



# A Hybrid Honeypot Scheme for Distributed Denial of Service Attack

**Hazem Sallowm, Mohammed Assora, Mohammed Alchaita, Mohamad Aljnidi**

Higher Institute for Applied Sciences and Technology, Damascus, Syria

**Email address:**

hazemsallowm@yahoo.com (H. Sallowm), m.assora@yahoo.co.uk (M. Assora), malchaita@yahoo.com (M. Alchaita), mjnidi@gmail.com (M. Aljnidi)

**To cite this article:**

Hazem Sallowm, Mohammed Assora, Mohammed Alchaita, Mohamad Aljnidi. A Hybrid Honeypot Scheme for Distributed Denial of Service Attack. *American Journal of Electrical and Computer Engineering*. Vol. 1, No. 1, 2017, pp. 33-39. doi: 10.11648/j.ajece.20170101.15

**Received:** March 12, 2017; **Accepted:** April 6, 2017; **Published:** May 24, 2017

---

**Abstract:** The main challenge in network security is keeping up with all threat types that arise every day. Traditional security mechanisms such as firewalls and Intrusion Detection System (IDS) do not provide detection for new attacks or helping in learning new attackers' techniques. This paper presents a hybrid honeypot scheme that combines low and high interaction honeypots to mitigate the shortcomings of both types. The low interaction honeypots are used to emulate operating systems and services, and for any outbound connection, they act as a proxy to forward the packets to real systems in high interaction honeypot. The scheme is tested by applying Distributed Denial of Service attack (DDOS) against the system, and a significant enhancement to the system security is achieved. The results show that the performance of the IDS has been improved comparing with traditional IDS. Furthermore, the false positive rate is reduced, and the true positive rate is enhanced.

**Keywords:** Intrusion Detection System, Hybrid Honeypot, Distributed Denial of Service Attack

---

## 1. Introduction

Network Security becomes the keystone in modern societies because of new threats that arise every day. These threats demand advanced security solutions that are no more available by traditional tools. Traditional solutions, such as Firewalls and IDS (Intrusion Detection System), are unable to keep up with the evolution of the attackers and their techniques, as they suffer from detecting and stopping unknown attacks and new attackers' methods. To achieve a good level of security, the security tools should not only be interested in the defense mechanisms, but also must rely on deceptive attackers and must have the initiative to disclose the attack when it occurs.

A honeypot is a system that is built and set up in order to be hacked, therefore it can be deployed to consume the attackers' resources and to waste their time on honeypots instead of attacking production systems. These purposes help to detect new attacks and learn more about attackers' techniques. This paper uses a hybrid honeypot to enhance the efficiency of IDS detection, and to improve the overall system security.

This paper is organized as follows; in section II, we review honeypots. Section III includes related works in honeypots and IDS domain. After that in section IV we present a

proposed hybrid honeypots framework. Section V discusses the case study and shows results, and finally section IV ends the paper with a conclusion and future works.

## 2. Honeypots

A honeypot [1] is a trap that seems to contain valuable information or resources and appears to be a part of the network, but it is actually isolated and monitored. It works by fooling attackers into believing that it is a legitimate system. All of this helps us to collect valuable data about attackers' tools and methodologies. The main functions of honeypots are [2, 3]:

- Divert attackers' attention from the real network in order to protect the production systems.
- Collect information about the attackers such as their tools, techniques, and methods.
- Capture new attacks (viruses, worms, etc.) for future studies.
- Reduce false positive and false negative in IDS, because every time a honeypot generates an alert, it most likely is a real attack and not a false alarm.
- Identify new vulnerabilities and risks of various operating systems and programs, which are not

thoroughly identified now.

- f) Monitor only the traffic that comes directly to them, thus it collects a small amount of data that contains highly valuable information.

