

Markov Chain Based Roaming Schemes for Honeypots

**Shashikala Tapaswi · Adnan Mahboob ·
Ajay Shankar Shukla · Ishan Gupta · Pranshu Verma ·
Joydip Dhar**

Published online: 1 May 2014
© Springer Science+Business Media New York 2014

Abstract The paper proposes a reactive roaming scheme for honeypots. The main aim of a honeypot is to capture the activities of the attacker. If the attacker detects honeypot on a system, its value drops. So, the concept of roaming honeypots is being proposed, to prevent the attacker from detecting the honeypot, which in turn increases the efficiency of honeypot and allows collecting rich data about activities of active attackers. The honeypot is shifted to another system which is most probable to be attacked within the network. The concept of Markov chain analysis is being used to detect the most probable system to be attacked based on the current status of the network. Further, using IP shuffling and services on/off concepts, honeypots roam on the network to the most probable system to be attacked using the threat score. Snort is used to capture data about the number of attacks on each of the nodes of the network and the data collected is then used as an input for Markov chain analysis to identify the most probable system where honeypot can be roamed/moved. The roaming scheme has been implemented for both high interaction honeypots and low interaction honeypots. The high interaction implementation helps in capturing in depth information on a shorter range of IP addresses, whereas the low interaction implementation is efficient in capturing information on a large range of IP addresses. The main advantage of this approach is that it predicts the

S. Tapaswi (✉) · A. Mahboob · A. S. Shukla · I. Gupta · P. Verma · J. Dhar
Atal Bihari Vajpayee-Indian Institute of Information Technology and Management,
Gwalior, Madhya Pradesh, India
e-mail: stapaswi@iiitm.ac.in

A. Mahboob
e-mail: adnanmahboob.iiitm@gmail.com

A. S. Shukla
e-mail: abhijay84@gmail.com

I. Gupta
e-mail: ishan230389@gmail.com

P. Verma
e-mail: pranshverma17@gmail.com

J. Dhar
e-mail: jdhar@iiitm.ac.in

frequency of attacks on the nodes of a particular network and takes a reactive step by starting the honeypot services on that particular node/system on the network.

Keywords High interaction honeypot · Low interaction honeypot · Snort · Sebek · Honeyd · Markov chain · Roaming honeypots · IP shuffling

1 Introduction

The last two decades have been marked by an immense growth in the use of the Internet and its capabilities. This immense growth has brought a wealth of valuable information at the fingertips of its user. But on the darker side, it has also led to a number of gateways for the hackers and attackers to get into a network or capture confidential information of the users. These attacks not only lead to monetary losses, but also are a cause of concern for the security and integrity of nations as the use of Information Technology has also expanded to the defense and administrative services of a nation. The attacks on high profile companies like Foxconn, Facebook, PayPal and the Sun grid [1–4] are an indication of the changing trends followed by the attacker and have raised the issue of a need for an intrusion detection scheme that can detect new attacks in contrast to the signature based techniques commonly used.

Case like Yahoo security breach which exposed 450,000 usernames and passwords from a site shows the loophole in existing security systems in high-profile companies [5]. It still indicates that a considerable amount of work needs to be done in the development of a security model. The model should be efficient enough in capturing the trends followed by the attackers and predict attacks based on these trends. Such a prediction based on the trends used by attackers is only possible if we survey the activities performed by the attackers. Thereby, necessary preventive measures can be designed; tools can be developed to stop the activities of attackers. Honeypots are efficient in capturing these activities which could be used for research work on security analysis. Honeypots also provide a good way of filtering information related to attacks as they process no production data and capture activities performed by the attacker along with the traffic directed towards or away from it [6]. Thus the tracked activities can be of great value for network security personnel in developing methods for detecting zero day attacks.

The paper presents an approach for roaming the honeypot in a network, in order to deceive the attacker. The purpose of roaming or shifting of honeypots is to achieve maximum valuable information from it. As it is known that honeypots are deployed for captivating the strategies and activities of the attacker. If an attacker detects the honeypot on a system, then the attacker will leave that system and may broadcast this information to other attackers also. This will defeat the purpose of a honeypot. So to prevent this event to happen, honeypots are not stabilized on a particular system, rather being roamed in a network. The concept goes with placing the honeypot on an IP address that is most probable to be attacked by an attacker based on the current state of the network. Markov chain based analysis is used to detect the probability of the most vulnerable system to decide on which machine the honeypot is to be shifted and then concepts of IP shuffling and service on/off are used to implement the roaming of the honeypots in the network. Snort is used to capture data about the number of attacks with the severity of attacks on each of the systems of the network and based on the number of attack the severity threat score is calculated as per Holsopple et al. [7] and input is provided for Markov chain analysis. This roaming mechanism enhances the capability of the honeypot in capturing high value data for analysis and detection of zero day attacks. The honeynet has

been implemented using Sebek client and server [6] for high interaction and honeyd [8] for low interaction implementation. Moreover the reactive roaming approach based on Markov chain presented in the paper assures that the honeypot is running on one of the most probable nodes for network attack at an instant [9]. The paper is organized as follows; related work already done in this area is given in Sect. 2, followed by a brief description about honeypots and the roaming honeypots in Sect. 3. Experimental setup details are mentioned in Sect. 4, Markov chain based honeypot roaming scheme implementation is provided in Sect. 5. The results and discussion are given in Sect. 6. Finally, Sect. 7 presents the conclusion and future work.

2 Related Work

In reference [10] impact of various attacks on systems and networks has been given based on four factors viz. CPU usage, memory usage, disk usage and network bandwidth utilization, a brief description of 146 attacks have also been mentioned.

