International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(2): 437-458
The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085)

Guidelines for Collecting and Centralizing Network Digital Evidences

Mohammed Abbas[1], Azizah Abdul Manaf[2,] Elfadil Sabeil[3]
[1,3] Faculty of Computer & Information Systems, Universiti Teknologi Malaysia (UTM), Johor Baru, Malaysia
[1] mabbsaleh@gmail.com, [3] alfadil.sabeel@yahoo.com
[2] Advanced Informatics School (AIS), Universiti Teknologi Malaysia (UTM), Kuala Lumpur, Malaysia [2] azizah07@citycampus.utm.my

ABSTRACT

The digital evidences emphatically are commonly considered as a backbone for the forensic body in order to deliver a reliable investigation when a breach occurred since a forensic basically based on them. However, there are challenges harming the integrity and reliability of these digital evidences such as removing or tampering with them since most of equipments of production environment are accessible to intruders because they normally assign an Internet Protocol (IP). Therefore, a hidden mechanism namely Honeynet Architecture which located in the middle between the equipments and intruders is proposed for the sake of overcoming these weaknesses. In this paper, firstly the proposed mechanism for collecting and centralizing network digital evidences is studied and investigated as well, and then a comparison among the proposed solutions is conducted in order to state their characteristics that lead to choosing the most suitable choices. Secondly, a methodology to collect and centralize network digital evidences in order to come up with the reliable investigation is introduced. Finally, the guidelines to collect and centralize network digital evidences in a successful manner are produced.

KEYWORDS

Forensic guidelines, network forensic guidelines, digital crime investigation, computer forensic, malware, botnets.

1 INTRODUCTION

Network forensic investigators have stated the significance of network digital evidences due to digital crimes science and depicted its ability to come up with rare solutions that limited to network forensic and meanwhile system (computer) forensic cannot. Normally, researchers and experts suggest and propose many different solutions such as Firewall's Logs, IDS/IPD Logs, Switches and Router's Logs and eventually incorporate more advanced systems like Honeywall Architecture to effectively help to investigate network digital evidences. Actually, Honeynet Architecture basically is built to simplify network forensic investigation operations through key features that help to collect and capture network inbound and outbound packets [1][2]. Honeynet Architecture is unique in terms of built-in and well configured tools and utilities which help to achieve the mission. A Honeynet is an architecture which its purpose is basically to build a highly controlled network that control and

capture all inbound and outbound network activities. Usually, within this architecture our Honeynets are placed. A Honeypot is a real system with valid services, open ports, applications and data files. One of the key Honeynet architecture components is the Honeynet gateway which called Honeywall operating system. The Honeywall operating system is a very significant element in the Honeynet architecture since it captures and controls all of the inbound and outbound activities [3].

In reality, traditional information technology environment consists of main critical digital components such as Routers, Firewalls, Intrusion Prevention Systems and operating systems used as servers in order to deliver its mission. Fig. 1 depicts an overview of these common parts of an environment that is available nowadays. Normally, these equipments being configured and assigned an Internet Protocol (IP) which explore and probes them all over the world means they could be accessible from outside to everyone. Actually, the mentioned feature presents risk toward an IT environment since it allows an attacker to bypass and circumvent the built security solutions in case there is a zero-day attack because everything is detectable and known form outside.

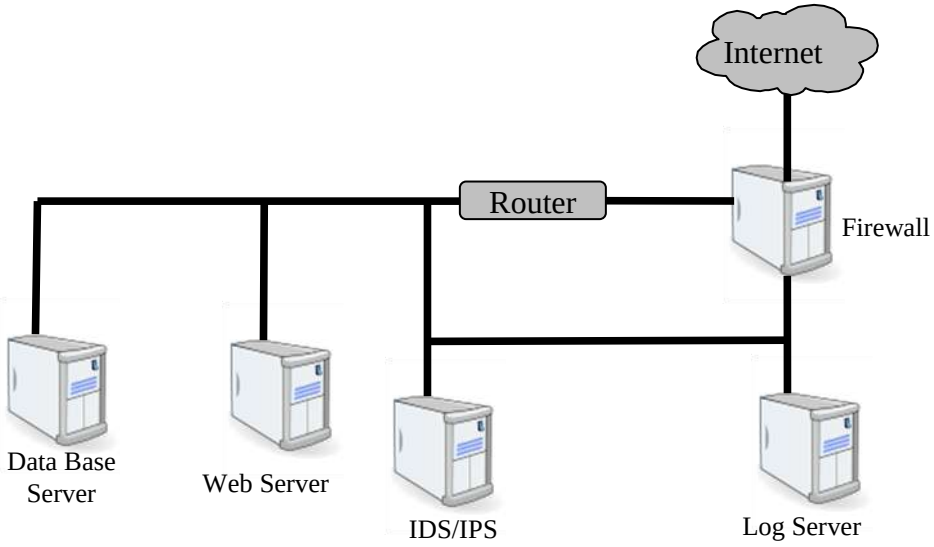

Fig. 1. Traditional IT Infrastructure.

Recently, attackers have grown to be more intelligent against investigations since they keep developing new techniques used to hide or overwrite the digital traces which might lead to grasp them. One of these expected crimes, overwriting all of operating system digital traces, firewall logs files, or intrusion/Prevention Systems (IDS/IPS) logs files and so on [4]. Furthermore, sometimes even worse, they use encrypted channels during conducting attacks which make digital traces analysis impossible without the decryption [5].

In the occurrence of attacks, it is enormously difficult to come up with a detailed analysis of how the attack happened and then depicting what the steps were especially against skilled

attackers who are clever enough at covering their tracks. The operating systems digital traces, Router logs files, Firewall logs files and intrusion detection alerts are unlikely to be sufficient for a serious investigation [4]. Therefore, the efficient solution is in the area of Network Forensics; a dedicated investigation technology that allows for the capture, recording and analysis of network packets and events in order to conduct proper investigation [6]. In general, network forensics defined as is the capturing, recording, and analyzing of network packets and events for the sake of investigative purposes.

From results in this research, concentrating on network forensic is more accurate and much reliable since it allows setting up incorporated hidden nodes or hidden points in network environment that are not detectable by attackers to be used for capturing the desired suspected packets in investigation processes and then these packets should be centralized as well for the sake of simplifying investigations. Honeynet architecture is mainly used here to achieve research aim.

Despite various proposed and developed solutions in field of network forensic digital evidences are shown but they lack handy tested obvious guidelines that demonstrate their usages step by step according to relevant laws or otherwise will not be admissible in the court and also are useless [7].

Zaid Hamzah noted and explained that having the evidence submitted is one thing but its value is another. Lawyers have to persuade the courts that the evidence presented strengthen their case or weaken the case of the other side, and also he linked to relevant legal aspects such as Chain of Custody and Admissibility of Evidences as follow: (1) Chain of Custody: A chain of custody is a sequence of events that shows how evidence was collected, analyzed, and preserved in order to be presented as evidence in court. Any forensics analyst should be careful to do not break the chain of custody; otherwise, the digital evidence may not be admissible in the court. (2) Admissibility of Evidence: Admissibility means whether the evidence could be presented in the court or not. Not all digital evidence gathered is admissible in the court; in addition, some digital evidence may be admissible in one country and not admissible in others. For example, log files are admissible in some countries, but it is considered hearsay in other countries. Therefore, this paper eventually came out mainly to contribute to establish such guidelines that govern a smooth applying [8].