Level of interaction means how much the attackers are able to interact with the system. More level of interaction brings more information about attackers and more risk in the network security. There are two categories of level of interaction in honeypots [2, 4, 5]:

- a) Low interaction honeypot emulates services, network stacks, or other aspects of a real machine. This type allows an attacker a limited interaction with the target system, and allows us to learn mainly quantitative information about attacks. Here there is no real operating system to be involved. It is used to detect hackers and deceive them by emulating the operating system services and port services. Some well-known examples of low interaction honeypots are Honeyd, Specter and Dionaea.
- b) High interaction honeypot involves real operating systems and applications. The goal of this type is to give the attacker full access to a monitored real system where nothing is emulated or restricted. Thus, significant information can be gained and collected about the attacker's tools, tactics, and motives to analysis for future studies. The risk associated with these honeypots is higher because the attacker can have full control of them. Hence, it can be used as launch pads for attacks, or to attack another system such as for launching a DDOS attack. Common examples of this type are honeynets, HIIHAT, and Sebek.

HONEYD [6]: is an open source low interaction honeypot solution and designed for UNIX systems. Honeyd simulates the TCP/IP stack of operating systems to create virtual honeypots, and supports TCP, UDP and ICMP. In addition, Honeyd can also support simulation of arbitrary routing topologies included latency, packet loss, and bandwidth characteristics. According to the evaluation of CERT Polska [7], Honeyd appears to be useful in everyday CERT work, see Figure 1.

### 3. Related Works

Several solutions have been already proposed to deal with many types of attacks. Weiler [8] proposes an implementation of a cluster of physical honeypot servers similar to the one used by the production environment. This gives the attacker a real world feeling to lure him to believe that he successfully compromises a real system. This solution wastes the environment's resources, because every honeypot needs a separate physical server. Vidwarshi et al [9] introduce two high interaction honeypots with a separate link with a router. The researchers propose a mechanism to transfer outbound traffic from one honeypot to the other one. Das [10] introduces a concept of "Active Servers" (AS). Every production server is hidden behind an AS that acts like a honeypot thus protecting the real server from being attacked. This solution depends on physical honeypots so it wastes resources and requires hard

maintenance. In addition, it slows down processes that relate to real users, because every request, if it is real or suspicious, needs to pass through the AS. Mohssen [11] proposes a new technique called "Double Honeynet" with two high interaction honeypots, hence can collect sufficient amount of worm instances. This goal is achieved by forming a loop which allows any outbound connection from one honeynet to be delivered to the other honeynet. Deshpande [12] creates a network of virtualized honeypots, called Honeymesh, within the existing infrastructure with minimal cost and minimal maintenance overheads. There is a router between Honeymesh and the Demilitarized zone (DMZ) to isolate any attack traffic. However, the proposed solution does not have any mechanism to capture and control traffics that pass-through honeypots. In addition, it needs a mechanism to protect routers from being flooded with malicious requests.

DETECTION SCOPE	ACCURACY OF EMULATION	QUALITY OF COLLECTED DATA	SCALABILITY AND PERFORMANCE	RELIABILITY	EXTENSIBILITY	EASE OF USE AND SETTING UP	EMBEDDABILITY	SUPPORT	COST	USEFULNESS FOR CERT
MULTI	★ ★	★	★ ★ ★ ★	★ ★ ★ ★	★ ★ ★ ★	★ ★ ★ ★	★ ★	★	★	★

Figure 1. Evaluation of Honeyd by CERT [7].

## 4. Proposed Hybrid Honeypot Framework

The need for this proposed technology is to provide an integrated framework that combines both low interaction honeypot and high interaction honeypot to take advantages of their positivities and to minimize their negativities. This framework represents a typical solution to some of the traditional methods of network security. This proposal offers a network of adaptable honeypots based on IDS that simulates the actual production network with their topologies, systems and services. Figure 2 shows the proposed hybrid honeypot framework that consists of four sections: production systems, Honeyd, honeynets, and honeywall. In the first section, there is more than one production system in different operating systems (Linux, Windows). This section is connected directly to LAN switch together with Honeyd server (section 2). This solution requires the use of static IP addresses (for honeynets) and the availability of a relatively large number of free IP addresses (for Honeyd). More free IP addresses correspond to higher chances for detecting intruders, as more fake systems would be mined in the network. Honeyd assumes the identity of any IP address that does not have a valid system, so when an attacker attempts to connect to a system that does not exist, Honeyd receives the connection attempt, assumes the identity of the nonexistent system and then replies to the attacker. As shown in Figure 2, there is no router between production computers and honeypot network. To forward the traffic of all of these systems to the Honeyd, we rely on the concept of ARP spoofing by using Farpd utility that runs on Honeyd system. Farpd watches all the ARP traffic on the LAN, so it knows what production systems on the local network are active by

consulting the Honeyd ARP table. When there is an attempt to connect a nonexistent system on the local network that its IP

has no entry in Honeyd's ARP table, Farpd will then attempt to send ARP request packet (unicast packet) to the destination IP.