Introduction on virtual honeypots and their utility in tracking the activities of the hackers is provided in [8]. Abbasi et al., have given an overview of a Generation III virtual honeynet which has been implemented using the Sebek client and server in [11]. A detailed overview of Sebek client and server is presented in [12] and this paper provides a description of the capabilities of the Sebek tool, its installation and configuration process.

Zargar et al. [13] have identified the impact of certain TCP/IP headers in identifying various attacks. They have identified the accuracy with which they could identify the attacks with the help of certain parameters and principal component analysis for data reduction and selection. In [14] the authors have given an algorithm that detects attacks based on patterns produced on parallel coordinates. Parallel coordinates have been formed for source address, destination address, destination port and packet length and then patterns for well known attacks have been identified. The authors propose that these patterns can be used for identifying zero day attacks if the traffic pattern produced by a zero day attack is similar to the pattern generated by one of the known attacks. Kim et al. [15] have proposed an algorithm that identifies attacks based on the flow header information and traffic patterns. The algorithm involves setting of various threshold values that vary in different networks.

In [16], a concept of roaming honeypots is put forward. A model is proposed which works by determining IP flow legitimacy by identifying early features of DoS attacks. This helps in roaming of honeypots and making them undetectable. In [17], Khattab et al., have proposed the concept of roaming honeypots to lessen the severity of the effects of DoS attacks. In these attacks the attacker systems misuses the victim's server causing great damage. The roaming honeypots scheme detects and filters attack traffic from outside as well as inside a firewall. A proactive roaming scheme is proposed in [18], which make use of backward hash chains. The use of backward hash chains has an overall impact on the overhead of the system. The paper presents a much simpler approach to implement reactive roaming honeypots based on Markov chain analysis, and reduces the overhead as in the reactive roaming approach the honeypot is roamed/shifted to a machine where the severity of attack is more.

3 Brief Introduction of Honeybots

Honeybot is a tool that is used to track attacker's activity. It lures the attacker to attack the system. Honeybots have no production value in fact their value lies in being probed, attacked or compromised. The main goal in deploying a honeybot is to get it attacked. It may act as a

production honeypot within an organizations environment to help mitigate risk or as research honeypot which adds value to research in computer security by providing a platform to study the threat. An important point about honeypots is that it is not a solution to network attack but is a tool to analyze network traffic so that data can be captured about zero day attacks [19].

Low-interaction honeypots emulate services, network stacks, or other aspects of a real machine. They allow an attacker a limited interaction with the target system and allow us to learn mainly quantitative information about attacks. Low-interaction honeypots can primarily be used to gather statistical data and to collect high-level information about attack patterns. An attacker is not able to fully compromise the system since he/she interacts just with a simulation. Low-interaction honeypots construct a controlled environment and thus the risk involved are limited [8]. In our approach *honeyd* is used for the implementation of low interaction honeypots.

High-interaction honeypots offer a complete system to the attacker. The system deployed as a high interaction honeypot does not emulate any services, base operating system or any functionality. This approach gives an opportunity to learn about the tools, tactics and motives of the attacker. Although these honeypots give complete details about the attacker activity, but implementing them requires high cost and also a large amount of time is required [8]. In our approach we have implemented a virtual high interaction honeypot using *Sebek*, as it gives an opportunity to do an in depth analysis of network traffic and also the cost involved is relatively low. In the experiments virtual honeynet has been used and is described next.

3.1 Virtual HoneyNet

A HoneyNet is a network of honeypots placed behind a reverse firewall. The reverse firewall is used to capture all inbound and outbound traffic from the honeypot network. The reverse firewall also limits the amount of inbound and outbound traffic flowing through the honeypots so that the systems may not be able to infect other systems on the network [20].

A virtual honeynet is a honeynet implemented using virtualization software. They are known as virtual honeynets because different operating systems used for implementing the honeypots on the host system give the appearance of independent computers [21]. Virtual honeynets provide advantages of reduced cost and easier management. Also, because of virtualization the attacker deals with the virtual operating system only and host operating system remains safe from the attacker. It may happen that the attacker is advanced enough to cause harm to the host operating system also. Therefore, virtualization provides just a level of security while implementing honeypots. Implementing honeypots on a single computer may provide complete control of the virtualization software to the attacker and may lead to compromise the entire honeynet.

3.2 Roaming Honeypots

The purpose of creating honeypots is to trap activities of the attackers. But they are unable to fulfill their purpose efficiently because they are deployed on fixed systems which are detectable. With fixed systems, honeypots will not be able to collect information about hackers attacking active computers. After some time any attacker may detect honeypot on the system and broadcast this information to attacker community. After knowing this no attacker will try to attack on that honeypot and the honeypot becomes useless. So, the concept of roaming honeypots is being proposed. Because of this mechanism, honeypots keep on roaming on different locations continuously within a network, thereby becoming unpredictable, and remains undetectable by the attacker.