2 HONEYNET ARCHITECTURE

As stated in the previous section, Honeynet Architecture will be mainly used here to collect and centralize network digital evidences [9]. In fact, Honeynet Architecture logically encompasses of three different subsections Gen III, Virtual Honeynets and Distributed Honeynets. In this section, the comparison among three main types of Honeynet is conducted in terms of security, time, and cost of network evidences collection and centralization [10]. The necessity of a comparison is to assists to present key characteristics of Honeynet types which help to generate guidelines. Table 1 summarizes their advantages and disadvantages.

2.1 Gen III Honeynets

Gen III is the third and latest generation of Honeynets which based on the Roo-1.4 Honeywall operating system. Gen III, is designed mainly to be a production solution to be used by individuals and organizations around the world in order to collect and centralize network data that might used as evidences. Many improvements have added to previous version Gen II which is called Eeyore [**5**]. Regarding to the applied security of Gen III, is basically considered enough secure which all integrated systems are real and a gateway that Honeywall OS that is separated from other integrated systems. Honeywall OS installed and configured on bridge mode that hardens it hides its existence. Also since all other systems are separated a part of Honeywall OS, this technique helps to prevent single point of failure to be occurred. It is also flexible enough since allows all systems to be added easily into the laboratory as depicted in Fig 2.

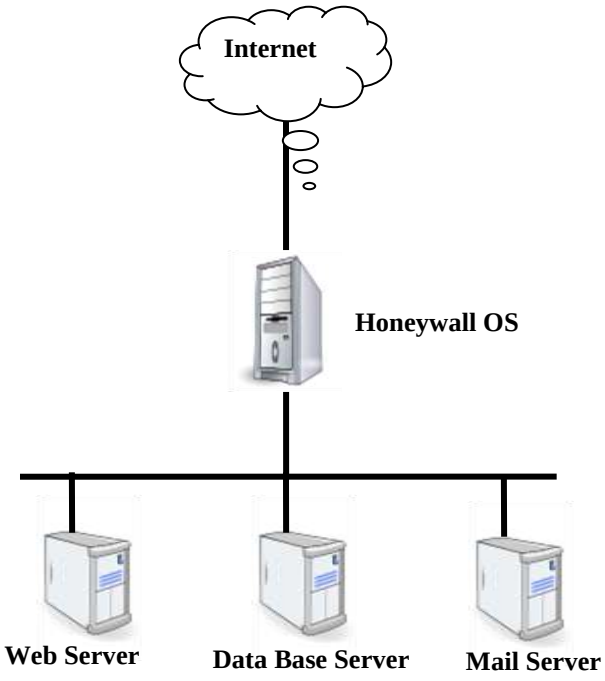

Fig. 2. Gen III Honeynets Architecture.

However, with the regard to Gen III Honeynets' time that needed for collecting and centralizing network digital evidences is often optimal since solely need time of delivering and logging of the wanted chosen inbound and outbound packets. In Contrast, deploying Gen III might be costly since each system must be separated and standalone. Conceptually, Gen III Honeynets costs more than Virtual Honeynets which it uses virtual machine for the deployment.

2.2 Virtual Honeynets

Virtual Honeynet architecture basically uses virtualization technology to combine all the various components of a Honeynets onto a single computer throughout of using virtualization software [10][11]. However, instead of running four Honeynets installed on four

physical computers (a Honeywall and three Honeypots) it easily possible to run such a deployment on a single computer by using virtualization software such as VMWare software.

In particular, virtual honeynet consists of two subsections: Self-Contained Honeynets and Hybrid Honeynets. The deference between them is that Self-Contained combined all components even a gateway (Honeywall OS) beside honeypots within single physical computer , whereas Hybrid Honeynet is that the network gateway (Honeywall OS) only installed within virtual machine, but others servers are real systems. Regarding to virtual honeynet's applied security is considered the least offered feature and very weak because using of virtual machines which inherit many risks. The first risk of Virtual Honeynet is still suffering from single point of failure that in case a system that holds others systems failed, all other system will fail too. The second risk is that limited only to special types of operating systems to be served within it. The third risk which considered extremely dangerous is that can be detected and probed remotely by attackers through check of its fingerprint. This because of assigning special numbers on their MAC addresses. The following MAC addresses are significantly stated for VMWare: **00-50-56 , 00-05-69 , 00-0C-29, 00-1C-14.** In addition, another way used to detect existence of Vmware is using Back door I/O port since VMWare uses the I/O port 0x5658 to communicate with the virtual machine. It is obvious this port is not real. The verification is simple as following:

i. The magic number 0x564D5868 is loaded in the EAX register.
ii. The proper parameter of the command that is to be sent is loaded in EBX register.
iii. The command to be used is loaded in the ECX register. For example, the command 0x0A which prints out the VMWare version.
iv. It is read from VX port. If we have VMXh in the EBX register, this means that we are under Vmware.

However, the cost deploying Self-Contained Virtual Honeynets apparently is counted the least since that needs only virtual machine software and therefore all the servers and Honeywall OS will be setup within single machine by helping of a virtual machine. In contrast, Hybrid Honeynets is more little bit different than Self-Contained Virtual Honeynets. In a Hybrid Honeynets only a gateway Honeywall OS will be installed in a virtual machine but others Honeypots systems are real and separated systems.

Finally, regarding to time used to deliver network evidences collection and centralization mission depend on Virtual Honeynet types, in self-Contained Virtual Honeynet it considered the fastest offered solution because all of the systems installed on a single machine so that the time counted only for sniffing and logging. In opposition, Hybrid honeywall is slower than Self-Contained Virtual Honeynet that needs extra time to contact with the gateway Honeywall OS in order to login the network evidences. In general, delivery time to sniff and log network evidences on Virtual Honeynets is the fastest among other proposed solutions.

Moreover, Self Contained Virtual Honeynets has extra several advantages such as they are portable which means can be moved or changed to anywhere under anytime because all on one system, they play and catch which

means running one system as running all of systems and cheap and take little space as demonstrated in Fig 3. Despite of that, Self-Contained Virtual Honeynets suffers from many shortcomings like still there is single point of failure because in case the system that carries out all goes down the other systems will go down too. Also, they need high quality computer which take responsibilities of all installed systems to improve the performance. Additionally, it limited to special types of Intel architecture that do not give much options to other types such as SPARC OS and so on.