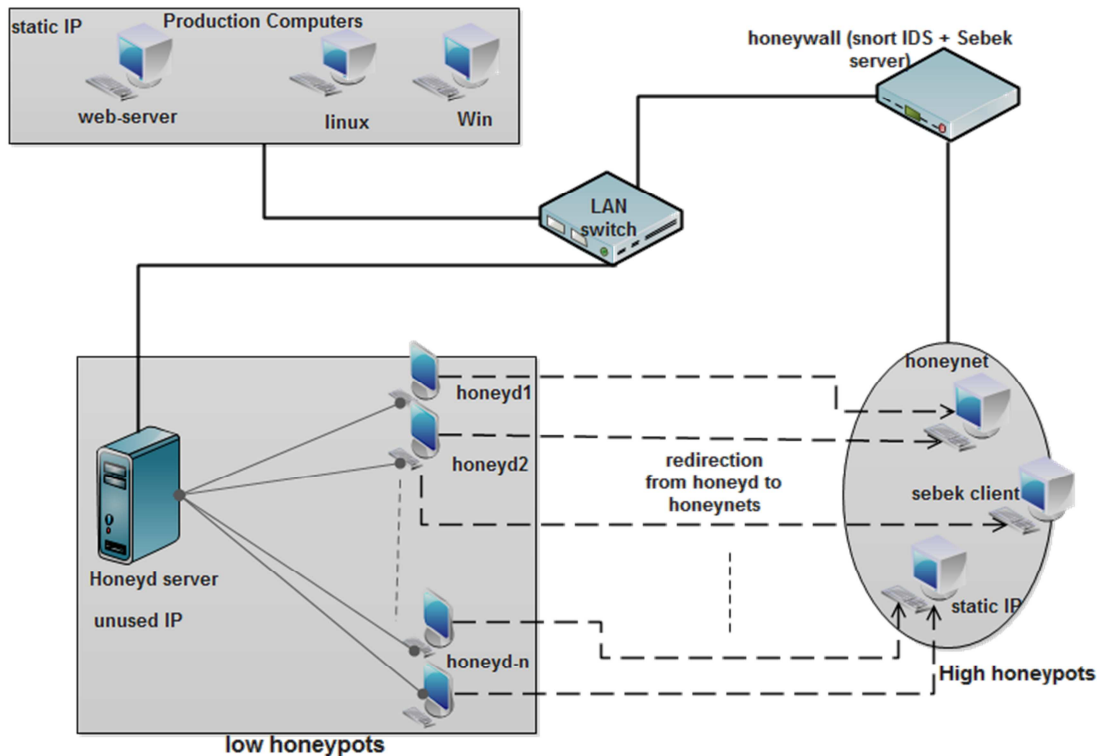


Figure 2. The proposed hybrid honeypot framework.

If Farpd gets an ARP reply from the destination, it will assume the system is present and will add that entry to the Honeyd's ARP table and ignore the connection attempt, because it is most likely a legitimate traffic. However, if Farpd does not get a response from the intended IP, it will assume the system does not exist and this action is unauthorized (malicious action). Hence, it executes its ARP spoofing attack, replies to the ARP request from the attacker. The attacker will interact with Honeyd server without being aware that he fell into the trap. In other words, the attacker believes this response and accomplishes his attack against IP victims, using the MAC address of the Honeyd. The Honeyd receives the packet at the link layer, assumes the identity of unused IP address and takes over the connection. Then, Honeyd identifies what port the attacker is targeting and reacts based on configuration of Honeyd like open port and emulated service. For example, if an attacker connects to TCP port 80, the Honeyd can initiate a Web server emulator and interact with the attacker. As shown in Figure 2, these Honeyds receive their directed traffic and divert them to high interaction honeypots where attackers engage with real services.