Table 1 Specifications of hardware used

Feature	Product	Specifications
Host operating system	Microsoft Windows 7 Professional @[27]	HW Vendor: Intel(R)core i5 Processor:3.20 GHz RAM:2 GB NIC: Gigabit Ethernet
Guest operating system 1	Linux, Roo-Honeywall 1.4 based on Cent OS Release 5	Single Processor Virtual Machine RAM:512 MB Storage:32 GB NIC 1:100 Mbps Bridged interface eth0 NIC 2:100 Mbps host-only interface eth1 NIC 3:100 Mbps Bridged interface eth2
Guest operating system 2	Linux Ubuntu 7.10 (Gutsy Gibbon) with hardy updates	Single Processor Virtual Machine RAM:512 MB Storage:32GB NIC: Bridged
Guest operating system 3	Microsoft Windows XP Professional ®	Single Processor Virtual Machine RAM:512 MB Storage:32GB NIC: Bridged
Virtualization Software	VMware workstation	VMware workstation 6.5.1 for windows
Honeywall	Roo	Roo-1.4
IDS	Snort [28]	
IPS	Snort_inline [28]	
Data Capture Tool	Sebek [6]	

Roaming of honeypots refers to transferring the data capture daemon from one node of the network to another based on various parameters such as network traffic, number of attacks, severity of attack etc. This approach not only helps us in capturing the activities of attackers by using the capabilities of the honeypot but also helps in increasing the probability of capturing data on a system that has the highest probability of being probed at a particular instance of time.

4 Experimental Setup

In order to perform the experiments, we have implemented a honeynet which consists of two virtual machines, one running Ubuntu server 7.10 [22] and the other running Windows XP professional [23]. Sebek client [24] is configured on both the virtual machines which are setup using VM-ware workstation 6.5.1 [25]. Another virtual machine is setup using Roo 1.4-CDROM which is based on CentOS version 5 and is used to configure the honeywall [26]. The honeywall acts as a gateway to the honeynet and can be configured to allow inbound and outbound traffic. The honeywall also provides a management interface named Walleye. The management interface is a Graphical User Interface (GUI) that facilitates the management and configuration of the honeywall.

A detailed overview of the tools and the hardware used in the experiment is given in Table 1. The architectural details of the high interaction honeypot scheme implemented is shown in Fig. 1.

For experimentation VMware virtualization software has been used which emulates a complete x86-based computer system and allows the user to run more than one operating system over a host operating system. The operating system that runs inside the host machine

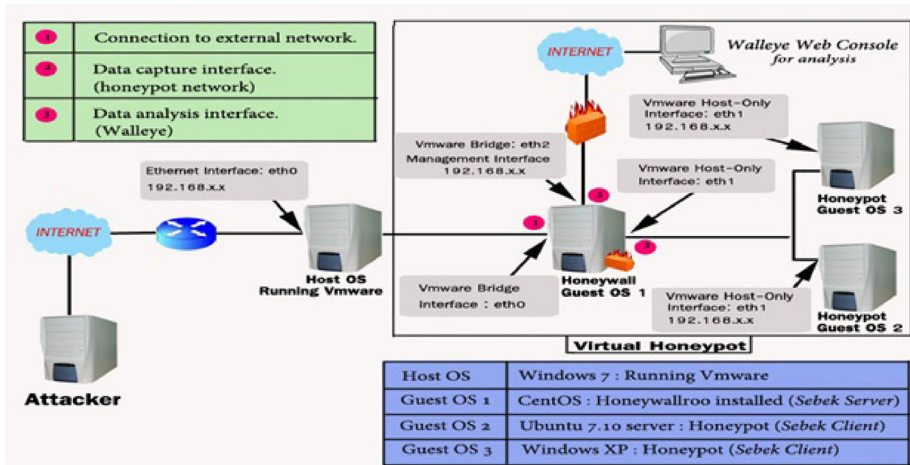


Fig. 1 Implemented architecture

is usually known as the guest operating system [10]. VMware acts like a complete emulator and provides a virtualized set of hardware to the guest operating system. It is also possible to configure the hardware according to the needs of the guest operating system. Three types of network adapters can be configured on VMware.

- Bridged: The guest machine is on the same network as the host machine.
- Host-Only: It provides a network connection between the virtual machine and the host computer. It is helpful in creating an isolated virtual network.
- NAT: NAT is used to set up a virtual private network between the guest operating systems which have access to the internet [25].

4.1 Sebek Server and Sebek Client

A sebek server has been used to capture data from the honeypots (i.e. systems that has sebek client installed on them) so that an in depth analysis of the data can be performed afterwards [6]. In our experimental setup we have used the Roo Honeywall CDROM which has a pre-installed Sebek server on it. The Roo1.4 honeywall is based on Cent OS version 5 instead of Fedora (used in the previous versions), as CentOS has a longer support period. After installing Roo1.4 on the virtual machine we configured it according to our network. The Roo Honeywall provides various pre-installed tools for data analysis which are listed below.

1. Snort [28]. Intrusion Detection System (IDS).
2. Snort inline [28]. Intrusion Prevention System (IPS).
3. Argus, Pof. Passive OS fingerprinting tool.
4. Tcpdump [29]. Viewing of packet headers.
5. Hflow2 [30]. A data coalescing tool for Honeyynet data analysis.
6. Walleye. Web based interface for Honeywall configuration, administration and data analysis.
7. Sebek [31]. Sebek is a data capture tool designed to stealthily capture attacker’s activities

Configuration of Honeywall requires three network adaptors. In our setup we have used VMware to configure the three network adaptors. The description on usages of all the three network adaptors is given below:

1. Eth0: The eth0 Ethernet interface is used to connect the honeywall to the external network or the internet. This interface has a hidden IP address so that it cannot be identified on the external network. eth0 uses bridged networking.
2. Eth1: The eth1 Ethernet interface is used to connect the honeywall running sebek server to the honeynet containing honeypots running sebek client. This interface also provides external connectivity to the honeypots through the honeywall. This interface does not have an IP address. The eth1 interface is a host-only interface.
3. Eth2: The eth2 Ethernet interface is used as the management interface. This interface is helpful in remote management of the honeywall. The IP address of the eth2 interface is used to access the Walleye interface using https protocol. The eth2 interface uses bridged networking.

