Review on efficient log analysis to evaluate multiple honeypots using ELK

Ibrahim Yahya Mohammed AL-Mahbashi[1,] Prashant Chauhan[2], Shivi Shukla[3] and M. B. Potdar[4]

[1]*PG Student, Network Security, PG School, Gujarat Technological University, Ahmedabad, Gujarat, India*
[2]*Project Scientist, Bhaskaracharya Institute for Space Applications and Geo-informatics, Gandhinagar, India*
[3]*Project Engineer, CDAC, Pune, India*
[4]*Project Director, Bhaskaracharya Institute for Space Applications and Geo-informatics, Gandhinagar, India*

Abstract

Security is still be a concern and an important objective to secure the critical business assists. Network security is part of a whole security mechanism at any organizations. Network security not just implementing safeguards like IPS/IDS, firewalls and so forth, but also continuous evaluation to the effectiveness and efficiency of these safeguards to check if they are doing their job or not. One way to enhance the security in the network is by analyzing the logs collected from different sources. Log analysis can answer so many questions as an example are these safeguards efficient enough or not, so we can be able to review our security mechanisms. In this survey I'm going to examine the efficiency of IDS/IPS and other honeypots with ELK to help me analyze the collected logs.

Keywords: *honeypot, log analysis, log monitoring, network attack.*

1. Introduction

As long as most of the business have shifted to the cyber world their assists are interacting directly to this cyber world, which means their critical assists will be face to face with attackers and criminals. One thing we must focus on is network security to keep monitoring our systems and the safeguards efficiency to mitigate the impact of any potential risks to our critical assists and sensitive information. Log analysis can help the network security guys to extract useful information about every single action happened in their system by which they can detect the malicious activities from their first initiation.

Log analysis can also help us to measure the capabilities and efficiency of our safeguards which they are supposed to be our first line of protection. Logs generated from different sources have all the information needed to evaluate our security to continuously enhance it. The amount of logs collected and the diverse format of logs and unstructured structure drove many network administrators to depend on safeguards without looking back to the logs unless in sever incidents detected from those logs and that leaded to the continuous of system breaches and more of the types of attacks.

In my survey I will use different types of safeguards which they are (conpot, cowire, dionaea, elasticpot, glastopf , and honeytrap) and ELK stack (Elasticsearch, Logstash, Kibana) to handle all the log analysis process on an Ubuntu Server 16.04, beside using kali Linux and windows 8/XP as attackers to use two types of attacks: passive attack and active attack. In this survey I will give brief information about each of them individually to have a clear idea about this experiment.

2. Background

In this chapter I will briefly mention the main elements of this evaluation experiment and their capabilities.

1) **Logs sources:** For my experiment I need to use different types systems that will interact directly with the attacks to evaluate their efficiency and compare between them. I will mention each of them in this section.

- **ConPot:[1][2] :** It is a low interactive honeypot, server-side that is easy to deploy and maintain. It is featured to support simulating some protocols like Hypertext Text Transfer Protocol (HTTP), Simple Network Management Protocol (SNMP). The logs generated from this honeypot will give us useful information about the source of the IP address, the type of the request and the resources that was requested[3].

- **Cowrie**: It is a medium interaction type of honeypot[4] It will be used to detect and log the sell interaction and brute force attack like SSH, and Telnet that will be used by the attacker.

- **Dionaea**: Dionaea is a medium- interaction honey pot used to emulate the windows operating services vulnerabilities, and emulate some protocols like FTP, TFTP,mysql, http,etc.[5][6]

- **Elasticpot**: It is a simple honeypot from elasticsearch.

- **Glastopf**: Is a low interaction web application honeypot[7], that is used to detect suspicious interaction on web applications.

- **Honeytrap[8]:** It is a honeypot used for network security to watch the TCP and/or UDP attacks. Many attackers are fooled with honeytrap and interact with it as real one.

- **Suricata:[9]** Is an open source Intrusion detection system IDS. Its engine is capable to detect real-time intrusions.

2) Log Management : Elasticsearch Logstash Kibana (ELK) stack[10] is going to be used as a centralized server to collect, parse, store, query, and visualize the results in smooth graphical charts, bars, pies, and so forth, to easily understand the reports. ELK installation[11] and deployment is not very complex but you got to install each of them individually and link them to work together as a powerful monitoring system.

Elasticsearch is an open-source full-text search and analytics engine. It allows you to store, search, and analyze big volumes of data quickly and in near real time[12] It is capable to handle variety and different types of searches structured, and unconstructed data. It stores the data in indices that makes retrieving the searched data fast, it is important to know that it is a schema free, which means that if there are two documents of the same type they can have sets of fields[13].

Logstash is an open source tool used to collect the data/logs, parse, index and then store the logs into a specific engine[14] It consists of three main parts (INPUTS, FILTERS, OUTPUTS).

- The INPUT can support variety types of inputs from different sources at the same time, meanwhile.
- The FITLERS parse the transform the data. During the data transfer from the source to where it will be stored, it filters each event and identify named fields. And then converts the collected data to an easy to understand format. It can dynamically fix the issue of logfiles complexity by structuring the unstructured data, locate the source of the attackers by IP address, processing the data with the care of the independency of the sources, format or schema of the data. And it has a rich library filters.
- The OUTPU provides delivering your collected data to any engine of your choice. Routing the data to anywhere you choose gives more flexibility to overcome the constraints and limitations of other competitive tools.

Kibana provides the visualization to the Elasticsearch data. It visualize the results on the dashboard based on your demands you can customize how you want to visualize the output result of your processed data.

ELK advantages

All together can provide a very useful solution to collect, analyze and visualize the data from one or more than one sources. ELK has great capabilities and it handles high scalability very efficiently and can deal with all kinds of data. It helps you to correlate the events and display smoothly and deliver the output in an understandable format that will help in better monitoring your network. Customization is the key that made it the first choice for many companies working in variety different fields.

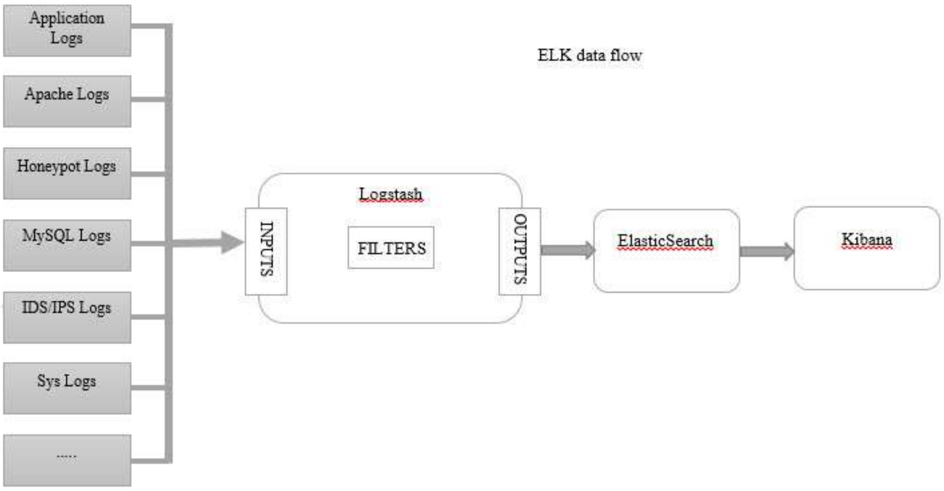


Fig 1: 10 ELK Data Flow

3. Implementation:

Ubuntu server 16.04 will be used a server to install ELK and all other honeypots on it using docker, to allow me to run multiple honeypots on the same network interface and prevent any issues. Docker[15] encapsulate the honeypots and isolate them to run independently and made it easy to update and maintain. This[16] explained in details docker. In the followed figure is the architecture of the implementation.