This feature of redirection from Honeyds to Honeynet is the heart of our proposal because it provides a better level of security against most types of threats that depend on flooding or spreading techniques like (DDOS attack, and worms) and unknown attacks like zero-day attacks. The third section is a network of physical honeypot, which is separated from the production systems by gateway called honeywall. These honeypots comprise real systems and services (Win-

Linux ...). They use static IP addresses and, in our framework, are virtual machines on one physical machine by using tools (virtualbox or VMware) to decrease the cost and to restore a compromised virtual machine easily and quickly. In this section, to capture any action of the attacker with honeypot, such as keystrokes, system logs, etc., we use Sebek tool [13] that is based on client-server architecture as shown in Figure 3. In every honeypot in the Honeynet, we install Sebek client tool that copies all of the attacker's activities, and then sends them to the remote server in honeywall. The fourth section in our framework is a honeywall [14, 15]. It is a bridged gateway to our system, and the point of entry and exit for all network traffic for this framework. The honeywall has three network interfaces: first interface is connected with external network or production system; the second interface is connected with honeynets and the third one is intended for management or remote logging.

Since the honeywall works in the data link layer, the first and second interfaces have no IP addresses. Therefore, this bridged feature provides no time-to-live (TTL) decrement and packet routing that make it hard to be detected. Honeywall with some tools can provide three main tasks as follows: (1) data capture to confirm that all data traffic from and to the system are detected, recorded, and stored, (2) data control to prevent the attacker when he compromises a honeypot to attack the production system and (3) data analysis to analyze and utilize the stored data.

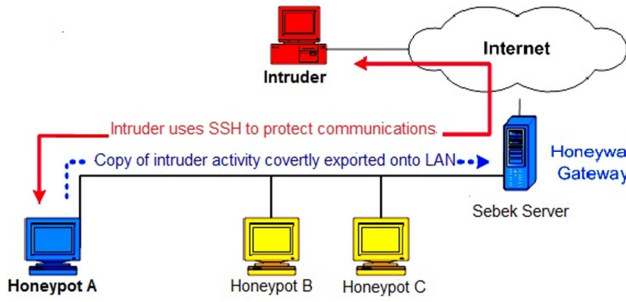


Figure 3. Sebek client-server architecture [13].

The main contributions of our proposed hybrid honeypot framework approach are:

- Large number of Honeyds can be simulated on one physical machine. These Honeyds need extra modification just in the event of change in the actual environment. As a result, the system requires minimum human administrative intervention.
- The possibility of adding a simulated system or services is easy and doesn't consume any resources since there are free IP addresses.
- The presence of high interaction honeypots behind the Honeyds receiving the redirection traffic from them, makes Honeyds from the attacker's viewpoint look like an actual physical system.
- The loop between Honeyds and honeynets with the presence of the honeywall prevents the attacker from compromising any production system.
- The ability of integrating and deploying the system by using virtual environments, allows backups and re-activate them easy.
- The possibility of detecting internal threats, since the complete structure is in the same network.

## 5. Case Study and Results

We created a small lab that is set up close to reality to test our proposed framework, see table 1. We set up two machines for the production system section with Snort IDS and tools for test. One machine is used for Honeyd server, and we create many honeypots with fake services inside it. Two machines with real systems are set up for honeynets network, and one machine is used to create the honeywall. To test the functionality of our solution, we launch some types of DDOS attack from two machines with two different operating systems. The full experiment is divided into four stages as follows:

### 5.1. Stage 1- Testing the IDS Only

At this stage, we create a web server with Snort IDS to examine the IDS performance. It is very important to undertake this stage before applying any kind of attacks to ensure the effectiveness of IDS and its suitability to the experiment circumstances. After performing this stage, the results show that the IDS is able to capture the network traffic over the web server at a very good rate. Where it is able to

analyze more than 99.5 % of the captured data as shown in Figure 4, this Snort IDS is very suitable for our project.

```

=====
Packet I/O Totals:
Received:      21278
Analyzed:      21179 ( 99.535%)
Dropped:       0 ( 0.000%)
Filtered:      0 ( 0.000%)
Outstanding:   99 ( 0.465%)
=====

```

Figure 4. IDS status at the first stage.