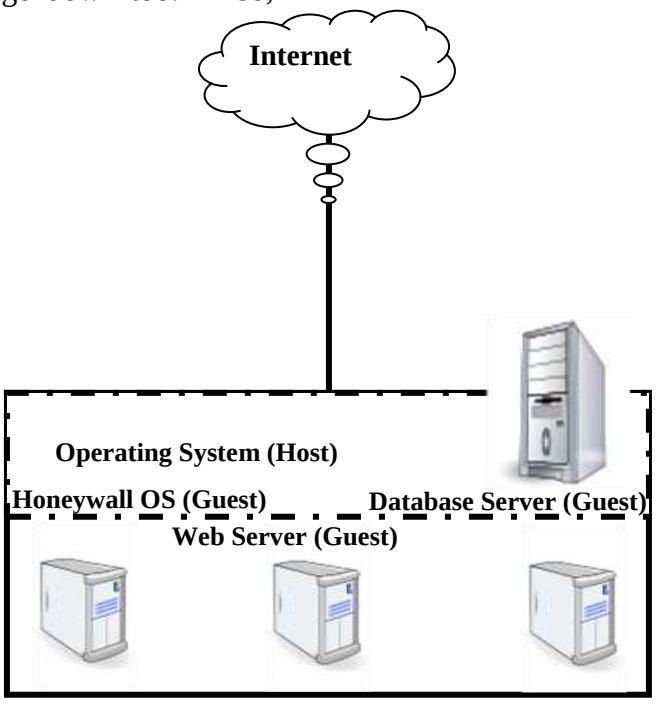

Fig. 3. Self-Contained Virtual Honeynets Architecture.

In comparing to Self-Contained, Hybrid Honeynets is considered more much secure than Self-Contained Honeynets because its gateway Honeywall OS is separated from honeypots' systems. Also, it considered flexible since allows others systems to be integrated with. But is not portable, cost more and take more space for deployments as explained in Fig 4.

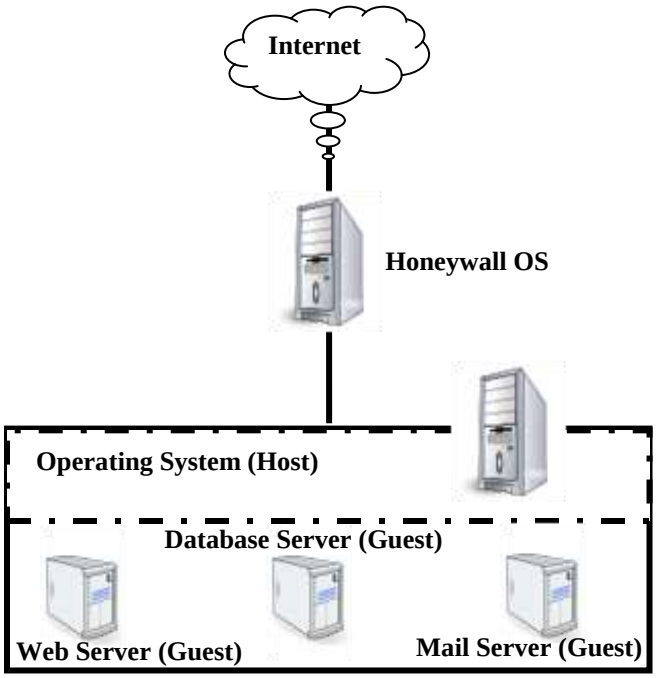

Fig. 4. Hybrid Virtual Honeynets Architecture.

2.3 Distributed Honeynets

The principal feature of Distributed Honeynets is to monitor activities occur across multiple networks concurrently from an analysis center. It basically comes into two kinds; first kind is called *Physically Distributed* Honeynet and the second kind is called *Honeynet Farms*.
Physically Distributed Honeynet is a logical extension of the Gen III Honeynet design. It provides an ability to perform centralized analysis of the collected network digital evidences by the multiple Honeynets across multiple networks. Physically distributed Honeynets independently audit and monitor their local network and send their captured data to a central system. But in contrast, the primary disadvantage that requires hardwares to be presented at the site as depicted in Fig 5.

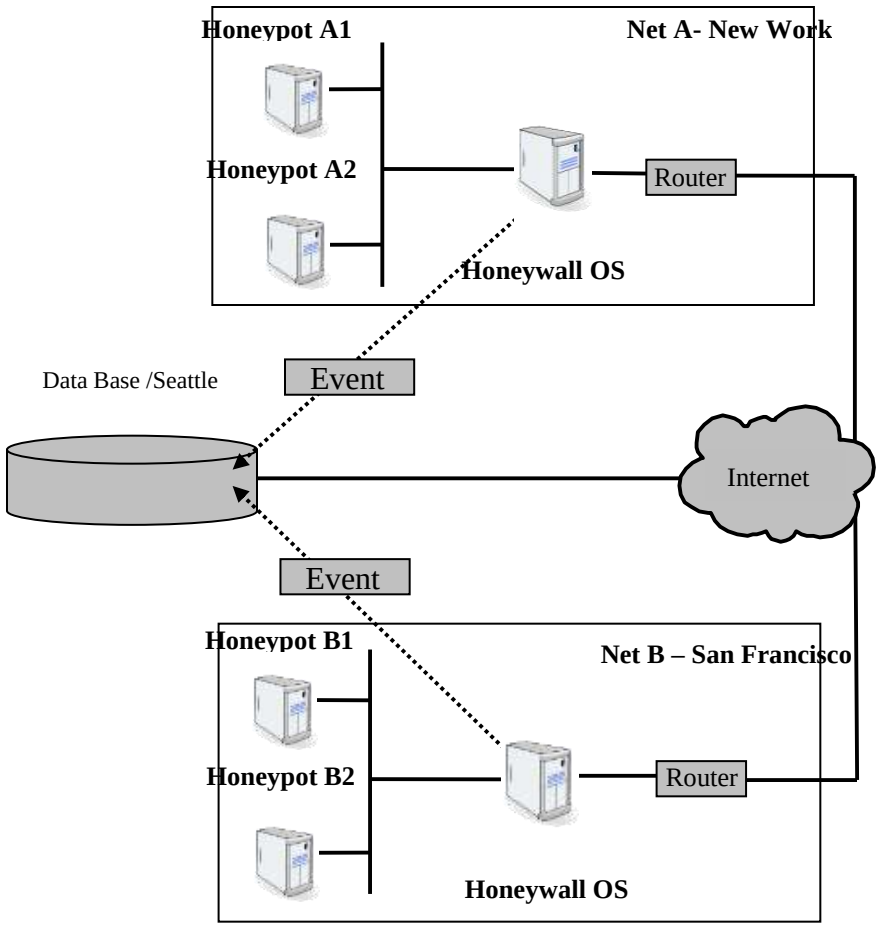

Fig. 5. Physical Honeynets Architecture.

Honeynet Farm is a more radical extension of Gen III design. Honeynet Farms combine Honeynets with Virtual Private Network (VPN) technology to virtually distribute Honeynets to significantly reduce the cost and time of deployment as explained in Fig 6. In fact, the security that relevant to Distributed Honeynets is considered often high. Physically Distributed Honeynets security counted as highest level since all used systems are separated and does not present a gateway (Honeywall OS) to the risk. Using of virtual distributed technology in Honeynets Farms causes positive and negative effects in the same time. It could be positive when it uses VPN to encrypt the traffic and negatively when it causes network latency problem that might lead or guide to existence of Honeywall OS.