- **Architecture**:

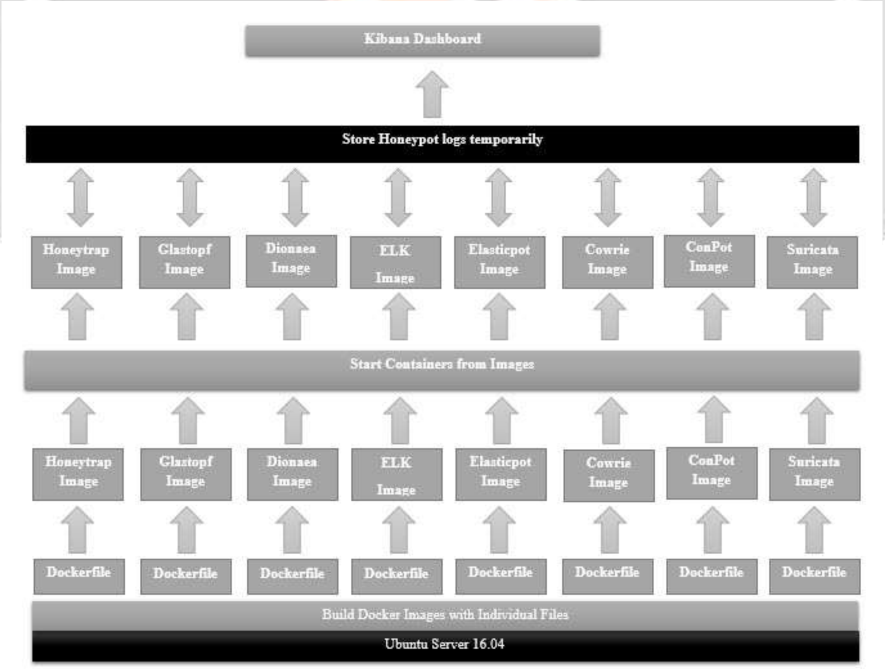


Fig 2: Server Architecture

Honeypot	Protocol	Ports
honeytrap	TCP	25, 110, 139, 3389, 4444, 4899, 5900, 21000
dionaea	TCP	21, 42, 135, 443, 445, 1433, 1723, 1883, 1900, 3306, 5060, 5061, 8081, 11211

dionaea	UDP	69, 5060
conpot	TCP	1025, 50100
cowrie	TCP	22, 23
elasticpot	TCP	9200

Table 1: Honeypot Protocol Ports

- **Attack Simulation**

For simulating the attack I'm going to use kali linux, windows XP, and Windows 8 as attacker. And I'm going to try different tools and methods to try different types of attacks. Mainly I'm going to do the attack in to two parts:

1) Passive attack: When the attackers have a target they need to collect as much information as they can about their victim. They use many tools and techniques to achieve this goal, because the information will make the hacking easier for them and will lead them to choose the proper tools and techniques to hack that systems.

For this kind of attack I will use "nmap"[17] tool to do a port scanning.

- **Port scanning:** special tools used to scan the ports on the systems to get information about the status of the ports, which ports are opened and what are the services running on that ports. By knowing the open ports and the services running the attackers will use that ports to get access to the victim systems. The ports are like the window by which attackers will try to launch their attack.

A very powerful tool used is NMAP which has different capabilities that can scan the ports

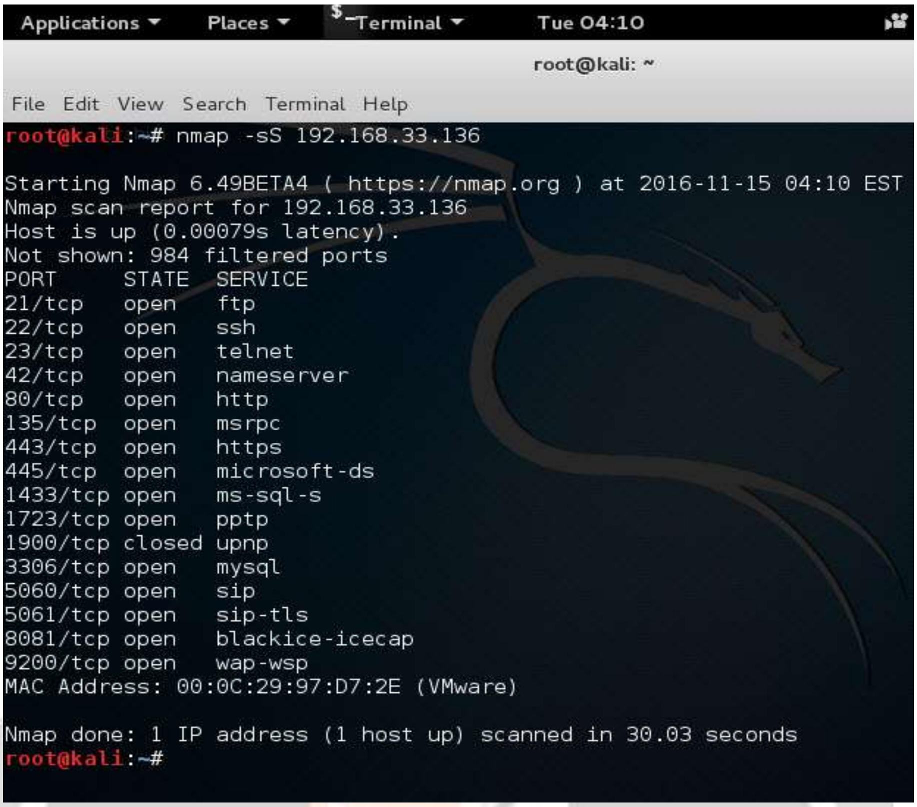
Here are some of the nmap commands[18] I tried and their task:

Syntax:

nmap [Scan Type(s)] [Options] {target specification}

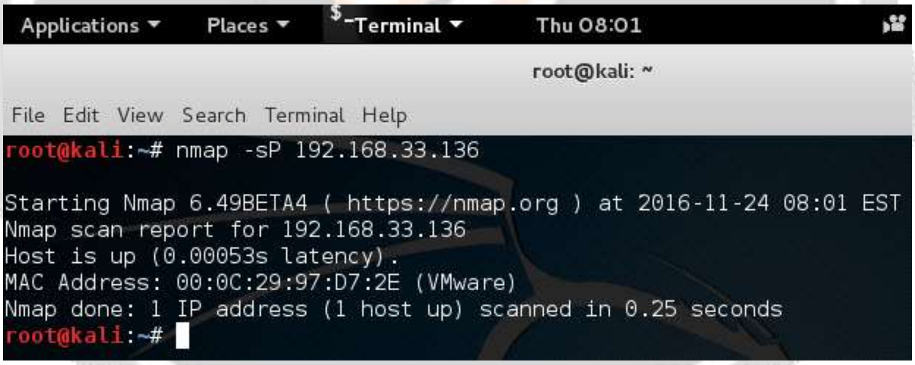
No.	Nmap Command	Task
1	nmap –sS 192.168.33.136	SYN Stealth Scan - Perform a stealthy Scan
2	nmap –sP 192.168.33.136	Ping Scan - Find out Live hosts in a Network
3	nmap –sA 192.168.33.136	ACK Scan - Scan a Host to Detect Firewall
4	nmap –O 192.168.33.136	OS Fingerprinting - Enable OS Detection with Nmap