Table 1. Proposal prototype summary.

Machine Name	Operating System	Features and Tools
Victim target	Linux Ubuntu	Apache web server + Snort IDS + darkstat tool
Production systems	Linux Win-XP	Two machines + normal services
Honeynets	Linux Win-XP	Two machines + Sebek client
Honeyd server	Linux Ubuntu	From one to ten Honeyds
honeywall	Linux CentOS	Sebek server + P0f+ snort-inline (IPS) tools
Remote management	Windows XP	Walleye tool
Attacker	Backtrack5+win 7	Two machines + Attacking tools

### 5.2. Stage 2- Applying Attacks on the Web Server

After confirming the suitability of IDS performance for our experiment, we apply the attack on the web server, before implementing the proposed project. We create two machines with two operating systems (Linux and Windows) to initiate a DDOS attack by attacking tools like Hping3 and SlowLoris. These DDOS tools depend on flooding the victim by sending a large amount of request packets that makes it unable to cope with. The applied traffic from the attacker's machines into the web server is more than 25 times of normal traffic for a short period. The results of this stage show that the IDS receives the packets but it can analyze just 2% of the whole packets and the rest of them in outstanding state as shown in Figure 5. In other words, the web server is over flooded with these requests and the IDS is not able to handle this amount of traffic.

```

=====
Packet I/O Totals:
Received:      3559267
Analyzed:      65643 ( 1.844%)
Dropped:      3493609 ( 49.535%)
Filtered:      0 ( 0.000%)
Outstanding:   3493624 ( 98.156%)
Injected:      0
=====

```

Figure 5. IDS status at the second stage.

### 5.3. Stage 3- Implementing Honeyd with IDS

Here, we create a Honeyd server that encloses many virtual honeypots with many emulated systems and services; one of them is an emulated web server, which listens to TCP port 80. As we have noted above, as more free IP addresses correspond to higher chances for detecting intruders, as more fake systems are mined in the network. We implement a different number of fake systems in Honeyd server to determine the probability of



attacking fake systems, see Figure 6. In Figure. 6, the curve rises exponentially as a function of the number of free IP addresses. If the number of fake systems is equal to production systems, the probability of attacking fake systems is equal to attacking production systems. When the number of fake systems increases, a greater chance is achieved to attract and deceive attackers. At this stage, we apply the same attacks that were used at the previous stage on our lab network. The results illustrate that the Honeyd server attracts the attacker's machines and responds to their requests. By interacting with an emulated web server, the traffic against the real web server is decreased. Figure 7 presents a list of top connections by destination IP addresses on Honeyd server. The IP 10.0.0.10 is assigned to honeypot that simulates a web service, and other IP addresses in this chart have been assigned to other services. The figure illustrates that the attack is focused on emulated web service on IP 10.0.0.10. Figure 8 shows the list of IP addresses that connect to Honeyd, which presents attacker's addresses. The attacker machine which is implemented in our project has the IP address: 10.0.0.11, and we execute DDOS attacks to compromise the web server directly. The other IP addresses have been generated randomly by the DDOS attacking tool. Figure 9 illustrates in details a network traffic with Honeyd as a statistical result, and shows the amount of

exchange information. It shows clearly that the most of connections have been in TCP protocol on port 80 as shown in previous figures. At the end of this stage, we found that the performance of IDS improved by 15 percent compared to the previous stage, see Figure 10.