Sebek client is a kernel module that needs to be installed on the systems those are to be configured as honeypots. All inbound and outbound traffic from the sebek client passes through the honeywall or the Sebek server. The main purpose of the sebek client is to capture all data about the activities performed on the honeypot and then send them to the sebek server. In our experiments we have deployed two honeypots. One of the honeypot is implemented on Ubuntu server 7.10 and the other on Windows XP professional. We downloaded the file `sebek_disable_raw_socket_replacement-lin26-3.2.0b-bin.tar.gz` for Ubuntu and `Sebek-Win32-3.0.5.zip` file for installation on Windows XP from projects.honeynet.org site. The virtual machines used to implement the sebek clients are configured to use host-only networking in order to form a virtual network of honeypots that can be accessed only through the honeywall.

4.2 Walleye

Walleye is a management interface for administration of honeypots as well as for the analysis of data collected on activities performed on the honeypot. Walleye runs through the eth2 interface of the Roo-honeywall. It provides a GUI to perform all management and configuration activities on the honeywall. The Walleye provides options to view flows, gives information about packet header and also provides an interface for making changes to the configuration files of the honeywall. The Walleye GUI can be accessed through a browser with the uniform resource locator (URL) `https://ip-address/`, the IP address used is the IP address of eth2 interface of the honeywall which is the remote management interface for walleye.

5 Proposed Honeybot Roaming Schemes

The roaming scheme works by activating the *sebek* daemon in case of high interaction honeypot or *honeyd* daemon for the low interaction honeypot. The basic approach can be explained in terms of the threat level which is predicted with the help of markov chain analysis in the proposed setup. As markov chain analysis helps in predicting a future state, so, it has been chosen for predicting the threat level on the systems in a network. The approach makes use of a threat variable 's' which changes from 0 to 1 on a system when the threat level on the particular system is greater than the threat level on all other systems and correspondingly the value of 's' is changed back to 0 on all other devices, after the honeypot is shifted. The block diagram of the proposed scheme is shown in Fig. 2 and the details of the use of markov chain analysis method for prediction of threat level are discussed next.

The threat level is predicted, based on the priority of probes given by Snort for a particular IP address of the network. For this purpose a machine running Snort is deployed on the

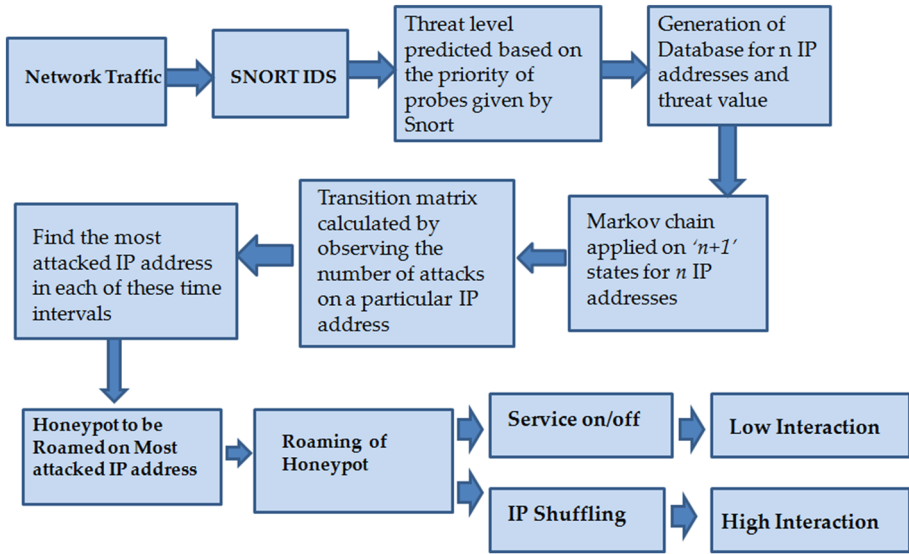


Fig. 2 Proposed approach of a reactive roaming honeypots

network to be monitored and corresponding switches take place based on the most probable system to be attacked. Snort classifies the alerts on the scale of 1–3 with 1 being the most severe [32], using snort priorities and number of alert, we calculate the threat score and then Markov chain approach is used to implement roaming based on these threat scores. The threat score is calculated as follows [7]:

$$Threat\ score = \sum_{n=1}^3 (alertcount_n) \times (2^{3-n} - 1) \tag{1}$$

where n is priority of attack and $alertcount_n$ is the total number of attacks of priority n . The Markov chain has to be applied to $n+1$ states for n IP addresses under consideration where n states correspond to each of the n IP addresses being under attack and $(n + 1)^{th}$ state corresponds to no IP address under attack. The most attacked IP address is found by observing the threat score on a particular IP address for 50 time intervals. Then the transition probabilities are calculated using the following two formulae:

$$P_{ij} = Prob(X_{n+1} = j | X_n = i) \tag{2}$$

$$Prob(X_{n+1} = j | X_n = i) = \frac{No.of\ event((X_{n+1} = i) \cap (X_n = j))}{Total\ event(X_n = i)} \tag{3}$$

where P_{ij} is called transition probability which represents the probability at time t , the system being in state i , passes to state j at time $t+1$. The events $X_n = i$ and $X_{n+1} = j$ respectively expresses that the system is in state i at the moment n and the system is in the state j at the moment $n+1$. Then the event $(X_n = i) \cap (X_{n+1} = j)$ expresses that the system is in state j after having been in state i .