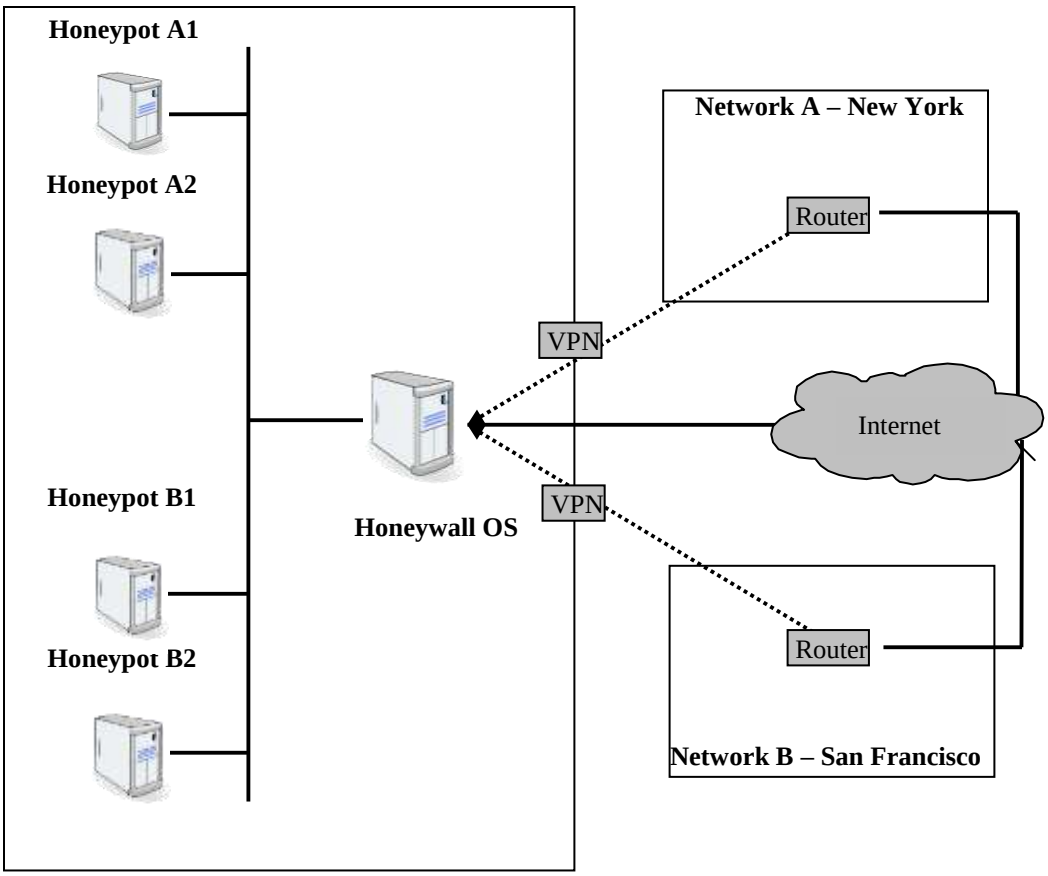

Fig. 6. Farm Honeynets Architecture.

Network latency has a substantial effect on the performance of any request-driven file transfer protocol, since it causes a delay between the time the client issues a request for a file and the time when the data for that file starts to arrive. This is one of the principal disadvantages of purely on-demand transfer strategies, such as downloading individual files. The request latency problem is compounded when the available bandwidth is high, since the penalty associated with each request is the product of the latency and the bandwidth (David Hovemeyer, 2001). In contrast, the provided security is very high and regarding to the cost because it needs separated systems and would be distributed anywhere as required. Here is totally opposite to Virtual Honeynets. Also, time to deliver the mission will take the longest time because in addition to time of logging, it adds time of connecting the central data base and also spend more time to encrypt the traffic. Generally, cost of Honeynets farm is considered less than Physically Distributed Honeynets.

Table 1: A Comparison among Honeynets Subsections.

Honeynet Type	**Security**	**Time**	**Cost**
Gen III	very high	high	medium
Self-Contained Virtual Honeynet	very low	very high	very low
Hybrid Virtual Honeynet	low	high	low
Physically Distributed Honeynet	very high	very low	very high
Farms Distributed Honeynet	medium	very low	medium

3 RESEARCH'S METHODOLOGY

As Fig. 7 demonstrates, our centralizing network digital evidences methodology encompasses two logically dissimilar phases. (1) Digital evidences collection (capturing). (2) Digital evidences centralization. Firstly, the goal of network digital evidences collection phase is simply to capture attackers' activities as many as possible. However, developing a robust scalable infrastructure to achieve this goal is challenging and is a target of numerous researchers. In particular, any designed and developed infrastructure should scalable enough to support wide range of digital evidences collection. In addition, special actions must be implemented to prevent any part of system to behaving malfeasance. However, various different mechanisms are used in order to overcome the mentioned problems and therefore one solution (utility) is a candidate to represent each mechanism. In special, IPTable firewall utility represents firewall mechanism, Snort utility represents an intrusion prevention system [12], Logsys utility represents logging system and lastly Sebek utility represents key logger for an encrypted network packets [5][13][14]. However, these utilities are explained in more details in next sections.