Table 2: nmap [Scan Type(s)] [Options] {target specification}



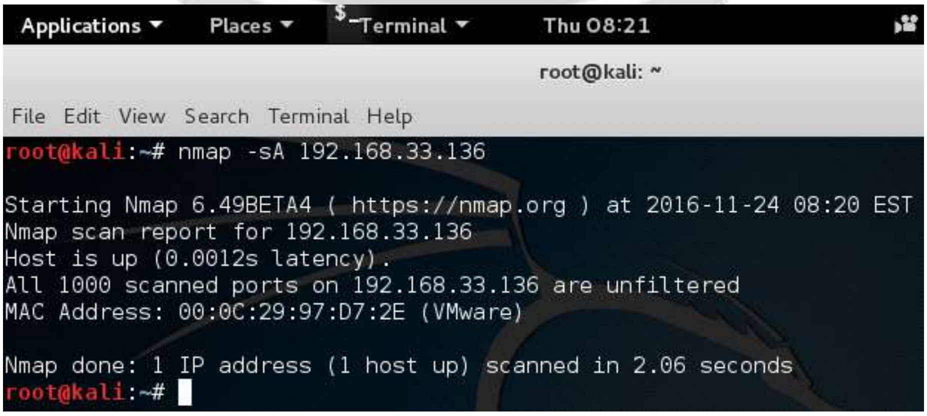
```
root@kali:~# nmap -sS 192.168.33.136

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-11-15 04:10 EST
Nmap scan report for 192.168.33.136
Host is up (0.00079s latency).
Not shown: 984 filtered ports
PORT      STATE  SERVICE
21/tcp    open   ftp
22/tcp    open   ssh
23/tcp    open   telnet
42/tcp    open   nameserver
80/tcp    open   http
135/tcp   open   msrpc
443/tcp   open   https
445/tcp   open   microsoft-ds
1433/tcp  open   ms-sql-s
1723/tcp  open   pptp
1900/tcp  closed upnp
3306/tcp  open   mysql
5060/tcp  open   sip
5061/tcp  open   sip-tls
8081/tcp  open   blackice-icecap
9200/tcp  open   wap-wsp
MAC Address: 00:0C:29:97:D7:2E (VMware)

Nmap done: 1 IP address (1 host up) scanned in 30.03 seconds
root@kali:~#
```

Fig 3: NMAP - Scan using TCP SYN scan



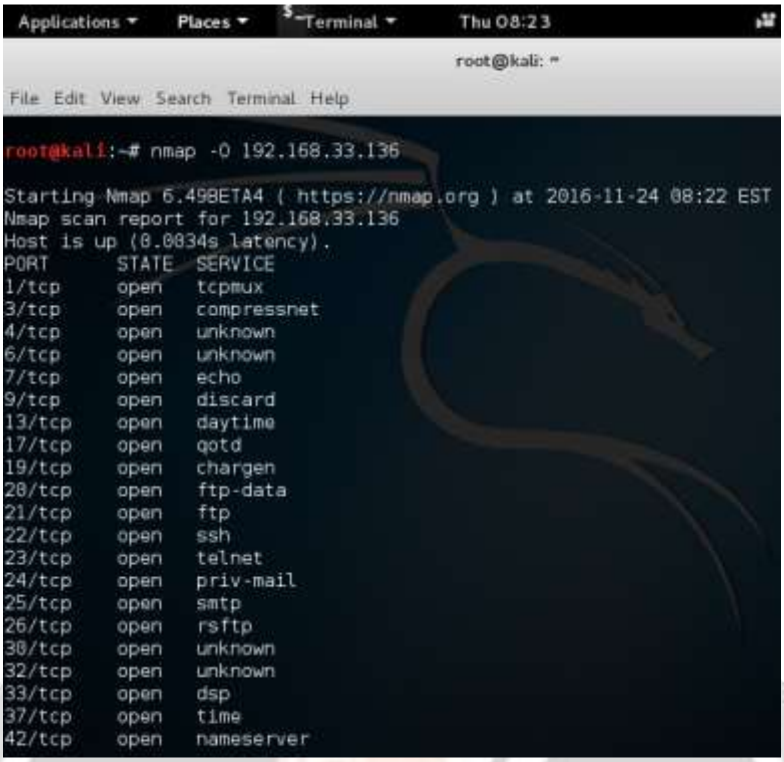
```
root@kali:~# nmap -sP 192.168.33.136

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-11-24 08:01 EST
Nmap scan report for 192.168.33.136
Host is up (0.00053s latency).
MAC Address: 00:0C:29:97:D7:2E (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
root@kali:~#
```

Fig 4: nmap Ping Scan



```
root@kali:~# nmap -sA 192.168.33.136

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-11-24 08:20 EST
Nmap scan report for 192.168.33.136
Host is up (0.0012s latency).
All 1000 scanned ports on 192.168.33.136 are unfiltered
MAC Address: 00:0C:29:97:D7:2E (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2.06 seconds
root@kali:~#
```

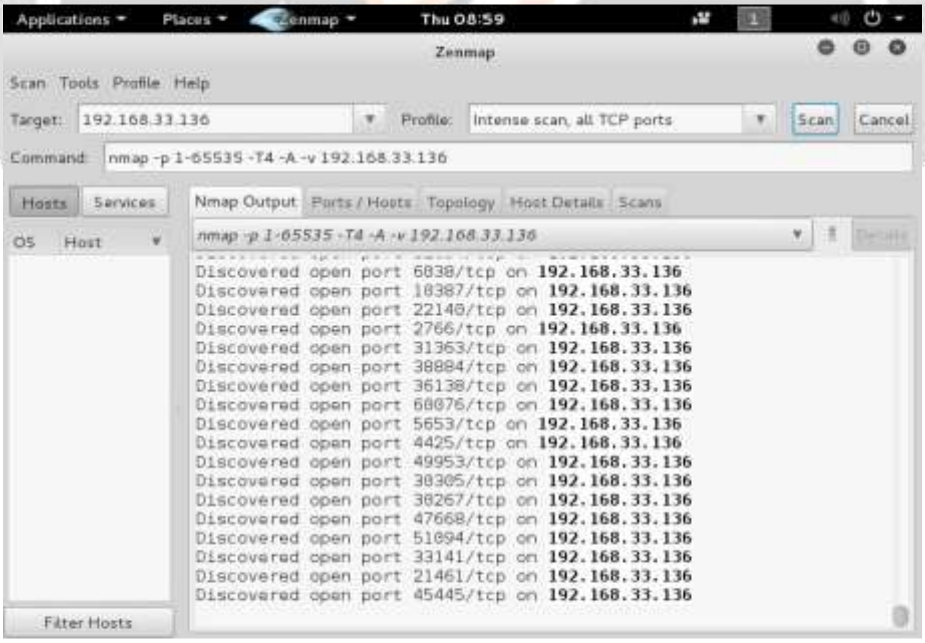
Fig 5: nmap ACK Scan



```
root@kali:~# nmap -O 192.168.33.136

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-11-24 08:22 EST
Nmap scan report for 192.168.33.136
Host is up (0.0034s latency).
PORT     STATE  SERVICE
1/tcp    open   tcpmux
3/tcp    open   compressnet
4/tcp    open   unknown
6/tcp    open   unknown
7/tcp    open   echo
9/tcp    open   discard
13/tcp   open   daytime
17/tcp   open   qotd
19/tcp   open   chargen
20/tcp   open   ftp-data
21/tcp   open   ftp
22/tcp   open   ssh
23/tcp   open   telnet
24/tcp   open   priv-mail
25/tcp   open   smtp
26/tcp   open   rsftp
30/tcp   open   unknown
32/tcp   open   unknown
33/tcp   open   dsp
37/tcp   open   time
42/tcp   open   nameserver
```

Fig 6: nmap ACK Scan

Now I will zenmap in kali linux to scan all TCP ports on the server



Target: 192.168.33.136
Profile: Intense scan, all TCP ports
Command: nmap -p 1-65535 -T4 -A -v 192.168.33.136

```
nmap -p 1-65535 -T4 -A -v 192.168.33.136
Discovered open port 6838/tcp on 192.168.33.136
Discovered open port 18387/tcp on 192.168.33.136
Discovered open port 22140/tcp on 192.168.33.136
Discovered open port 2766/tcp on 192.168.33.136
Discovered open port 31363/tcp on 192.168.33.136
Discovered open port 38884/tcp on 192.168.33.136
Discovered open port 36138/tcp on 192.168.33.136
Discovered open port 60076/tcp on 192.168.33.136
Discovered open port 5653/tcp on 192.168.33.136
Discovered open port 4425/tcp on 192.168.33.136
Discovered open port 49953/tcp on 192.168.33.136
Discovered open port 38305/tcp on 192.168.33.136
Discovered open port 38267/tcp on 192.168.33.136
Discovered open port 47668/tcp on 192.168.33.136
Discovered open port 51094/tcp on 192.168.33.136
Discovered open port 33141/tcp on 192.168.33.136
Discovered open port 21461/tcp on 192.168.33.136
Discovered open port 45445/tcp on 192.168.33.136
```

Fig 7: :enmap - all TCP ports scan

For space limitations I will not able to show all the results founds and I am only concerned in this paper to simulate the attack.