The transition probability is then used to find the most probable IP address for attack along with the initial probability matrix for attacks on each IP address. The honeypot is then shifted on the most probable IP address for attack so that minimum damage could be caused to the

Table 2 Results of the number of attacks per IP addresses collected from Snort IDS

IPAddress	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
192.168.39.81	0	0	10	8	98	82	0	0	6	122	10	40
192.168.39.82	0	6	6	12	10	122	0	0	0	24	0	22
192.168.39.83	0	12	4	42	116	64	22	0	6	56	0	10
192.168.39.84	0	6	8	30	100	126	22	0	0	172	0	6
IPAddress	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
192.168.39.81	12	38	26	78	34	0	0	12	24	12	10	0
192.168.39.82	6	24	70	20	34	0	20	28	56	24	64	0
192.168.39.83	14	12	14	60	20	0	16	6	0	40	34	0
192.168.39.84	8	0	54	26	20	0	6	48	0	6	98	0
IPAddress	P25	P26	P27	P28	P29	P30	P31	P32	P33	P34	P35	P36
192.168.39.81	8	0	200	0	0	8	42	10	50	16	78	68
192.168.39.82	18	12	284	0	0	0	48	22	6	12	4	12
192.168.39.83	44	6	122	0	0	10	36	0	30	20	122	6
192.168.39.84	128	12	22	0	0	8	40	6	0	30	26	52
IPAddress	P37	P38	P39	P40	P41	P42	P43	P44	P45	P46	P47	P48
192.168.39.81	0	0	226	132	0	120	152	0	44	168	302	0
192.168.39.82	0	74	0	238	0	228	16	12	10	16	206	0
192.168.39.83	0	52	6	262	0	116	38	24	44	94	242	0
192.168.39.84	0	78	42	72	0	142	40	16	0	0	22	0
IPAddress	P49	P50										
192.168.39.81	0	0										
192.168.39.82	58	0										
192.168.39.83	264	6										
192.168.39.84	62	12										

production systems and maximum data related to attacker’s activities could be collected on the honeypot.

5.1 Data Collection

In our experiments, we have setup a subnet in the institute ABV-Indian Institute of Information Technology and Management, Gwalior and we observed number of attacks for four IP addresses (192.168.39.81, 192.168.39.82, 192.168.39.83, 192.168.39.84), the results are presented over 50 time periods of time duration ‘t’ and duration of each time period are considered as 1 min (the time period can be increased or decreased). The threat score observed on particular IP addresses is shown in Table 2.

From Table 2, we find the most attacked IP address for a time period with an assumption that in starting periods (say P1, P2, P3, and P4) no IP address should have same threat score. If during any period all IP addresses have zero threat score, i.e., no IP address is under attack, the assumption is made that the most-attacked IP address is noted as 0. If two (or more) IP addresses have same threat score in any period, we assume that the most-attacked IP address is the one which have been attacked the least during the former attack periods. Table 2 shows

Table 3 List of the most-attacked IP address

Periods	P1	P2	P3	P4	P5
Most attacked IPAddress	null	192.168.39.83	192.168.39.81	192.168.39.83	192.168.39.83
Periods	P6	P7	P8	P9	P10
Most attacked IPAddress	192.168.39.84	192.168.39.83	null	192.168.39.83	192.168.39.84
Periods	P11	P12	P13	P14	P15
Most attacked IPAddress	192.168.39.81	192.168.39.81	192.168.39.83	192.168.39.81	192.168.39.82
Periods	P16	P17	P18	P19	P20
Most attacked IPAddress	192.168.39.81	192.168.39.82	null	192.168.39.82	192.168.39.84
Periods	P21	P22	P23	P24	P25
Most attacked IPAddress	192.168.39.82	192.168.39.83	192.168.39.84	null	192.168.39.84
Periods	P26	P27	P28	P29	P30
Most attacked IPAddress	192.168.39.81	192.168.39.82	null	null	192.168.39.83
Periods	P31	P32	P33	P34	P35
Most attacked IPAddress	192.168.39.82	192.168.39.82	192.168.39.81	192.168.39.84	192.168.39.83
Periods	P36	P37	P38	P39	P40
Most attacked IPAddress	192.168.39.81	null	192.168.39.84	192.168.39.81	192.168.39.83
Periods	P41	P42	P43	P44	P45
Most attacked IPAddress	null	192.168.39.82	192.168.39.81	192.168.39.83	192.168.39.81
Periods	P46	P47	P48	P49	P50
Most attacked IPAddress	192.168.39.81	192.168.39.81	null	192.168.39.83	192.168.39.84

the results of the threat score for each of the IP addresses, collected from Snort Intrusion Detection System(IDS). From Table 2, the most-attacked IP address is found for each time period and is shown in Table 3.

5.2 Markov Chain Analysis Modelling

In our experiments, there are five states to model the Markov chain. There are four states corresponding to the four IP addresses which are our pre-defined IP addresses and the fifth state is when no IP address is under attack.

From Eq. (2), Eq. (3), and data of Tables 2 and 3, we can calculate transition probability matrix at any time instant and is given as follows:

$$P = \begin{pmatrix} 0.231 & 0.231 & 0.207 & 0.077 & 0.154 \\ 0.375 & 0.125 & 0.125 & 0.125 & 0.250 \\ 0.333 & 0.083 & 0.167 & 0.250 & 0.167 \\ 0.428 & 0.143 & 0.286 & 0.000 & 0.143 \\ 0.000 & 0.222 & 0.444 & 0.222 & 0.111 \end{pmatrix} \tag{4}$$

Initially in the network there is no IP address (system/node) is under attack and we assume that we have no IP address under attack.