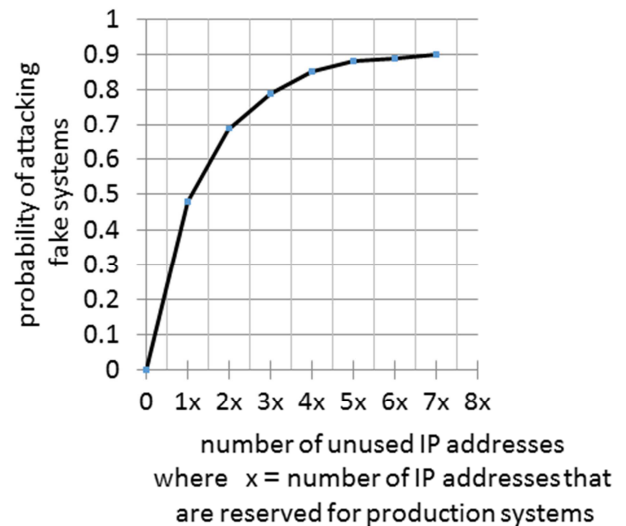


Figure 6. Probability of attacking fake systems.

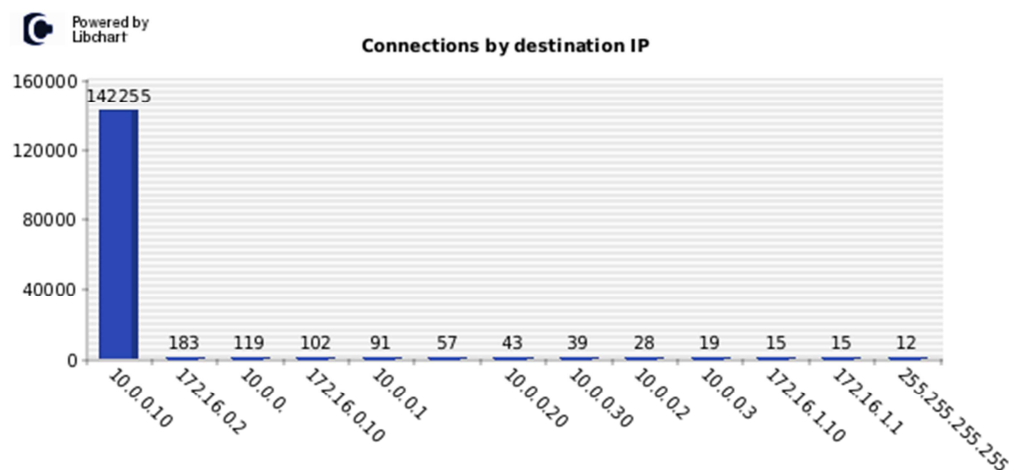


Figure 7. Connections by destination IP.

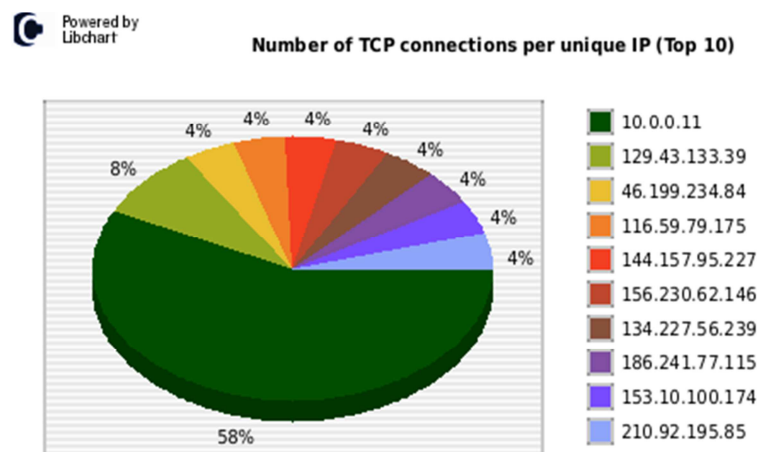


Figure 8. List of IP addresses that connect the honeypots.

darkstat 3.0.715

graphs

hosts

homepage

10.0.0.10

-----

Hostname: (none)

MAC Address: 08:00:27:66:35:aa

Last seen: 2016-08-23 08:30:25 UTC+0000 (4 mins, 16 secs ago)

In: 718,309,492

Out: 8,819

Total: 718,318,311

TCP ports

(1-1 of 1)

Port	Service	In	Out	Total	SYNs
80	http	718,309,240	0	718,309,240	17,957,731

UDP ports

(1-1 of 1)

Port	Service	In	Out	Total
5353	mdns	0	8,407	8,407

IP protocols

(1-4 of 4)

#	Protocol	In	Out	Total
6	tcp	718,309,240	0	718,309,240
17	udp	0	8,407	8,407
1	icmp	252	252	504
2	igmp	0	160	160

Figure 9. Statistical result of connections with honeypot system.