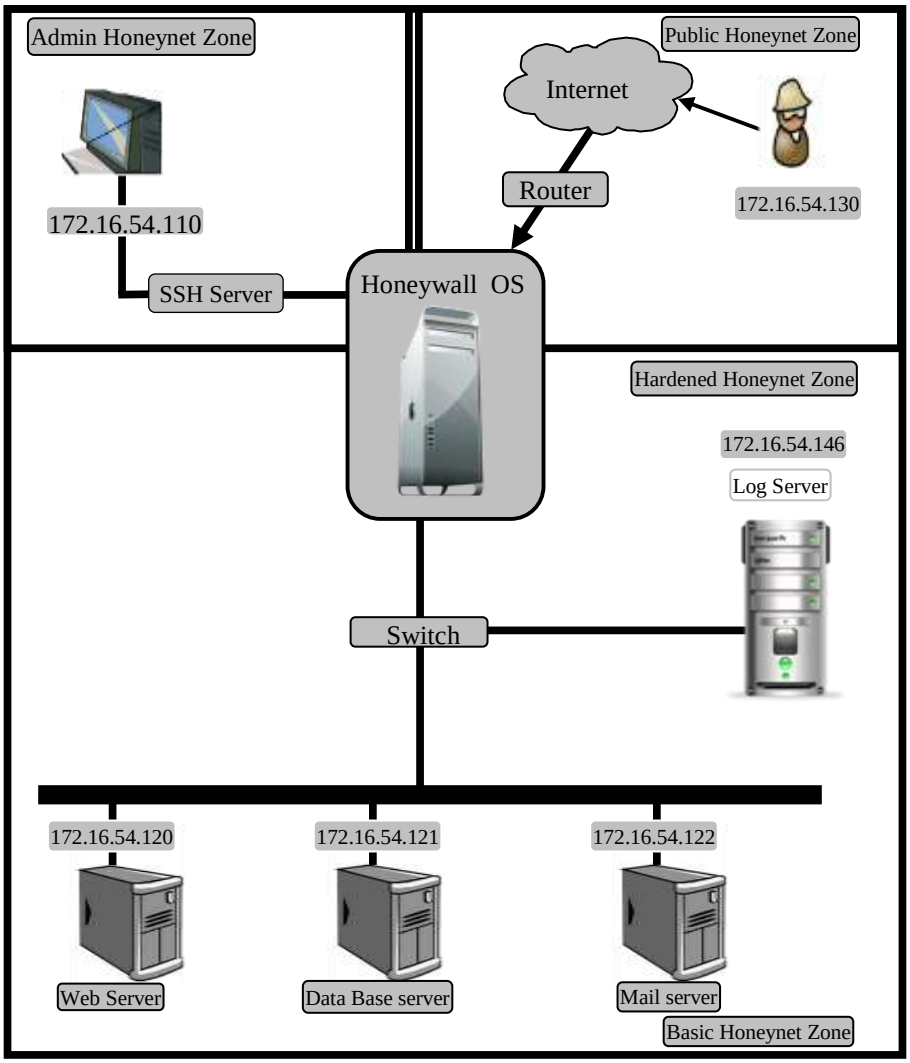

Fig. 7. Centralizing Network Digital Evidences.

An IPTables firewall installed on Honeywall OS which basically to be used for capturing attackers' activities and actions that against a victim since Honeywall is configured as a hidden media in bridge mode [15][16] as depicted in Fig. 8.

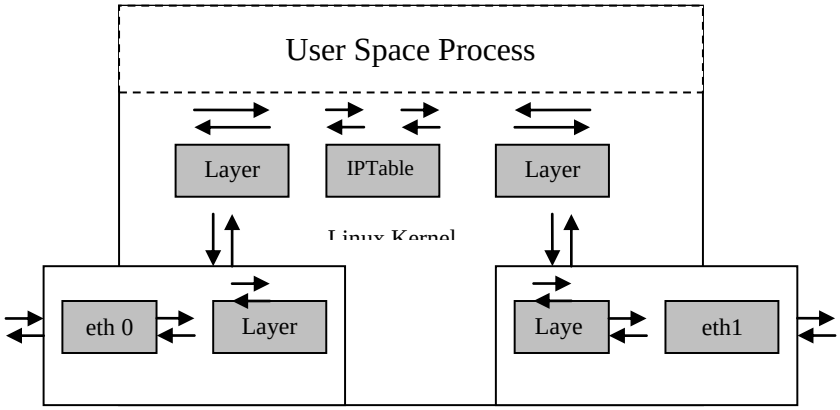

Fig. 8. IPTables and Bridging Mode.

Therefore, IPTables firewall is immune to be detected by attackers [**17**] [**18**][**19**] [**20**]. By default, IPTables log its message to a /var/log/ directory. The next sample depicts an interested potential an attacker's activity he attempted as against a target:

Sep 7 08:56:25 mohd kernel: INBOUND TCP: In=br0 PHYSIN=eth0 OUT= br0

PHYSOUT= eth1 SRC=192.168.0.5 DST=10.0.0.7 LEN=64 TOS=0x00

PREC=0x00 TTL=48 ID=42222 DF PROTO=TCP SPT=1667 DPT=21

WINDOw=65535 RES=0x00 URGP=0

The previous firewall log output explains very significant information about an attacker's activity which to be used by investigators to reveal and analyze attacks and could be analyzed as following:

The Date and time: Sep 7 08:56:25

The Source IP address of an attacker: 192.168.0.5

The destination IP address of an attacker: 10.0.0.7

The protocol being used by an attacker: TCP

The source port of an attacker: 1667

The destination port of an attacker: 21

Snort with *Active Mode* or *Snort_InLine*, an intrusion prevention version of Snort; installed along with an IPTables firewall on Honewall in order to deliver the mission of the intrusion prevention system since highly required and could be used as another layer of protection and also digital evidences collection method as well [21][22]. Fig. 9 shows the logical integration of Snort with IPTables on Honeywal OS.

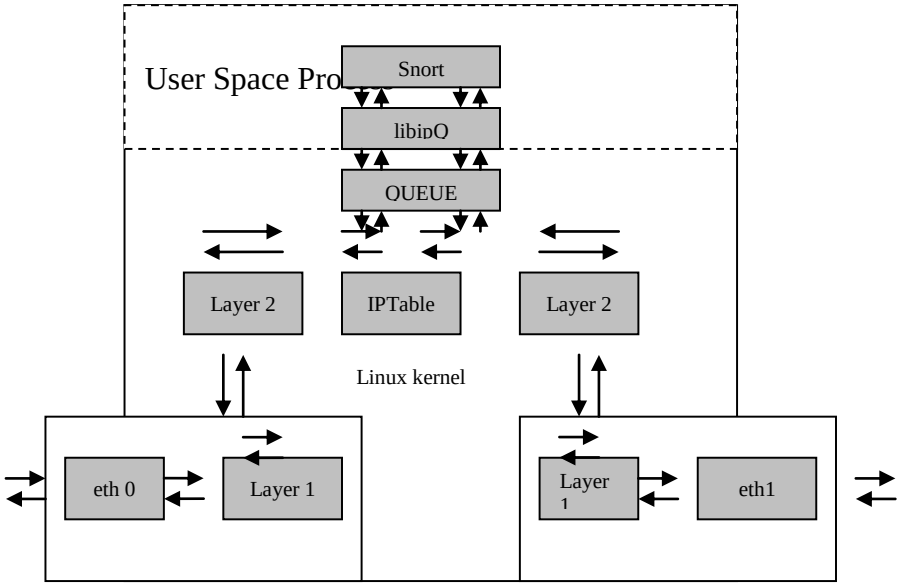

Fig. 9. Snort integrated with Honeywall OS.

Snort primarily uses a rule driven language which combines benefits of signature, protocol, and anomaly based inspection methods. The next sample shows the collected digital evidences using this rule:

alert tcp any any -> any 80 (msg: "Sample alert"; classtype:misc-attack; sid: 2002973; rev:1;)

[**] [1:2002973:1] Sample alert [**]

[Classification: Misc Attack] [Priority: 2]

12/12-15:35:22.130162 test_client:35524 -> test_server:80

TCP TTL:64 TOS:0x0 ID:35734 IpLen:20 DgmLen:52 DF

A* Seq: 0x5F3B46F0 Ack: 0x85067266 Win: 0xB7 TcpLen: 32

TCP Options (3) => NOP NOP TS: 49925498 1529581

Log messages provide worth details about the events of devices and the applications running on these devices as well. Log messages could be used to discover and analyze security incidents, operational problems and policy violations which useful in auditing and forensics situations. The next sample shows the collected digital evidence:

mohd@ubuntu:~$ tail -f /var/log/messages

Mar 27 11:34:35 ubuntu kernel: [165.459827] Bridge firewalling registered 44

Mar 27 11:34:35 ubuntu kernel: [165.563138] Bluetooth: SCO socket layer initialized