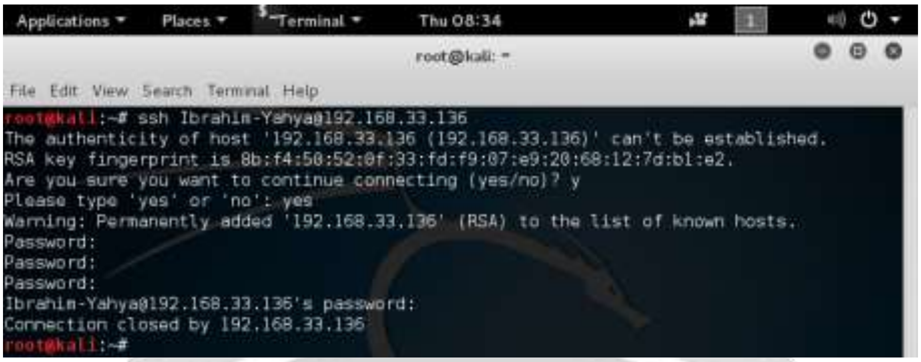
- Active attack:

The passive attack is the attack in which the attacker start to interact directly to the victim's system and attempt to affect the operation and make modification in the system or attack the availability of the services provided by the organizations like DoS attack and DDoS attack specially in the cloud based services [21].

As an example of this is when the attacker finds vulnerabilities in the system and attempt to exploit that vulnerabilities and maintain his attack to get more control.

For this type of attacks I will try to do some active attacks by interacting directly with the server by trying using some ports are opened.

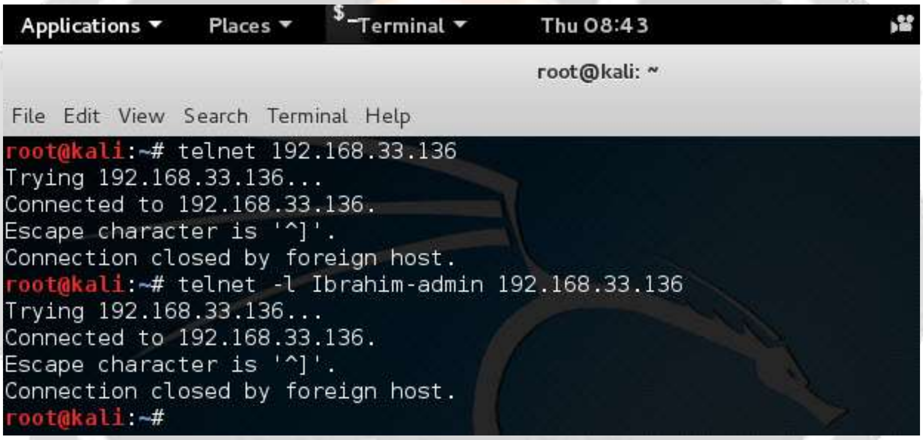
First I will use ssh command to try connect remotely to this server using wrong username and password.



```
root@kali:~# ssh Ibrahim-Yahya@192.168.33.136
The authenticity of host '192.168.33.136 (192.168.33.136)' can't be established.
RSA key fingerprint is 8b:f4:50:52:0f:33:fd:f9:07:e9:20:68:12:7d:b1:e2.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '192.168.33.136' (RSA) to the list of known hosts.
Password:
Password:
Password:
Ibrahim-Yahya@192.168.33.136's password:
Connection closed by 192.168.33.136
root@kali:~#
```

Fig 8: SSh remote login

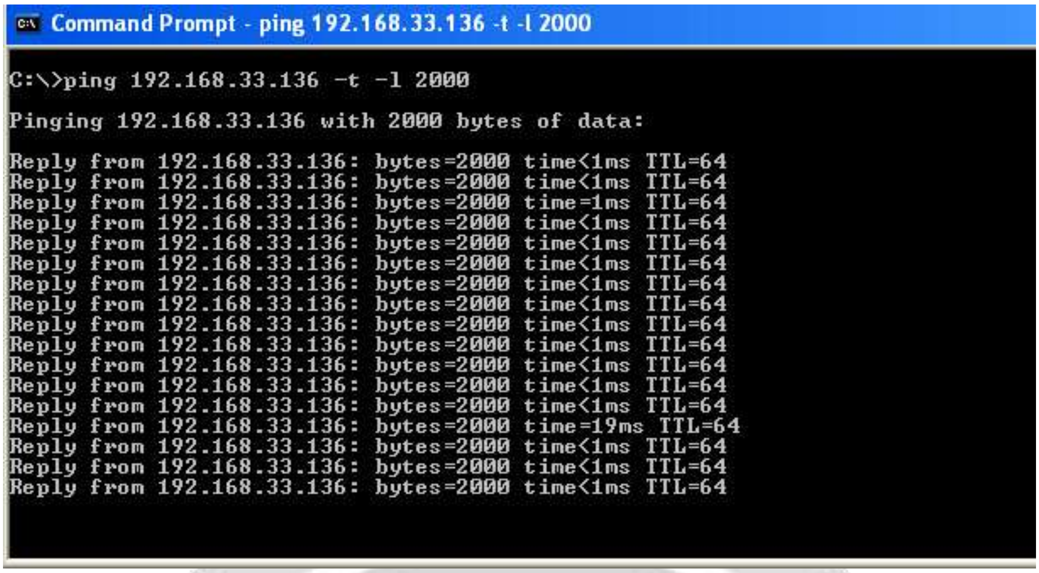
Now I will try to login using telnet one time without username and one time with anonymous username.



```
root@kali:~# telnet 192.168.33.136
Trying 192.168.33.136...
Connected to 192.168.33.136.
Escape character is '^]'.
Connection closed by foreign host.
root@kali:~# telnet -l Ibrahim-admin 192.168.33.136
Trying 192.168.33.136...
Connected to 192.168.33.136.
Escape character is '^]'.
Connection closed by foreign host.
root@kali:~#
```

Fig 9: telnet login

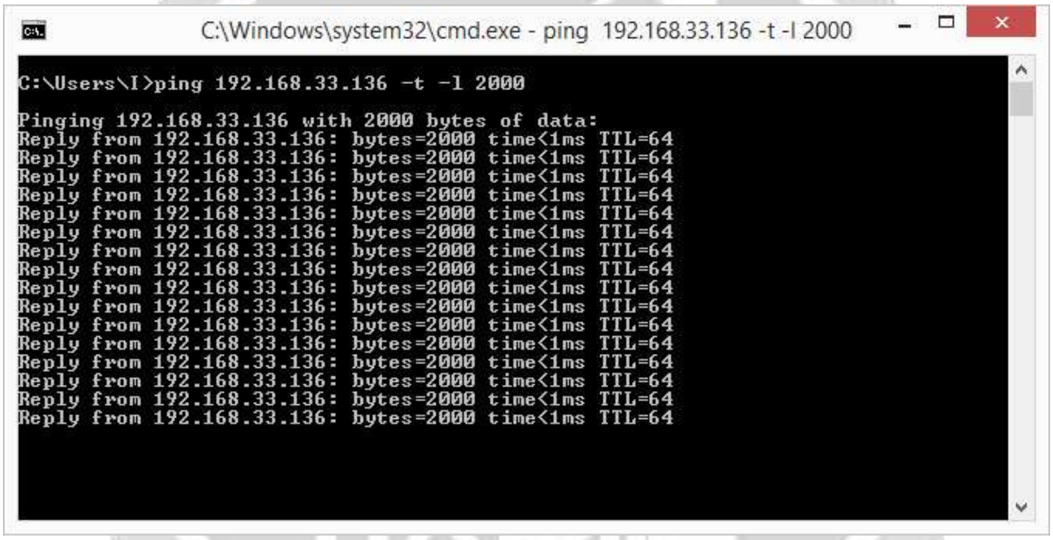
DoS attack can be done using ping command even by command line from windows, so I will use two machines to ping to the server from windows 8 with the IP address 192.168.10.102 and from windows xp with IP address: 192.168.33.132 and I will ping in the same time by both machines to simulate DDoS attack.