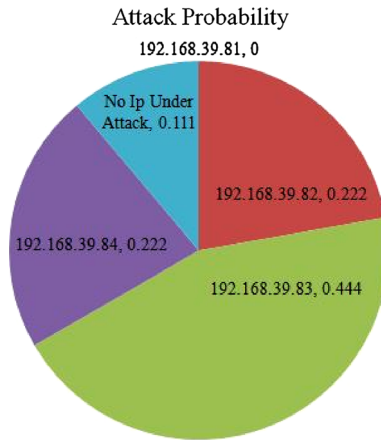


Fig. 3 Details of attack on IP addresses

So, we get an initial attack probability as

$$A(0) = [0 \ 0 \ 0 \ 0 \ 1]$$

According to the study that the probabilistic state of the system after time duration 't', we can predict:

$$A(t) = A(0)P$$

Thus, we get

$$A(t) = [0.000 \ 0.222 \ 0.444 \ 0.222 \ 0.111]$$

So, it can be predicted that IP address 192.168.39.83 is most-attacked. Figure 3 shows the probability of attacks on the four IP addresses in percentage at a time instant, and helps in roaming the honeypot to the IP address which is likely to be attacked the most.

5.3 Roaming Schemes

The roaming schemes have been implemented using two techniques. (i) IP Shuffling (ii) Service ON/OFF. The IP shuffling and Service ON/OFF techniques are used for the implementation of roaming based on the probabilities calculated by the markov chain analysis method. The details of these techniques are described below.

5.3.1 Honeypot Roaming Using IP Shuffling

In order to implement roaming of honeypot first the most probable IP address for the attacks was identified using markov chain analysis, then it was freed by modifying the NAT table at the router. The freed IP was then assigned to the honeypot system so that all the data pertinent to the attack is collected in the honeypot system. The main advantage of this approach is that all the data could be collected at a centralized location. Although there may be a small amount of delay in assigning the IP address to the honeypot system.

Table 4 Predictions made by Markov chain analysis for roaming

Time period (mins)	Attack probability predicted by Markov analysis					Current honeypot	Predicted honeypot
	192.168.39.81	192.168.39.82	192.168.39.83	192.168.39.84	No IP under attack		
Initial state	0.000	0.222	0.444	0.222	0.111	No honeypot running	192.168.39.83
0–10	0.271	0.163	0.224	0.159	0.182	192.168.39.83	192.168.39.81
10–20	0.246	0.182	0.236	0.161	0.175	192.168.39.81	192.168.39.81
20–30	0.207	0.191	0.186	0.291	0.124	192.168.39.81	192.168.39.84
30–40	0.192	0.177	0.283	0.235	0.112	192.168.39.84	192.168.39.83
40–50	0.175	0.264	0.213	0.179	0.168	192.168.39.83	192.168.39.82
60–70	0.206	0.189	0.218	0.183	0.204	192.168.39.82	192.168.39.83
70–80	0.163	0.197	0.223	0.174	0.243	192.168.39.83	192.168.39.83
80–90	0.209	0.218	0.192	0.172	0.208	192.168.39.83	192.168.39.82
90–100	0.243	0.176	0.183	0.204	0.194	192.168.39.82	192.168.39.81

5.3.2 Honeypot Roaming Using Service ON/OFF

This scheme activates the data capture daemon of the honeypot on a machine that is most probable for attack and turns off the daemons at all other nodes of the network. In the case of low interaction honeypot this technique is performed by manipulating the *honeyd* daemon, whereas, in the case of high interaction honeypot the service to be activated or deactivated is the *sebekd* daemon. The advantage of this daemon is that the switching takes place almost instantaneously through a script that runs on the system itself, whereas the data collected in this case is distributed on several systems.

6 Results and Discussion

The main advantage of this approach is that it strengthens the ability of the honeypot to capture data of high value. Compared to the static honeypot setup this approach is less likely to be fooled by the attacker, because even if the attacker detects the honeypot and stops to attack, the honeypot has moved to a system which is most likely to be attacked with the predictive roaming scheme and still the attacker activities be followed and captured. Also the markov chain analysis model used to predict the most valuable system for collection of data is a well established and vastly accepted model and thus the roaming method predicts the next possible attack IP address location.

The approach is based on Markov chain analysis for prediction of probable IP address for attacks. The predicted IP address to which the honeypot is roamed in each of the four IP periods is shown in Table 4 along with the attack probabilities for each of the four IP addresses considered. The probable IP address for attack predicted by markov chain analysis is then used as the IP address of the honeypot with the help of Service ON/OFF or IP shuffling. We have used a time interval of 10 min to roam the honeypots. This setup has been deployed for a period of more than 2 months and the data captured is approximately 5 GB, presenting all the results is out of the scope of the paper, thus partial results have been presented in the paper, to illustrate the proposed roaming scheme.

Table 5 Comparison of static and roaming honeypot schemes

Time period (h)	Static honeypot	Roaming honeypot	
		IP shuffling	Service ON/OFF
0–1	1,264	1,572	1,620
1–2	2,524	2,068	2,042
2–3	1,004	2,172	2,231
3–4	931	2,572	2,681
4–5	837	2,891	2,875

In order to compare the proposed approach with a static honeypot setup we deployed a static honeypot and a roaming honeypot using IP shuffling as well as Service ON/OFF approach for the same instant of time on the same network and then compared the number of flows to/or from the honeypot in both the cases. The data pertaining to the number of network flows was collected using the Walleye interface in the case of high interaction honeypot and using the honeyd log files analysis using a python script in case of low interaction honeypot. Although the number of flows is a naive method and does not guarantee that each flow is a probable attack. In order to compensate for this disadvantage we kept the network idle for the test period which is 5 h so that most of the flows can be thought of as malicious. The number of flows captured by the honeypot in each of the three approaches are shown in the Table 5.