Mar 27 11:34:39 ubuntu kernel: [170.004085] eth4: link up, 100Mbps, full-duplex

However, the outputs has generated by typing tail command. The analysis of the first output file analyzed as following:

Time: Mar 27 11:34:35

Host: ubuntu

Facility: kernel

Message: [165.459827] Bridge firewalling registered

In fact, however, the current iteration of Sebek tool is mainly designed not only to record keystrokes, but to record all *sys_read* calls too. For an instance, if a file is has uploaded into Honeypot, Sebek immediately records the file which producing an identical copy. In general, Sebek consists of two components; a client and server as appeared in Fig. 10. The client part captures off Honeypot's data of a Honeypot and exports it to the server through the network. The server collects the data from one of two possible sources: the first is a live packet capture from the network, the second is a packet capture archive stored as a tcpdump formatted file. Once the data is collected then either will be uploaded into a relational database or the keystroke logs are immediately extracted. However, the client part resides entirely on kernel space within the Honeypot and implemented as a Loadable kernel Module (LKM). The client can record all data that a user accessed via the *read()* system call. This data is then exported to the server over the network in a manner that is difficult to detect from the Honeypot running Sebek. The server then received the data from all of the Honeypots sending data.

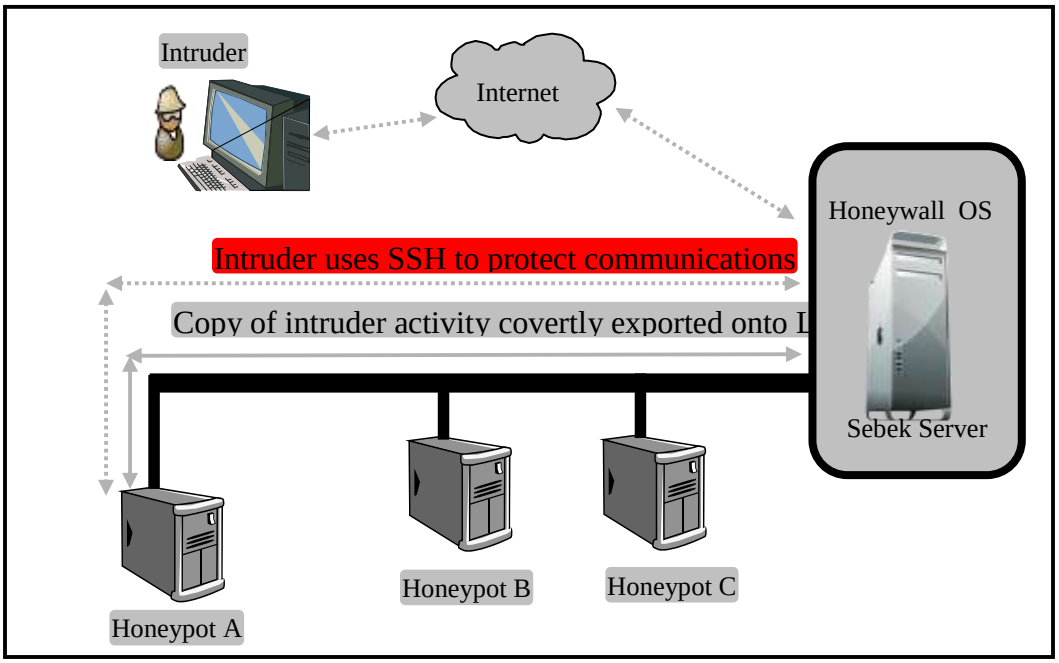

Fig. 10. Sebek Sever/Client Architecture.

Moreover, Sebek generates its own packets and sends them directly to the device driver thus no ability for an attacker to block the packets or sniff them by using a network sniffer since Sebek uses its own implementation of the raw socket interface which programmed to silently ignore Sebek packets. This technique demonstrated clearly in Fig. 11. Sebek packets are defined as those that have both a predetermined destination UDP port and

the proper magic number set in the Sebek header. If these two values match what is expected means this a packet to be ignored. The implementation simply does nothing with Sebek packets; it drops them on the floor and moves on to the next packet in the queue. The end result is that an intruder is unable to capture the Sebek packets.

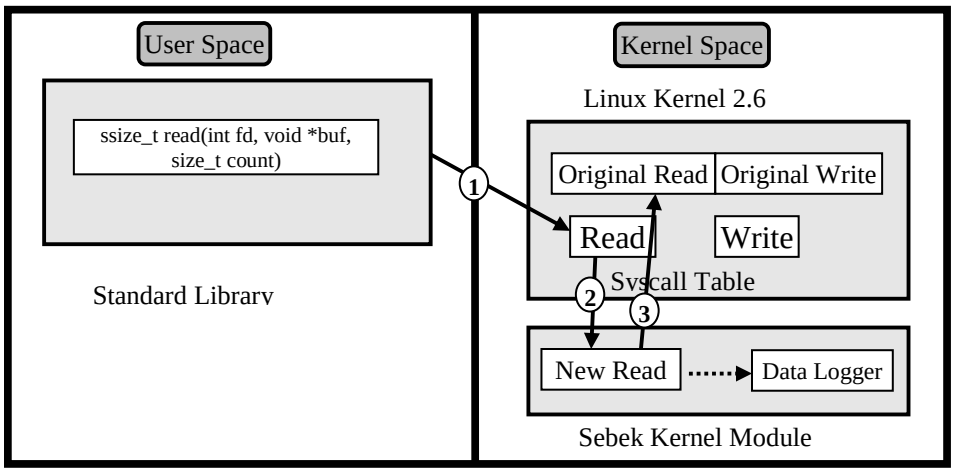

Fig. 11. Sebek Module Conceptual Logic.

Secondly, the main goal of network digital evidences centralization phase is merely to centralize the captured inbound and outbound network packets in arranged ways for simplifying investigation purposes since all evidences will be stored in one place. Nevertheless, developing a robust solution to achieve this goal is challenging and should be scalable enough to support all sorts of collected evidences. Therefore, all the previous evidence collection mechanisms are primarily used here and the next section demonstrates how they handled to achieve this aim.

The sequential logic of experiment's flow initially started by configuring and adapting main utilities that used in this research. The first step is to configure Log Server to accept and log the remote logs transferring that could be allowed by adopting the option to SYSLOGD="-r" [23]. Moreover, this configuration also involved for centralizing IPTables logs, since IPTables logs to system's logs. On the client side (Honeypots), their configuration option should be changed too to transfer their logs to the remote log server which achieved by adding *.* *@172.16.54.146* into /etc/syslog.conf file. Then, Snort utility has been adopted perfectly for capturing and recording unencrypted channels properly. The captured network packets are recoded twice; into database and into flat files. Finally, Sebek server/client architecture has configured as well to deal with encrypted channels. The Internet Protocol (IP=172.16.54.1) has assigned to Sebek Server which will be used to receive captured Sebek Clients packets. Also, UDP port number 1101, accept and log, and magic value options has chosen. After that, Sebek Client has installed into all honeypots and

configured too. Sebek Client has adapted it's variables like UDP Port Number and Internet Protocol according to Sebek Server.