Command Prompt - ping 192.168.33.136 -t -l 2000

```
C:\>ping 192.168.33.136 -t -l 2000

Pinging 192.168.33.136 with 2000 bytes of data:

Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time=1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time=19ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
```

Fig 10: Simple DoS attack by win XP



C:\Windows\system32\cmd.exe - ping 192.168.33.136 -t -l 2000

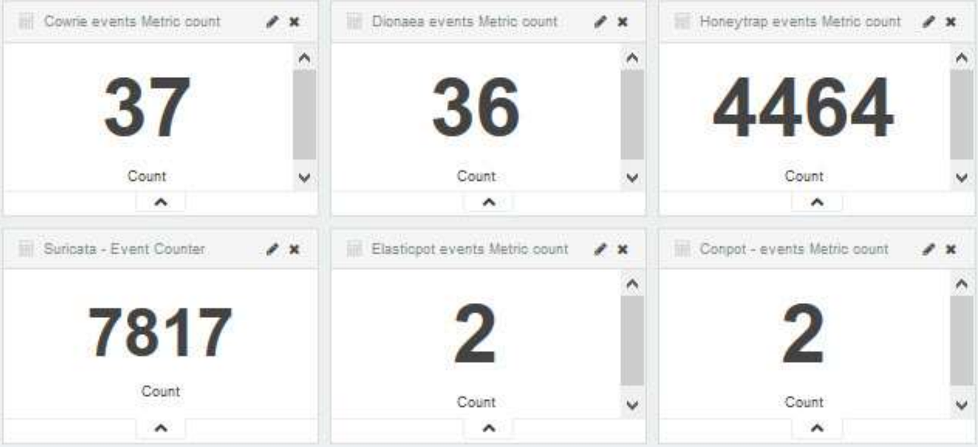
```
C:\Users\I>ping 192.168.33.136 -t -l 2000

Pinging 192.168.33.136 with 2000 bytes of data:
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
Reply from 192.168.33.136: bytes=2000 time<1ms TTL=64
```

Fig 11: Simple DoS attack using win 8

ELK Report Dashboard:

Now after simulating some types of attacks I have customized the dashboard to fit our need for this survey to compare between all honeypots we have implemented to for evaluation.

First I have created special counters to count all the events being captured by all honeypots individually and demonstrated them as you will see in the followed figure. And I will narrow the report to show only today’s events that are related to our survey.



Cowrie events Metric count
37
Count

Dionaea events Metric count
36
Count

Honeytrap events Metric count
4464
Count

Suricata - Event Counter
7817
Count

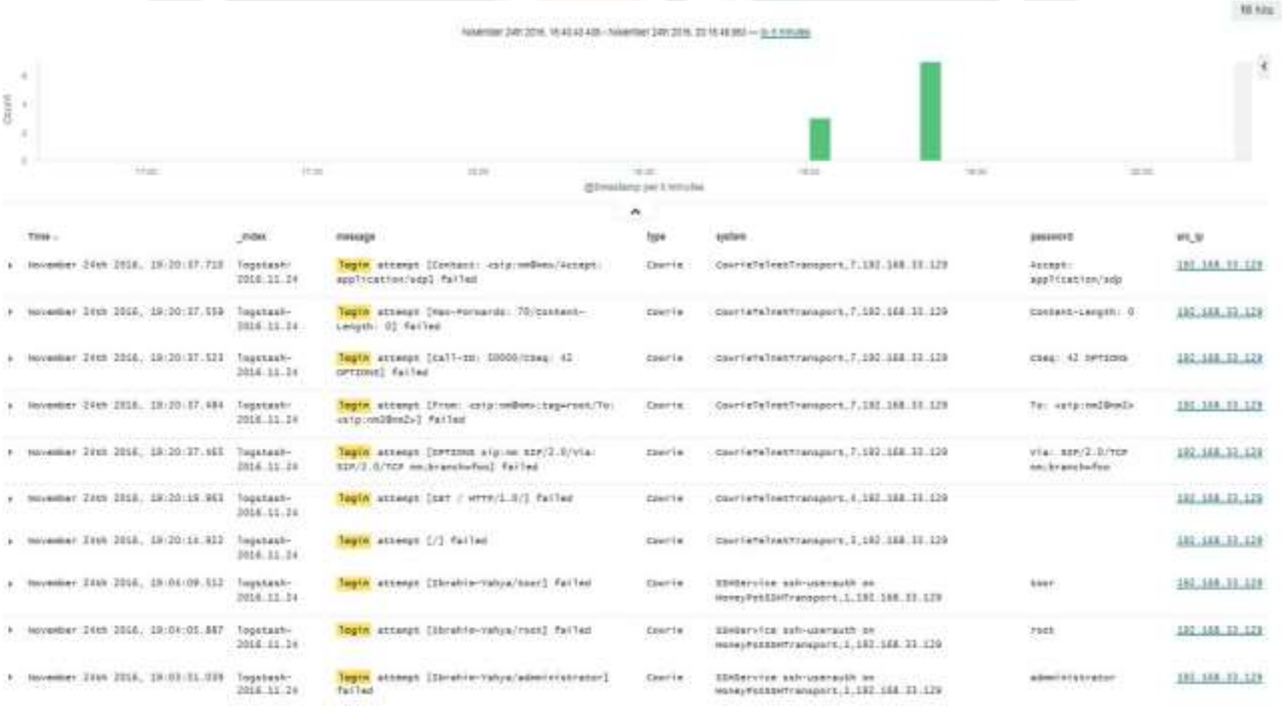
Elasticpot events Metric count
2
Count

Conpot - events Metric count
2
Count

Fig 12: Events count in all honeypots

As we can see that the total count of events collected vary from one pot to another. Honeypot count events are the largest one then the cowire and the least once are the conpot and elasticpot.

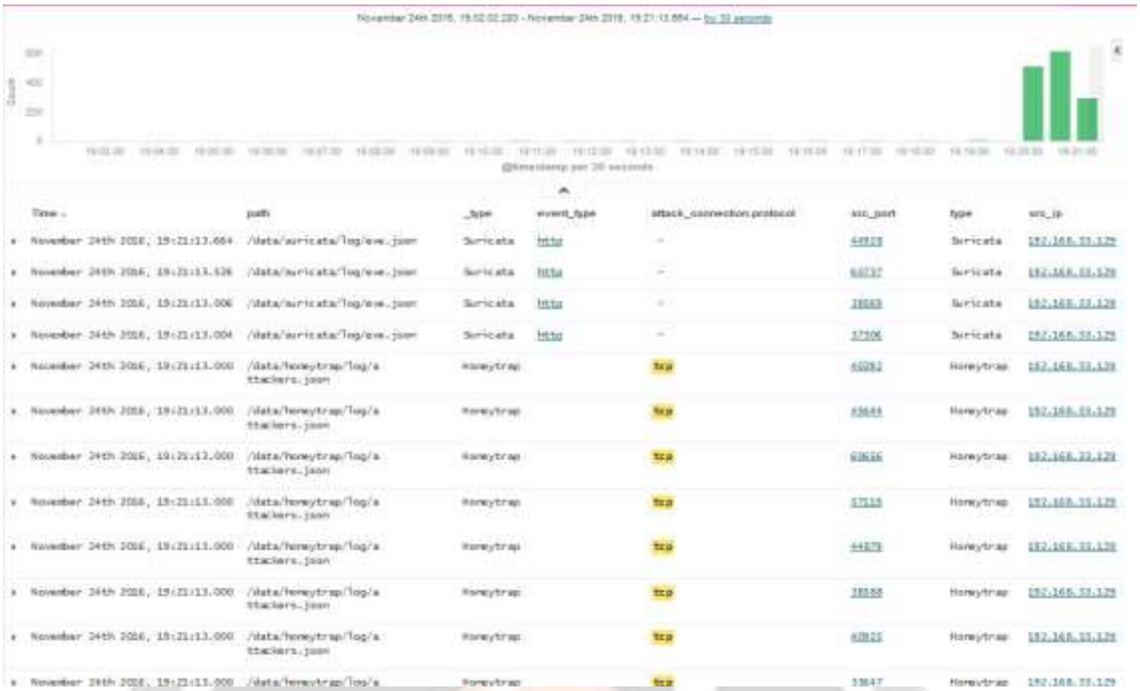
In the following I have filtered the logs that has all the attempts I used to login remotely and it showed what username and password I have used for the attack, that will help in studying the attackers behaviors and that will give us a clear idea about the techniques and methods used by attackers and how much knowledge they have about our system. As an example the admin has changed his password and he discovers in the log files that there are some attempts to login using the previews password, in this case the attacker could be an internal employee or could be an attacker who knows by somehow our sensitive information and this is a serious issue that we have to be concerned about.