It can be inferred from Table 5 that the number of flows in the initial hour were moderate in the case of static honeypot which then increased considerably in the second hour. Most probably this was the time when the static honeypot was seen as a production system by the attacker. Then a steep fall in the number of flows in the third hour and then the considerable decrease in the flows can be attributed to the discovery of honeypot by the attacker and the attacker moving on to some other system. As many attackers usually communicate such information amongst them the value of the static honeypot decreases considerably once it has been discovered by an attacker. Whereas in the case of roaming honeypots the number of flows related to the honeypot at any instant of time kept increasing as the honeypot follows the attacker and as the attacker detects a honeypot on one of the IP address the honeypot moves on to some other IP address as predicted by markov chain model thus following the attacker.

7 Conclusion and Future Work

The approach proposed in this paper will help in understanding the attacker's activities in more detail as the location of the honeypot remains undetectable from the attacker and it captures the maximum details of attacker's activities. The Markov chain analysis is one of the most accepted approach used as a predictive model and is used to denote a sequence of events based on the transition probability matrix at a given time. This makes it possible to know the most probable IP address to be attacked during a given period based on the threat scores. It helps in predicting the next location where an attacker may move and thus can be of great use in following the attacker's motives. Thus the contribution of this paper is in increasing the efficiency of the honeypot system in terms of capturing the attacker's activities and can be of great value in the detection of zero day attacks. The work done in this paper proposes a generalized approach for roaming honeypots in conjunction with markov chain analysis which can be combined with other decision support techniques for prediction of

honeypot location as a future work. The data captured can be analysed for types of attacks and to develop the tools to secure the network.

Acknowledgments The authors would like to thank Deity, New Delhi and Atal Bihari Vajpayee Indian Institute of Information Technology and Management, Gwalior for extending the support for the project on Development of Reactive Roaming Scheme for Honeypots.

References

1. Pauli, D. (2010). Paypal hit by ddos attack after dropping wikileaks. <http://www.zdnet.com/news/paypal-hit-by-ddos-attack-after-dropping-wikileaks/489237>.
2. Shankland, S. (2006). Hackers hit twitter and facebook. <http://news.bbc.co.uk/2/hi/8188201.stm>.
3. Shankland, S. (2006). Sun grid hit by network attack. http://news.cnet.com/Sun-Grid-hit-by-network-attack/2100-7349_3-6052968.html.
4. The Times of India, India.: Apple supplier foxconn's website hacked. (2012). <http://timesofindia.indiatimes.com/tech/news/internet/Apple-supplier-Foxconn-website-hacked/articleshow/11846659.cms>.
5. Yahoo security breach shocks experts. (2012). <http://www.csoonline.com/article/710804/yahoo-security-breach-shocks-experts>.
6. Sebek home page. (2007). <http://old.honeynet.org/tools/sebek/>.
7. Holsopple, J., Yang, S. J., & Sudit, M. (2006). Tandi: Threat assessment of network data and information. In *Proceedings of SPIE: The International Society for Optical Engineering* (vol. 6242). www.scopus.com. Cited By (since 1996):10.
8. Provos, N., & Holz, T. (2007). Virtual honeypots: From botnet tracking to intrusion detection. Reading: Addison Wesley Professional.
9. Kimou, K. P., Barry, B., Babri, M., Oumtanaga, S., & Kadjo, T. L. (2010). An efficient analysis of honeypot data based on Markov chain. *Journal of Applied Sciences*, 10, 196–202.
10. Biswas, A. (2008). Impact analysis of system and network attacks. Master's thesis, Utah State University, Netherlands. <http://digitalcommons.usu.edu/etd/199>.
11. Abbasi, F. H., & Harris, R. J. (2009). Experiences with a generation iii virtual honeynet. In *Proceedings of the IEEE Australasian telecommunication networks and applications conference ATNAC* (pp. 1–6). doi:10.1109/ATNAC.2009.5464785.
12. Honeynet Project. (2003). Know your enemy: Sebek a kernel based data capture tool. <http://www.honeynet.org>.
13. Zargar, G. R., & Kabiri, P. (2010). Selection of effective network parameters in attacks for intrusion detection. In *ICDM* (pp. 643–652).
14. Choi, H., Lee, H., & Kim, H. (2009). Fast detection and visualization of network attacks on parallel coordinates. *Computers & Security*, 28(5), 276–288.
15. Kim, M.-S., Kong, H.-J., Hong, S.-C., Chung, S.-H. & Hong, J. W. (2004). A flow-based method for abnormal network traffic detection. In *Network operations and management symposium, 2004. NOMS 2004. IEEE/IFIP* (vol. 1, pp. 599–612). doi:10.1109/NOMS.2004.1317747.
16. Zeng-gang, X., & Xue-min, Z. (2010). Design and implementation of a new roaming honeypots defense strategy. In *2010 The 2nd international conference on computer and automation engineering (ICCAE)* (Vol. 4, pp. 535–538). doi:10.1109/ICCAE.2010.5451592.
17. Khattab, S. M., Sangpachatanaruk, C., Mosse, D., Melhem, R., & Znati, T. (2004). Roaming honeypots for mitigating service-level denial-of-service attacks. In *2004 Proceedings 24th international conference on distributed computing systems* (pp. 328–337). doi:10.1109/ICDCS.2004.1281598.
18. Kalashnikova, I., & Shinde, R. (2007). Optimizing the location of honeypots in a finite computer network. www.stanford.edu/rbs/Projects/Optimal_honeypot_location.pdf.
19. Spitzner, L. (2002). *Honeypots: Tracking hackers*. Reading: Addison Wesley.
20. Levine, J., LaBella, R., Owen, H., Contis, D., & Culver, B. (2003). The use of honeynets to detect exploited systems across large enterprise networks. In *Proceedings of the conference on IEEE Systems man and cybernetics society Information assurance workshop* (pp. 92–99). doi:10.1109/SMCSIA.2003.1232406.
21. Honeynet Project. (2007). Know your enemy: defining virtual honeynets. <http://old.honeynet.org/papers/virtual/>.
22. Ubuntu server 7.10. <http://old-releases.ubuntu.com/releases/7.10/>.
23. Windows XP professional. <http://windows.microsoft.com/en-IN/windows/xp-downloads>.
24. Sebek client. <https://projects.honeynet.org/sebek>.