4 RESULTS AND ANALYSIS

In this section, two major contributions are discussed and explained intensively. Firstly, the outlines of obtained results of the experiment which prepared based on the designed methodology in the previous section are discussed. Then, handy guidelines to collect and centralize network digital evidences are produced.

4.1 Discussing obtained results of the experiment:

In fact, a case study has been used here to evaluate the designed methodology and test the proposed solution which mainly based on network digital evidences rather than computer digital evidences. A scenario constructed as following:

The production environment consists physically of four zones which each zone has its own components, tools and requirements. The first zone is a Basic Honeypots Zone acts as a local network that holds company's servers. These servers are Web Server and Secure Shell (IP=172.16.54.120), Data Base server, and Mail Server [18]. Ubuntu Server operating system is basically installed and configured on all of these servers according to their requirements. Then, Sebek client utility also installed on Web Server in order to capture Web Server's activities that sent through a secure channel and then immediately transfer them to Sebek Server. After that, Honeywall OS is installed on a local network and acts as entry point of the lab's network. The Honeywall OS consists of various tools that used to collect and centralize the network's evidences. These tools are SNORT 2.8.4 application that used as main sniffer and also as Intrusion Detection or Prevention Systems (IDS/IPS). SNORT application is installed on Honeywall operating system and then customized in order in to log to a data base and dump network's packets [21]. Lastly, Sebek Server is installed on Honeywall OS and configured to collect and log the encrypted connections to a data base. The second zone is a Hardened Honeypot Zone that configured just to allow connection from specified production environment's servers [17][18][19][20][24]. This zone has log Server (IP=172.16.54.130) that mainly used to record all of production environment's servers activities [23]. Indeed, Log Server accessing restrictions have hardened by using of an IPTables firewall. The third zone that is a Administrator Honeypot Zone that basically is used just by the administrator in order to configure and adapt Honeywall OS. The accessibility to a Honeywall OS only allowed through an encrypted channel by using of Secure Shell Server (IP=172.16.54.110). The fourth zone is a Public Internet Zone which is generally offers the production environment services to be accessible for the world. This zone is often untrusted and has users they might be legal or illegal.

However, an attacker compromised the production environment remotely when he obtained the root password through brute forcing SSH server. He then firstly uploaded two scripts files used for obfuscating crimes traces. These files mainly launched to remove chosen files

and folders such as log's files and folders within web server. After that, he connected through encrypted channels (SSH) to avoid intrusion detection systems [25]. After that, he uploaded another public key file to encrypt the contents of web application and data base and overwrite the original files as well.

Now, analyzing results collected by the proposed solution helps us to discover digital crime's clues. Firstly, the attacker intentionally tried to login SSH server through brute force attack and he succeeds for logging as Fig. 7 depicts that.

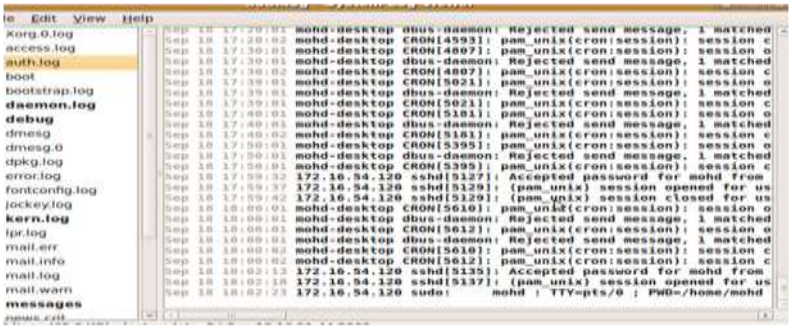

Fig. 12. Log Server received Honeypots' Logs**.**

Then, he uploaded two scripts namely BH-FE.rb [26] and BH-LSC.pl [27] through Secure Shell server (SSH) in order to bypasses the intrusion detection system (IDS). After that, he encrypted the web application files and the data base files too by using an uploaded public key. Finally, he intentionally launched BH-FE.rb [26] and BH-LSC.pl [27] scripts in order to destroy the log files that normally used for forensic investigation. All of an attacker's commands, and encrypted or unencrypted operations captured by Snort and Sebek server and then logged into data base. The following figures show these results:

Fig. 13. Launching Snort Utility.

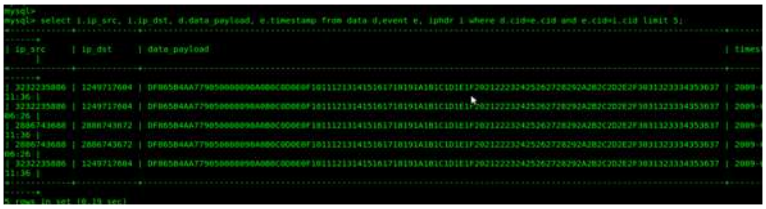

Fig. 14. Network Evidences Logged by Snort.

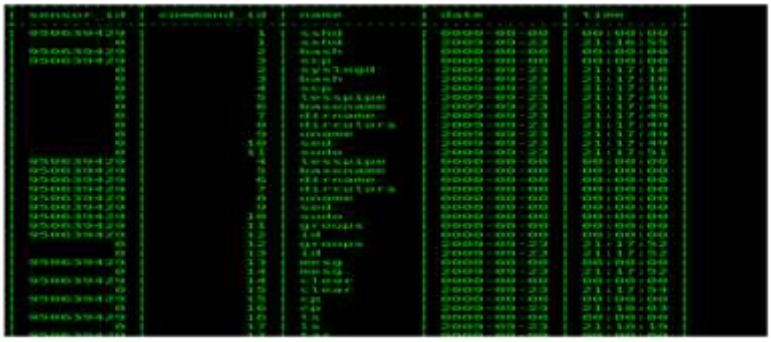

Fig. 15. Network Evidences Logged by Sebek Architecture.

Fig. 16. Network Evidences Logged by Sebek Architecture (Cont.).

4.2 Guidelines to collect and centralize network digital evidences:

Table 2 summarizes the guidelines for collecting and centralizing network digital evidences. The first column of table is special for the mandatory required network components, then followed by explaining their functions in the second column and ended with the best practices which depict the best adopted ways of deployment in the third column.

Table 2: Network Evidences Collection and Centralization Guidelines.