Time	_index	message	type	system	password	src_ip
November 24th 2016, 19:20:37.718	logstash-2016.11.24	login attempt [Contact: <sip:nm@nm>/Accept: application/sdp] failed	Cowrie	CowrieTelnetTransport,7,192.168.33.129	Accept: application/sdp	192.168.33.129
November 24th 2016, 19:20:37.559	logstash-2016.11.24	login attempt [Max-Forwards: 70/Content-Length: 0] failed	Cowrie	CowrieTelnetTransport,7,192.168.33.129	Content-Length: 0	192.168.33.129
November 24th 2016, 19:20:37.523	logstash-2016.11.24	login attempt [Call-ID: 50000/CSeq: 42 OPTIONS] failed	Cowrie	CowrieTelnetTransport,7,192.168.33.129	CSeq: 42 OPTIONS	192.168.33.129
November 24th 2016, 19:20:37.484	logstash-2016.11.24	login attempt [From: <sip:nm@nm>;tag=root/To: <sip:nm2@nm2>] failed	Cowrie	CowrieTelnetTransport,7,192.168.33.129	To: <sip:nm2@nm2>	192.168.33.129
November 24th 2016, 19:20:37.465	logstash-2016.11.24	login attempt [OPTIONS sip:nm SIP/2.0/Via: SIP/2.0/TCP nm;branch=foo] failed	Cowrie	CowrieTelnetTransport,7,192.168.33.129	Via: SIP/2.0/TCP nm;branch=foo	192.168.33.129
November 24th 2016, 19:20:15.963	logstash-2016.11.24	login attempt [GET / HTTP/1.0/] failed	Cowrie	CowrieTelnetTransport,4,192.168.33.129		192.168.33.129
November 24th 2016, 19:20:14.922	logstash-2016.11.24	login attempt [/] failed	Cowrie	CowrieTelnetTransport,3,192.168.33.129		192.168.33.129
November 24th 2016, 19:04:09.512	logstash-2016.11.24	login attempt [Ibrahim-Yahya/user] failed	Cowrie	SSHService ssh-userauth on HoneyPotSSHTransport,1,192.168.33.129	user	192.168.33.129
November 24th 2016, 19:04:05.887	logstash-2016.11.24	login attempt [Ibrahim-Yahya/root] failed	Cowrie	SSHService ssh-userauth on HoneyPotSSHTransport,1,192.168.33.129	root	192.168.33.129
November 24th 2016, 19:03:51.039	logstash-2016.11.24	login attempt [Ibrahim-Yahya/administrator] failed	Cowrie	SSHService ssh-userauth on HoneyPotSSHTransport,1,192.168.33.129	administrator	192.168.33.129

Fig 13: Kibana - Login attempts logs

With more details I have intersected the parts of logs that could be interesting you can notice which honeypot has detect the attempts to remotely login and that, as it was shown that Cowrie honeypot has exclusively detected all the login attempts to the server and provided very important details.



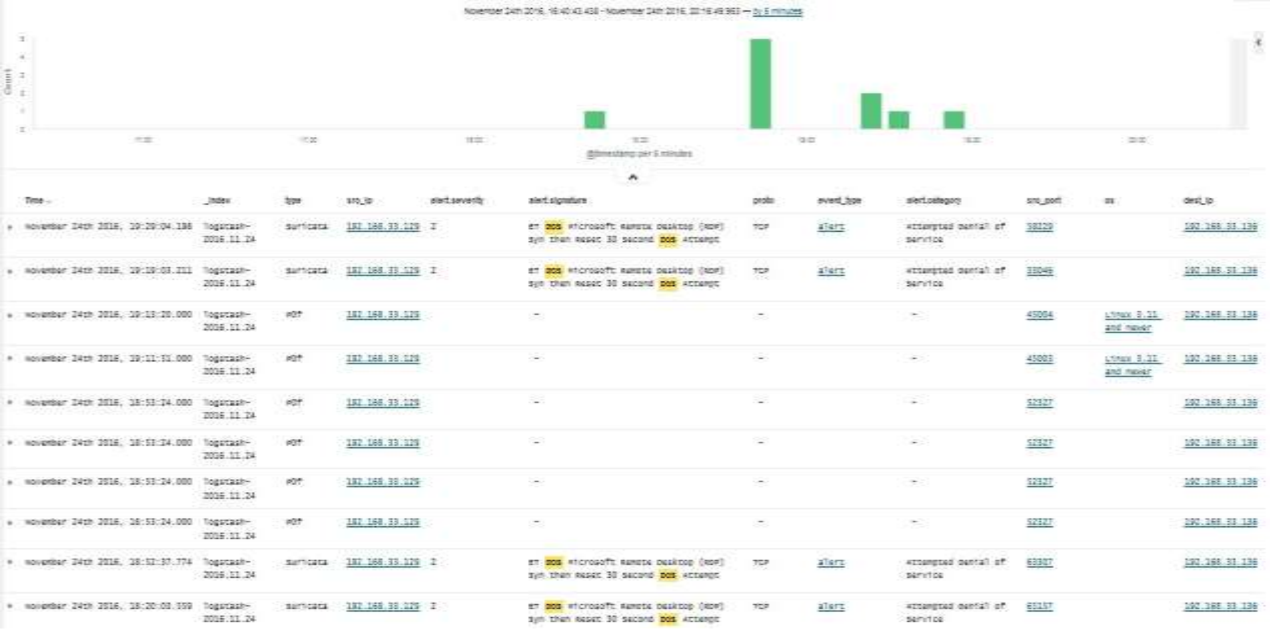
November 24th 2016, 19:02:02.283 - November 24th 2016, 19:21:13.664 — by 30 seconds

	Time	path	_type	event_type	attack_connection.protocol	src_port	type	src_ip
▸	November 24th 2016, 19:21:13.664	/data/suricata/log/eve.json	Suricata	http	-	44929	Suricata	192.168.33.129
▸	November 24th 2016, 19:21:13.526	/data/suricata/log/eve.json	Suricata	http	-	60737	Suricata	192.168.33.129
▸	November 24th 2016, 19:21:13.006	/data/suricata/log/eve.json	Suricata	http	-	38583	Suricata	192.168.33.129
▸	November 24th 2016, 19:21:13.004	/data/suricata/log/eve.json	Suricata	http	-	37306	Suricata	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/attackers.json	Honeytrap		tcp	45283	Honeytrap	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/attackers.json	Honeytrap		tcp	45644	Honeytrap	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/attackers.json	Honeytrap		tcp	60656	Honeytrap	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/attackers.json	Honeytrap		tcp	37115	Honeytrap	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/attackers.json	Honeytrap		tcp	44879	Honeytrap	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/attackers.json	Honeytrap		tcp	38588	Honeytrap	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/attackers.json	Honeytrap		tcp	40825	Honeytrap	192.168.33.129
▸	November 24th 2016, 19:21:13.000	/data/honeytrap/log/a	Honeytrap		tcp	33647	Honeytrap	192.168.33.129

Fig 14: Protocol attacked with IP source

As it is clear from the logs collected that all the honeypots have detected all attacks against TCP port and recorded the IP address of the attacker with a timestamp to easily track all the activities of the malicious attackers.