=====	
Packet I/O Totals:	
Received:	232390
Analyzed:	34396 ( 14.801%)
Dropped:	197935 ( 45.997%)
Filtered:	0 ( 0.000%)
Outstanding:	197994 ( 85.199%)
Injected:	0
=====	

Figure 10. IDS status at the third stage.

#### 5.4. Stage 4: Implementing the Proposed Hybrid Honeynets Framework

At this stage, we add the honeynets, which are virtual machines with real operating systems and services. The attack is carried out in the same mechanism as previous stages. In our framework, the Honeyd acts as a lightweight proxy or a front end to the honeynets. In other words, Honeyd forwards any outbound connections from it to the Honeynet systems. When the attacker sends a TCP/SYN packet to Honeyd on a port that is opened and configured to listen in Honeyd, the concerned Honeyd's honeypot responds and sends a SYN/ACK packet and waits to receive the next packet, as shown in Figure 11. If the attacker does not send another packet, Honeyd will drop the connection and assume it as a port scan. However, if the attacker sends an ACK packet, this means that the attacker will establish a full connection with the Honeyd. Later on, the attacker will use this connection to start an outbound connection from the Honeyd. In other words, the attacker will compromise the honeypot and use it to attack other computers on the local network. In order to prevent this outbound connection from compromising production systems and to

collect information about the attacker, the Honeyd now acts as a proxy and forwards packets to Honeynet to establish a full interaction with the attacker. The results of this stage show an increase in the percentage of data analysis in IDS compared to the third stage, the percentage reaches 40%, see Figure 12. Finally, we can claim that the proposed architecture is robust because it does not depend on a single component and this framework can be expanded by adding Honeyd's honeypots and honeynets to get a greater level of security. Another advantage of the proposed system is that false positive rate is reduced. The false positive rate (FPR) and the true positive rate (TPR) are obtained as follows:

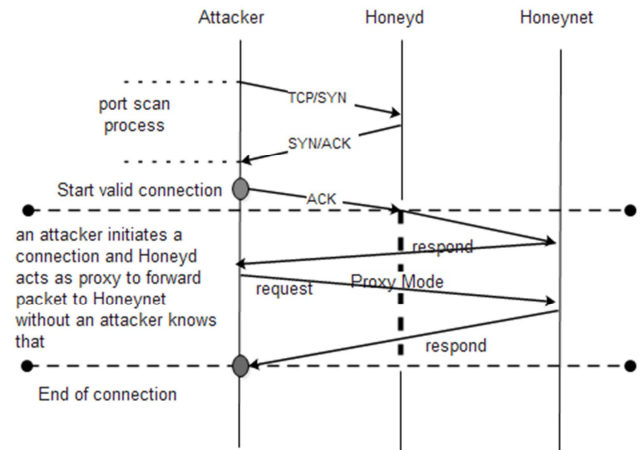


Figure 11. Mechanism of Honeyd's proxy mode.

=====	
Packet I/O Totals:	
Received:	271245
Analyzed:	108252 ( 39.909%)
Dropped:	162993 ( 37.535%)
Filtered:	0 ( 0.000%)
Outstanding:	162993 ( 60.091%)
Injected:	0
=====	

Figure 12. IDS status at the fourth stage.

$$FPR = FP / (FP + TN)$$

and

$$TPR = TP / (TP + FN)$$

Where:

FP: False Positive

FN: False Negative

TN: True Negative

TP: True Positive

and these terms are explained in table 2:

Table 2. Attacks and alerts terms.