25. VMware: Vmware support. (2011). <http://www.vmware.com/support/>.
26. Roo-honeywall. <https://projects.honeynet.org/honeywall/>.
27. Windows 7 professional. <http://windows.microsoft.com/en-us/windows7/products/home>.
28. Snort documentation. (2011). <http://www.snort.org/docs>.
29. TCPDUMP: Tcpcap and libpcap. (2010). <https://projects.honeynet.org/hflow>.
30. Honeynet Project: Hflow2 (2008). <https://projects.honeynet.org/hflow>.
31. Stallings, W. (2007). Intrusion detection. In *Computer security: Principles and practice* (4th edn). Pearson Education.
32. Caswell, B., & Beale, J. (2004). Snort 2.1 Intrusion detection. Addison Wesley Professional, Reading



Shashikala Tapaswi is a Professor in Information Technology at of AtalBihari Vajpayee-Indian Institute of Information Technology and Management, Gwalior (M.P.) India. She has obtained her Ph.D. (Computer Engineering) from Indian Institute of Technology, Roorkee, India in 2002, M.Tech. (Computer Science) from University of Delhi, India in 1993 and B.E. (Electronics Engineering) from Jiwaji University, Gwalior, India in 1986. Her primary research areas of interest are Artificial Intelligence, Neural Network, Fuzzy Logic, Digital Image Processing, Computer Networks, Mobile Networks and Network Security.



Adnan Mahboob has completed Integrated Postgraduate course (B.Tech + M.Tech(IT)) in 2013 from AtalBihari Vajpayee-Indian Institute of Information Technology and Management, Gwalior (M.P.) India. His research interests are Network Security and Cloud Computing. He has worked on Issues related to Resource Allocation as a part of his M.Tech Thesis Project. He has worked on development of roaming schemes for High Interaction Honeybots as a part of his B.Tech Project.



Ajay Shankar Shukla has completed B.Tech in CSE in 2009. He is presently working as Senior project fellow on project “Digital Information resource facility” at CSIR—National Institute of Science Communication and Information Resources (NISCAIR) and has worked as Research Engineer on project “development of reactive roaming schemes for honeypots” and as Research Associate on project “Information Security Education Awareness” sponsored by DeitY Govt. of India New Delhi.



Ishan Gupta has completed Integrated Postgraduate course (B.Tech + M.Tech(IT)) at AtalBihari Vajpayee-Indian Institute of Information Technology and Management, Gwalior (M.P.) India. His research interests are Mobile Network, Cloud Computing and Network Security. He has worked on Security Enhancements in virtualization environment as a part of his M.Tech Thesis Project. He has worked on development of roaming schemes for High Interaction Honeybots as a part of his B.Tech Project.



Pranshu Verma has completed Integrated Postgraduate course (B.Tech + M.Tech(IT)) in 2013 from AtalBihari Vajpayee-Indian Institute of Information Technology and Management, Gwalior (M.P.) India. His research interests are Network Security and Cloud Computing. He has worked on Issues related to Task Scheduling as a part of his M.Tech Thesis Project. He has worked on development of roaming schemes for High Interaction Honeybots as a part of his B.Tech Project.



Joydip Dhar is an Associate Professor in the area of Applied Sciences (Mathematics) of the Institute. Dr. Dhar has done his Ph.D. from IIT, Kanpur in 1997 after completing his M.Sc. (Mathematics) from Visva-Bharati University, Santiniketan. He has been associated with the teaching profession and research for the past 14 years and has also published about 60 papers in his areas of interest and proficiency. Under his guidance 2 Ph.D., 8 M.Phil, 9 M.Tech and 8 MBA theses completed. Currently 6 Ph.D., 4 M.Tech, 7 MBA students are pursuing their thesis. He is a life member of many professional societies ISTE, IMS, ISM-MACS and annual member of American Mathematical Society (AMS). Under his mentorship a team of three students of ABV-IIITM, Gwalior after winning ACM-ICPC IIT Kanpur chapter represented India to participate in the prestigious ACM-ICPC world finals at Stockholm, Sweden from 16–22 April, 2009. He has conducted National Conference on “Mathematical Modelling and Simulation” from 9–11 January, 2009, AICTE-ISTE sponsored, short term course on “Modelling, Simulation

of Engineering and Environmental Systems” from 15–19 March, 2004 and MHRD sponsored, short term course on “Modelling, Simulation in Science and Engineering” from 08–12, 2009. His current area of research is Mathematical Modelling and Simulation of Environmental, Managerial and Engineering systems.