In the following figure I have filtered the log files of all honeypots to detect only the attacks that targeted my OS because usually hackers will attack the OS fingerprinting when they find any vulnerability related to, clearly p0f [19]and

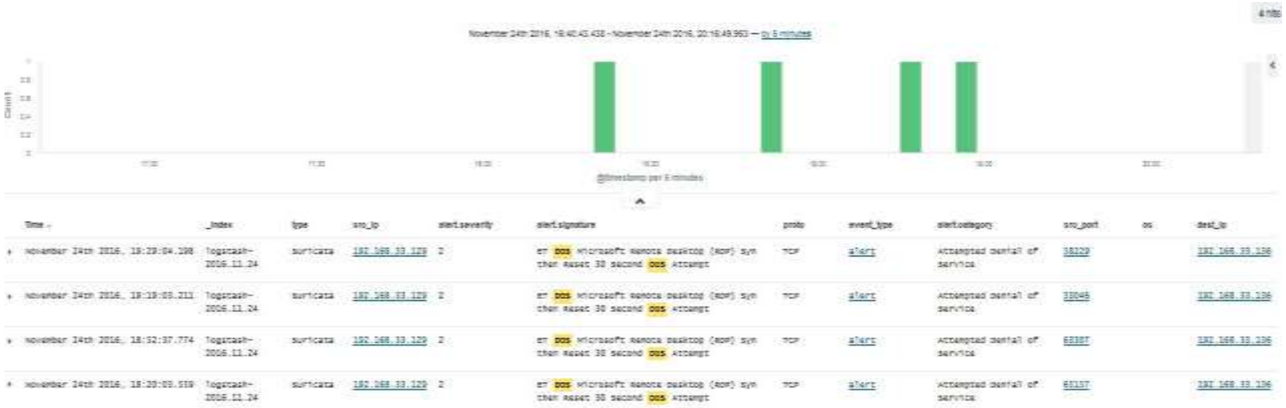


November 24th 2016, 16:40:43.438 - November 24th 2016, 22:16:48.963 — by 5 minutes

	Time	_index	type	src_ip	alert.severity	alert.signature	proto	event_type	alert.category	src_port	os	dest_ip
▸	November 24th 2016, 19:29:04.138	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	58229		192.168.33.136
▸	November 24th 2016, 19:19:03.211	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	33049		192.168.33.136
▸	November 24th 2016, 19:13:20.000	logstash-2016.11.24	p0f	192.168.33.129		-	-		-	45004	Linux 3.11 and newer	192.168.33.136
▸	November 24th 2016, 19:11:31.000	logstash-2016.11.24	p0f	192.168.33.129		-	-		-	45003	Linux 3.11 and newer	192.168.33.136
▸	November 24th 2016, 18:53:24.000	logstash-2016.11.24	p0f	192.168.33.129		-	-		-	52327		192.168.33.136
▸	November 24th 2016, 18:53:24.000	logstash-2016.11.24	p0f	192.168.33.129		-	-		-	52327		192.168.33.136
▸	November 24th 2016, 18:53:24.000	logstash-2016.11.24	p0f	192.168.33.129		-	-		-	52327		192.168.33.136
▸	November 24th 2016, 18:53:24.000	logstash-2016.11.24	p0f	192.168.33.129		-	-		-	52327		192.168.33.136
▸	November 24th 2016, 18:52:37.774	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	63307		192.168.33.136
▸	November 24th 2016, 18:20:03.559	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	65157		192.168.33.136

Fig 15: OS fingerprint

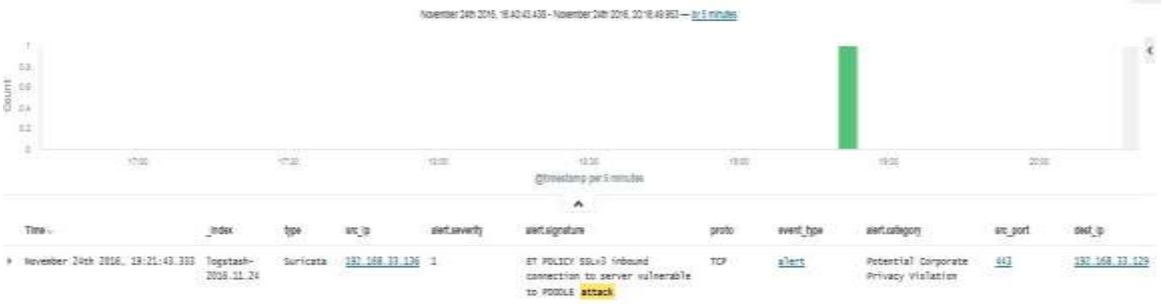
The attempts of DoS attack was detected in the following figure showing the attacks from attackers and even though Suricata IDS was the only one who detected such attack but it detected the attack from one source IP address, and I have used three machines to simulate this type of attack. The information revealed showed the IP address of the attacker with the timestamp beside the severity of the alert level.


4 hits
November 24th 2016, 18:40:43.438 - November 24th 2016, 20:18:49.963 — by 5 minutes
Count
@timestamp per 5 minutes

	Time	_index	type	src_ip	alert.severity	alert.signature	proto	event_type	alert.category	src_port	os	dest_ip
▸	November 24th 2016, 19:29:04.198	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	38229		192.168.33.136
▸	November 24th 2016, 19:19:03.211	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	33046		192.168.33.136
▸	November 24th 2016, 18:52:37.774	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	63387		192.168.33.136
▸	November 24th 2016, 18:29:03.519	logstash-2016.11.24	suricata	192.168.33.129	2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DOS Attempt	TCP	alert	Attempted Denial of Service	65157		192.168.33.136

Fig 16: DoS attack attempts

Another type of attack detected Suricate IDS has the only one that detected the poodle (Padding Oracle On Downgraded Legacy Encryption) vulnerability[20] in the server and the attempt to attack the ssl that will allow the attacker to steal the cookies of the HTTP as it is shown in the following figure that revealed information about the source of the attack and the severity level that was considered number 1.


November 24th 2016, 18:40:43.438 - November 24th 2016, 20:18:49.963 — by 5 minutes
Count
1
0.8
0.6
0.4
0.2
0
17:00
17:30
18:00
18:30
19:00
19:30
20:00
@timestamp per 5 minutes

	Time	_index	type	src_ip	alert.severity	alert.signature	proto	event_type	alert.category	src_port	dest_ip
▸	November 24th 2016, 19:21:43.333	logstash-2016.11.24	Suricata	192.168.33.136	1	ET POLICY SSLv3 inbound connection to server vulnerable to POODLE attack	TCP	alert	Potential Corporate Privacy Violation	443	192.168.33.129

Fig 17: SSL POODLE attack

Having knowledge about the attackers will help us to enhance our security mechanisms and to know what capabilities they have to refine our methodologies to handle different types of attacks used against us. One of the information that was revealed is the type of operating systems used by the attackers as it will be shown in the following figure.