Network Component	Function	Best Practices
Bridge Mode	Hide the existence of the Operating System Network stack since is working under Data Link Layer of OSI that uses MAC addresses for connections purposes and does not assign	- Study the network topology properly and then make a network design accordingly as well. - Do not adopt network gateway that is a Honeywall OS in *routing mode* which allow an attacker to probe it and check and verify its existence. - Only configure the gateway under *bridging mode* on network layer of OSI. Applying that ensure this Operating System that is a Honeywall OS will be detectable by an attacker.

	any Internet Protocol (IP).	- Do not assign any Internet Protocol (IP) on any others network interfaces. Except for Internet Protocol (IP) for management purposes just accessible and allowable for Administrator.
Honeywall Operating System (OS)	Hidden and undetectable form the outside world. It acts as a network gateway and its function is to be the best place to collect and centralize network evidences. It mediates between outside world and servers inside the lab network.	- Make sure network gateway is configured to work under bridging mode and then install and customize the Honeywall as mentioned previously. - Test the system by trying to access the web server to make sure they are accessible from outside the lab and everything should work as wanted and designed. - Make sure that the only the administrator is allowable to access the Honeywall for the sake of configuration and management through encrypted connections channels such as Socket Secure Layer (SSL) which is integrated with Apache web server and also through Secure Shell Server (SSH). - Make sure that Honeywall OS automatic log out has configured as well which helps to harden it. - Sensitive data such as a database, log files, and managing and controlling the system; make sure that each system's users has assigned the least privileges. - Ensure that all unused and unwanted services are removed or stopped and ports are closed too. - Ensure that Honeywall OS is installed on the network lab is updated versions and patched as well. - Customize Honeywall OS according to your requirements if needed. - Use alerting systems such as sending an e_mail or SMS to alert you if an error has occurred.
Network Sniffing (Snort)	Sniff, log, and record network data as evidences in run time of network	- Data Sniffing and recording should be applied and implemented through more than one mechanism which prevents single point of failure. - Data base and log files are the main

	activities.	methods of logging and storing the data. - Logging into binary and log server should be applied too. - Only network data of interest should be logged and recorded. - Tools used for logging processes should be implemented and configured properly. - make sure the tools that sniff and log network data keep original values out of changes. - Implement network data integrity and chain of custody. - Storage medias should be set up and configured based on network data of interest. - If necessary, sniffing and logging tools should be customized in order to apply new feature and requirements. - Only authorized persons or users allowed to access and manipulate these network data. - Sniffing and logging tools should be installed and set up on the gateway which is Honeywall OS. - Snort application with various configured methods and Sebek application are main tools to conduct this mission. - make a test to ensure they are working as designed and wanted.
Sebek Utility	Sniff and log operations and processes that has executed through encrypted channels.	- Sebek client application should be installed and configured properly on all network lab servers. - Sebek server application also should be installed and configured properly too to receive from Sebek clients application. - Ensure Sebek Client/Server should not change the original values and take cares with integrity. - Sniffing and logging processes should be occurred in run time of network activities. - Sebek data should be Logged into both; data base and terminal. - Make a test to ensure is working is designed and required.

Log Server	Record or log processes that has happened or occurred on local or remote server	- Log Server should be configured to receive remote loggings. - All network lab servers should configured and adapted to log remotely to Log Server too. - Accessing to Log Server should be hardened and allowed just for network lab servers by setting up powerful and strong Firewall rules. - All others unwanted services should be removed and stopped and ports closed too. - Log Server physically should be secured and only authorized persons can access it. - Ensure all network lab is up to date. - Make a test to ensure everything as designed.

5 CONCLUSION

In this research, the guidelines for collecting and centralizing network digital evidences which the main objective of the research are eventually stated and produced. Firstly, an intensive investigation and analysis of network forensic are fundamentally conducted in order to overcome the inherited shortcoming of computer forensic (operating system) which its traces could be removed or tampered with by an attacker as a final activity after the attacking has finished. Secondly, the methodology of the proposed solution that uses Honeynet Architecture to collect and centralize network digital evidences is designed, then results' outline of the experiment is demonstrated in granular details and subsequently the comparison among its subsections is achieved in order to address their main key features that lead to come up with suitable guidelines accordingly. Finally, handy tested guidelines to collect and centralize network digital evidences are produced.

6 ACKNOWLEDGMENTS

This work of research has been done in Universiti Teknologi Malaysia (UTM) under support of Ministry of Higher Education, vote number 78567.

7 REFERENCES

1. Talabis, R. Honeynets: A Honeynet Definition. The Philippine Honeynet Project. 1-4.
2. Justin Mitchell (2006). Fundamental Honeypotting. SANS Institute. 1-8.
3. Honeynet group. (2004). Know your Enemy (2nd ed.). Addison-Wesley.
4. Almulhem, A., and Issa Traore I. Experience with Engineering a Network.

5. Honeynet Project (2003). A kernel based data capture tool. Honeynet Project. 1-21.
6. Ranum, M., et al. (1997): Implementing a generalized tool for network monitoring. Proceedings of the Eleventh Systems Administration Conference.
7. Lin, A.C., et al (2005). Establishment of the standard operating procedure (SOP) for gathering digital evidence. Digital Object Identifier.7-9 Nov. 56 - 65.
8. Zaid Hamza. (2005). E-security Law & strategy. Malaysia: LexisNexis – Malaysian Law Journal.
9. Daniel Hanson, (2004). Multiple Security Roles With Unix/Linux.
10. Ponweera, R.J.C.; Koggalage, R.; Wickramage, N. (2007). Evaluation and demonstration of the usage of a virtual honeynet for monitoring and recording online attacks. Industrial and Information Systems. 9-11 Aug. 2007. 21-26.
11. Provos, N. and Holz, T. (2007). Virtual Honeypots. USA.: Addison-Wesley.
12. Ramirez, G., Caswell, B. and Rathuas, N. (2005). Nessus, Snort and Ethereal. Rockland.: Syngress Publishing, Inc.
13. Honeynet group. (2001). Know your Enemy (1st ed.). Addison-Wesley.
14. Peisert, S. (2005). Forensics for System Administrators. 1-7.
15. Hunt, C. (2002). TCP/IP Network Administration. (3rd ed.). USA.: O'Reilly.Scott, C., Wolfe, P. and Hayes, B. (2004). Snort for dummies. Hoboken.: Wiley Publishing, Inc.
16. Schiller, A. C., Binkley, J., Harley, D., Evron, G., Bradley, T., Willems, C. and Schroder, C. (2007). Linux Networking Handbook. (1st ed.). Sebastopol.: O'Reilly.
17. Alan Jones (2004). Netfilter and IPTables. SANS Institute. 3-15.
18. Cronje, G. (2003). Choosing The Best Firewall. SANS Institute.1-9.
19. Kevin Law (2005). An Introduction of Firewall Architectures and Functions. Winter. 1-6.
20. Suehring, S. and Ziegler, R. (2005). Linux Firewalls. (3rd ed.). USA.: Sams Publishing.
21. SANS Institute (2001) Intrusion Detection Systems. SANS Institute. 1-12.
22. Goeldenitz, T. (2002). IDS-Today and Tomorrow. SANS Institute. 1-9.
23. BalaBit (2007). Distributed syslog architectures with syslog-ng Premium Edition. BalaBit IT Security. 1-12.
24. Intoto (2002). Firewall White Paper. Firewall. 1-3.
25. Heather, M. .L (2002). SSH and Intrusion Detection. SANS Institute. 2-6.
26. Honeynet group. (2004). Know your Enemy (2nd ed.). Addison-Wesley.
27. BH-FE.rb script available at *http://aalagha.com/blog/2008/09/09/bh-final-eraser-version-05.*
28. BH-LSC.pl script available at *http://aalagha.com/blog/2008/04/20/bhlsc-linux-server cleaner*.