Attacks	Alerts	
	YES	NO
YES	TP	FN
NO	FP	TN

In general, the principle of honeypots considers every

connection with them is an attack because there is no real system or service in honeypots. Therefore, honeypots do not generate false positive since there is no production traffic with them. Consequently, the FPR for honeypots is close to zero. On the other hand, as the honeypots generate alarms for any interaction with them, and these interactions are definitely attacks, therefore honeypots do not generate false negative. Consequently, the TPR becomes closer to one. We conclude that the proposed framework tends to make TPR ratio closer to one, and the FPR ratio closer to zero compared with traditional IDS systems.

## 6. Conclusion and Future Work

The proposed hybrid honeypot system provides many levels of security that decrease the probability of an attacker targeting production systems. This design is achieved by implementing Honeyd systems, and making these systems look like physical machines; these Honeyds provide the first level of security. The second level is the honeynets, which are placed behind the Honeyds. This design makes the attackers unable to differentiate honeypots from production systems, thus the attackers consume their resources and waste their time on honeypots instead of attacking production systems. The results of this work show that the performance of the IDS is improved compared with traditional IDS. Furthermore, the false positive rate is reduced, and the true positive rate is enhanced. In this work, we do not analyze the gathered information in the honeynets. For future works, we suggest to analyze this valuable information to create comprehensive database, which includes all actions that are gathered from attackers. This database can be used to design a predictive system that creates proactive rules for new attacks. Furthermore, it is possible to take advantage of the virtualization feature that we have employed to emulate honeypots to implement them in Software-defined networking (SDN).

## References

- [1] Qassim Nasir and Zahraa Al-Mousa, 'Honeypots Aiding Network Forensics: Challenges and Notions', journal of communication, Vol. 8, No. 11, November 2013.
- [2] Spitzer Lance: 'Honeypot Tracking Hackers', Addison Wesley, 2002.
- [3] Pouget F., Holz T., 'A Pointillist Approach for Comparing Honeypots'. In: Julisch K., Kruegel C. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2005. Lecture Notes in Computer Science, Vol. 3548. Springer, Berlin, Heidelberg, 2005.
- [4] Niels Provos, Thorsten Holz, 'Virtual Honeypots from Botnet Tracking to Intrusion Detection', Addison Wesley, 2007.
- [5] McGrew, Robert, 'Experiences with honeypot systems: Development, deployment, and analysis', HICSS'06, Proc. of the 39th Annual Hawaii Int. Conf. System Sciences, IEEE, Vol. 9, 2006.
- [6] Niels Provos, 'Honeyd: A Virtual Honeypot Daemon', USENIX Security '03, 2003.
- [7] CERT Polska (NASK), 'Proactive Detection of Security Incidents Honeypots'(The European Network and Information Security Agency (ENISA)), 2012.
- [8] Natalie Weiler, 'Honeypots for distributed denial-of-service attacks', Proc. of Eleventh IEEE Int. Workshops on Enabling Technologies, 2002.
- [9] Snehil Vidw arshi, Atul Tyagi, Rishi Kumar, 'A Discussion about Honeypots and Different Models Based on Honeypot', International Journal of Advanced Computational Engineering and Networking, Vo. 3, No. 8, August 2015.
- [10] Vinu V. Das, 'Honeypot Scheme for Distributed Denial-of-Service', Proc. of the 2009 Int. Conf. on Advanced Computer Control, January 2009, pp. 497-501.
- [11] Mohssen M. Z. E. Mohammed, 'Automated Signature Generation for Zero-day Polymorphic Worms Using a Double-Honeynet', University of Cape Town, February 2012.
- [12] Hrishikesh Arun Deshpande, 'HoneyMesh: Preventing Distributed Denial of Service Attacks using Virtualized Honeypots', International Journal of Engineering Research & Technology (IJERT), Vol. 4, No. 8. August-2015.
- [13] Honeynet project, 'Know Your Enemy: Sebek A kernel based data capture tool', <http://www.honeynet.org/papers/sebek.pdf>.
- [14] Navita Sharma and Sukhwinder Singh Sran, 'Detection of threats in Honeynet using Honeywall', International Journal on Computer Science and Engineering (IJCSE) Vol. 3 No. 10 October 2011.
- [15] Fahim H. Abbasi and R. J. Harris, 'Experiences with a Generation III Virtual Honeynet', Telecommunication Networks and Applications Conference (ATNAC), 2009.