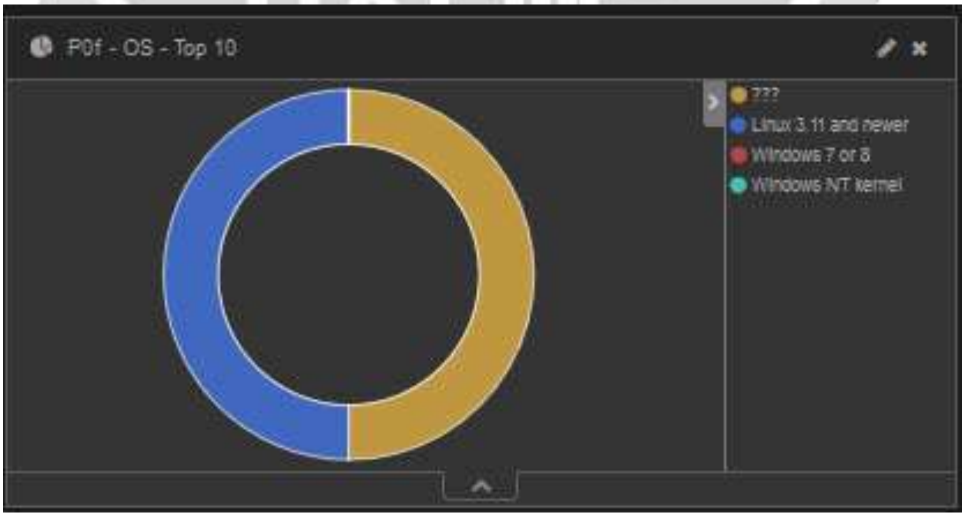


Fig 18: Attackers OS systems

Result:

After simulating different types of attacks using different types of operation systems against my server to evaluate the different types of honeypots and IDS system to compare their capabilities of detecting the attacks and reveal information about the source of the attacks. In the following table will show details about the results found in this survey.

Honeypot, IDS Name	Total Events captured	DoS attack	Attempts to remotely login	Detected OS of Attacker	Revealed Username/password used by hackers	Detected exploiting vulnerabilities
honeytrap	4464	No	No	No	No	No
dionaea	36	No	No	No	No	No
conpot	2	No	No	No	No	No
cowrie	37	No	Yes	No	Yes	No
elasticpot	2	No	No	No	No	No
Suricata	7817	Yes	No	Yes	No	Yes

Table 3 : Result

From this table we can notice that Suricata IDS has shown more capabilities to detect most of the events and detected different types of attacks beside revealing the some attempts to exploit a vulnerability in the OS found by the attacker targeting a weakness in the ssl to steal the cookies of the HTTP. In the another hand honeypot was the second in detecting the events, and that will help to dig deeper in its logs. One interesting thing about cowire honeypot is its capability to reveal the username and password used by attackers which can help us to study the attackers behavior and determine its identity if he has being discovered as an internal attacker as unfaithful employee, or may help us to know more about how much information the attackers have about us and our internal security mechanisms.

4. Conclusion:

As you can see that each honeypot and IDS has different capabilities to capture the events. Some were able to detect more events than others but it was not able to detect some of the attacks like others. One of them only was able to detect the attempts to remotely logins and revealed important information about the username and password used by hackers that will help to answer many questions related to our security mechanism by cowire honeypot, and that feature must be added in all IDS and honeypots. Many more useful information was found in the log files generated by different sources but due to the space limitation I was not able to show more information and capabilities that ELK stack can offer to handle all the processes of log file life cycle starting from collecting the logs from different sources until the smooth visualization of the customizable reports that can offer, that will be shown in the future.

After overall we can notice that implementing IDS/IPS and honeypots and monitoring the network is very important for securing the network to protect the business assets but we should pay more attention to analyze the log files generated by those safeguards to evaluate their efficiency and see if they are working properly as needed or we have to study the reasons behind its not efficiency.

5. Acknowledgment:

I would like to thank all the people who helped me and guided me during writing this paper. And a special gratitude to my guiders Dr.Madhukar PotdarScientist and Project Director, Mr. Prashant Chauhan Project Scientist, HPC ApplicationFrom BISAG – Gandhinagar India and Miss. Shivi Shukla Project Engineer at CDAC R&d for guiding me through my all phases of my ME project and advising me with their valuable information.

6. References:

[1] Serbanescu, A. V., Obermeier, S., & Yu, D. Y. (2015, September). ICS threat analysis using a large-scale honeynet. In Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research (pp. 20-30). British Computer Society.

[2] Jicha, A., Patton, M., & Chen, H. (2016, November). SCADA honeypots: An in-depth analysis of Conpot. In Intelligence and Security Informatics (ISI), 2016 IEEE Conference on (pp. 196-198). IEEE.

[3] D. Buza, F. Juhasz, and G. Miru. “Design and implementation of critical infrastructure protection system,” Budapest University of Technology and Economics, Department of Networked Systems and Services, 2013.

[4] Zammit, D. (2016). A machine learning based approach for intrusion prevention using honeypot interaction patterns as training data (Bachelor's thesis, University of Malta).

[5] Sochor, T., Zuzcak, M., & Bujok, P. (2016, July). Analysis of attackers against windows emulating honeypots in various types of networks and regions. In Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on (pp. 863-868). IEEE.

[6] Sochor, T., & Zuzcak, M. (2014, June). Study of internet threats and attack methods using honeypots and honeynets. In International Conference on Computer Networks (pp. 118-127). Springer International Publishing.

[7] Mphago, B., Bagwasi, O., Phofuetsile, B., & Hlomani, H. (2015, January). Deception in Dynamic Web Application Honeypots: Case of Glastopf. In Proceedings of the International Conference on Security and Management (SAM) (p. 104). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[8] Werner, T. Honeytrap. Dostupno na http://sourceforge. net/projects/honeytrap.

[9] Day, D., & Burns, B. (2011, February). A performance analysis of snort and suricata network intrusion detection and prevention engines. In Fifth International Conference on Digital Society, Gosier, Guadeloupe (pp. 187-192).

[10] Bagnasco, S., Berzano, D., Guarise, A., Lusso, S., Masera, M., & Vallero, S. (2015). Monitoring of IaaS and scientific applications on the Cloud using the Elasticsearch ecosystem. In Journal of Physics: Conference Series (Vol. 608, No. 1, p. 012016). IOP Publishing.

[11] Anicas, M. (2015). How To Install Elasticsearch, Logstash, and Kibana (ELK Stack) on Ubuntu 14.04.

[12] https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html

[13] Sasirekha, G. V. K., & Dasari, S. R. (2016, February). Big Spectrum Data Analysis in DSA Enabled LTE-A Networks: A System Architecture. In Advanced Computing (IACC), 2016 IEEE 6th International Conference on(pp. 655-660). IEEE.

[14] Xu, X., Weber, I., Bass, L., Zhu, L., Wada, H., & Teng, F. (2013, December). Detecting cloud provisioning errors using an annotated process model. In Proceedings of the 8th Workshop on Middleware for Next Generation Internet Computing (p. 5). ACM.

[15] Chung, M. T., Quang-Hung, N., Nguyen, M. T., & Thoai, N. (2016, July). Using Docker in high performance computing applications. In Communications and Electronics (ICCE), 2016 IEEE Sixth International Conference on (pp. 52-57). IEEE.

[16] Combe, T., Martin, A., & Di Pietro, R. (2016). To Docker or Not to Docker: A Security Perspective. IEEE Cloud Computing, 3(5), 54-62.

[17] Lyon, G. F. (2009). Nmap network scanning: The official Nmap project guide to network discovery and security scanning. Insecure.

[18] Czech, J. J., & Plotkin, K. J. (1998). NMAP 7.0 User's Manual (No. WR-98-13). WYLE RESEARCH LAB ARLINGTON VA.

[19] Barnes, J., & Crowley, P. (2013, October). k-p0f: a high-throughput kernel passive os fingerprinter. In Architectures for Networking and Communications Systems (ANCS), 2013 ACM/IEEE Symposium on (pp. 113-114). IEEE.

[20] Sheffer, Y., Holz, R., & Saint-Andre, P. (2015). Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS) (No. RFC 7457).

[21] Chauhan, P., Jhummarwala, A., & Pandya, M. Detection of DDoS Attack in Semantic Web.