	3.49
MEE/05/092	Ohikilome	Charles	Desmond	Engineering	Mechanical	M	300	4.57	3.31	4.71

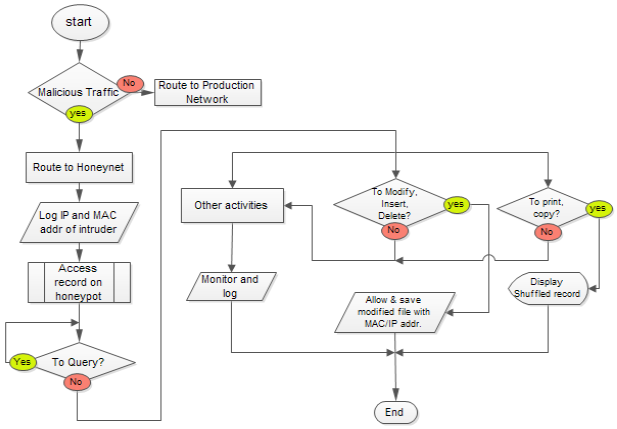

Figure 2. Activity Flowchart of the Honeynet Responses to an Intruder

The flowchart in Figure 2 gives an overview of this process. Only the executable file of the shuffling algorithm source code will be placed on the honeynet. Therefore access to the source code by an intruder is impossible. If the intruder probes further on why the copied file was different from what he saw, this further increases the workload of such intruder.

7. Conclusion and Future Work

Attackers that do face deceptive defenses do not go unscratched. In early experiments with deception defenses several results indicated that attackers were negatively impacted. Impacts included reduction in group cohesion, reduced desire to participate in attack activities, reduce enjoyment of activities, increased backtracking even when not under

deception, and reduction in performance levels [4]. Deception through IP address swaps and in TCP/IP layers have significantly improve the value of the honeypot. Data manipulation would further thicken the layers of deception as it is usually the aim of most of the attackers. Future work will look into the possibility of detecting if an attacker uses another system with different MAC/IP addresses to revisit a honeypot that he or she had once intruded. The probability that an intruder's personal behavioral pattern will affect the way he does his attack is not likely to be null.

8. Acknowledgements

Dahunsi Folasade received her BEng in Electrical Engineering from the University of Ilorin, Nigeria in 1999, the PhD in Network and Communication Engineering from University of the Witwatersrand in 2012. She is currently an academic staff of the Federal University of Technology, Akure. Her research interests include wireless network optimization, analysis and optimization of mobile services in developing countries, radiolocation techniques and geographical information system

9. References

[1]Anuar N. B., Zakaria O., and Wei, C., (2006) 'My Honeypot Through Web: The emerging of security application integration,' Issues in Informing Science and Information Technology, Vol. 3, pp 46-50.

[2]Doering, C. (2005) 'Improving Network Security with honeypots,' University of Darmstadt, Germany, Dept. of Informatics: Master's Thesis, 2005, pp. 4-7

[3]Spitzner, L., (2002) 'Honeypot Tracking Hackers,' Addison Wesley, pp. 271-287

[4]Cohen, F. (1998) 'A Note on the Role of Deception in Information Protection', IFIP-TC11, Computers and Security, Vol. 17, no. 6, pp. 483-506(24)

[5]Cai, J. Y. and Yegneswaran, V., Alfeld, C. and Barford, P., (2007) 'An Attacker-Defender Game for Honeynets' Journal of Combinatorial Optimization, Vol 20, pp 1- 4.

[6] Garg N., and Grosu D., (2007) 'Deception in Honeynets: A Game-Theoretic Analysis', Proceeding of the IEEE Workshop on Information Assurance, United states Military Academy, New York, 20-22nd June, pp. 1-7.

[7]Rammidi, G., (2009) 'Survey on Current Honeynet Research', Member of UNB Honeynet Project; Faculty of Computer Science; University of New Brunswick, Fredericton, Canada.

[8]The Honeynet Project, (2004), 'Know Your Enemy Learning About Security Threats', Addison Wesley, Boston, USA.

[9]Abbasi, F. H. and Harris, R., (2004). 'Building and Deploying a Gen. III Virtual Honeynet,' Massey university, New Zealand: Research Report, 159.799.

[10]Cohen, F. (2000) 'A Mathematical Structure of Simple Defensive Network Deceptions', IFIP-TC11, "Computers and Security", Vol. 19, No. 6, pp. 520-528, 2000.

[11]Spitzner, L., (2005). 'Know Your Enemy: Sebek,' The Honeynet Project,http://project.honeynet.org/ papers/sebek/ ((Access Date: 7 April, 2014).

[12]Honeynet Project, (2006) 'Know Your Enemy: The Honeywall' CDROM v.2.0; http:// www.honeynet.org; (Access Date: 7 April, 2014).

[13]Daisuke, M., Satoru, T., & Masaya, N., (2014). INTERCEPT: High-interaction Server-type Honeypot based on Live Migration. SimuTools , 142-152.

[14]Daratzi, P. (2012) 'Analysis And Testing Of Honeypots/Honeynet Systems', London: Masters Dissertation submitted to Kingston University.

[15]Navneet, K., and Lavleen, K. P. (2014). 'Honeypots: The Need of Network Security',. International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 5, No.5, pp 6098-6101.

[16]Niels, P., (2003). Honeyd: A Virtual Honeypot Daemon (Extended Abstract). 10th DFN-CERT Workshop, pp. 1-5, Hamburg, Germany.

[17]Thakar, U., Varma, S., and Ramani, A., (2005). HoneyAnalyzer – Analysis and Extraction of Intrusion Detection Patterns & Signatures Using Honeypot. The Second International Conference on Innovations in Information Technology (IIT'05), pp. 1-7.

