Contents

List of Figures

List of Tables

Listings

1 Introduction of Tools

In this section, i will be introducing the tools ,languages and API's which will be used throughout the report.

1.1 ELK Stack

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana.

1.1.1 Elastic Search

Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

1.1.2 Logstash

Logstash is an open source tool for collecting, parsing, and storing logs for future use.

1.1.3 Kibana

Kibana is an open source data visualization plugin for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.

1.2 Honeypots

In computer terminology, a honeypot is a trap set to detect, deflect or in some manner counteract attempts at unauthorized use of information systems.

In other words, A server that is configured to detect an intruder by mirroring a real production system. It appears as an ordinary server doing work, but all the data and transactions are phony. Located either in or outside the firewall, these are used to learn about an intruder's techniques as well as determine vulnerabilities in the real system."

Based on design criteria, honeypots can be classified as

1. Low-interaction honeypots
2. Medium-interaction honeypots
3. High-interaction honeypots

1.2.1 Low-interaction honeypots

Low-interaction honeypots simulate only the services frequently requested by attackers. Since they consume relatively few resources, multiple virtual machines can easily be hosted on one physical system, the virtual systems have a short response time, and less code is required, reducing the complexity of the security of the virtual systems.

Low-interaction honeypots present the hacker emulated services with a limited subset of the functionality they would expect from a server, with the intent of detecting sources of unauthorized activity. For example, the HTTP service on low-interaction honeypots would only support the commands needed to identify that a known exploit is being attempted.

1.2.2 Medium-interaction honeypots

Medium-interaction honeypots might more fully implement the HTTP protocol to emulate a well-known vendor's implementation, such as Apache. However, there are no implementations of a medium-interaction honeypots and for the purposes of this paper, the definition of low-interaction honeypots captures the functionality of medium-interaction honeypots in that they only provide partial implementation of services and do not allow typical, full interaction with the system as high-interaction honeypots.

1.2.3 High-interaction honeypots

High-interaction honeypots imitate the activities of the real systems that host a variety of services. It let the hacker interact with the system as they would any regular operating system, with the goal of capturing the maximum amount of information on the attacker's techniques. Any command or application an end-user would expect to be installed is available and generally, there is little to no restriction placed on what the hacker can do once he/she comprises the system.

According to recent researches in high interaction honeypot technology, by employing virtual machines, multiple honeypots can be hosted on a single physical machine. Therefore, even if the honeypot is compromised, it can be restored more quickly. Although high interaction honeypots provide more security by being difficult to detect, but it has the main drawback that it is costly to maintain.

1.2.4 Cowrie Honeypot

Cowrie is used for our research purposes.Cowrie is a medium interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker.

Interesting Features of Cowrie Honeypot

1. Fake filesystem with the ability to add/remove files. A full fake filesystem resembling a Debian 5.0 installation is included.
2. Possibility of adding fake file contents so the attacker can cat files such as /etc/passwd. Only minimal file contents are included.
3. Logging in JSON format for easy processing in log management solutions.

1.3 VirusTotal

VirusTotal is a website created by the Spanish security company Hispasec Sistemas. Launched in June 2004, it was acquired by Google Inc. in September 2012. The company's ownership switched in January 2018 to Chronicle, a subsidiary of Alphabet Inc. (Google's parent company).

VirusTotal aggregates many antivirus products and online scan engines to check for viruses that the user's own antivirus may have missed, or to verify against any false positives.Files up to 256 MB can be uploaded to the website or sent via email.Anti-virus software vendors can receive copies of files that were flagged by other scans but passed by their own engine, to help improve their software and, by extension, VirusTotal's own capability.

1.3.1 List of Anti-Virus Engines used by VirusTotal

AegisLab	Agnitum	AhnLab	Anity
Aladdin	Avast	AVG	Avira
BluePex	Baidu	BitDefender	Bkav
ByteHero	Quick Heal	CMC Antivirus	CYREN
ClamAV	Comodo	CrowdStrike	Doctor Web Ltd.
Emsisoft	Endgame	Eset Software	Fortinet
F-Prot	F-Secure	G Data	Hacksoft
Hauri	IKARUS	nProtect	Invincea
Jiangmin	K7AntiVirus	Kaspersky	Kingsoft
Malwarebytes	McAfee	Microsoft	eScan
Nano Security	Norman	Panda	Rising
Symantec	VIPRE	TotalDefense	TrendMicro

Table 1: List of Antivirus Engines

1.4 Github

GitHub (originally known as Logical Awesome LLC) is a web-based hosting service for version control using git. It is mostly used for computer code. It

offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

The code which is written for this project is entirely pushed to Github where it is bug free and version safe.

2 Installation of Tools

In this section, we will be mentioning about the installation of various tools which will be used in the project.

2.1 Elastic Search

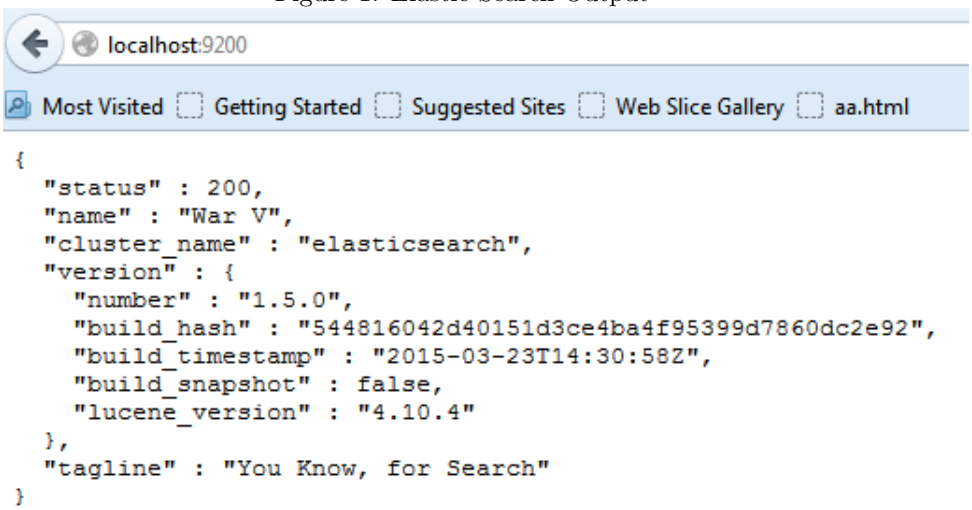
1. Download the Debian Installation file from the below link

2. Install it using the below command
   iotsys3@iotsys3-Precision-Tower-3420: dpkg -i elastic-search-6.2.0.deb

3. Now start the service using below command
   iotsys3@iotsys3-Precision-Tower-3420: sudo service elasticsearch start

Open the link *http://localhost:9200* in the browser and check for the ouput which should look like below.

Figure 1: Elastic Search Output



We can check for the working of elastic search in the browser by typing the url *http://localhost:9200* and verifying the output.

2.2 Logstash

1. Download the logstash debian installation file from below link
   https://www.elastic.co/downloads/logstash

2. Install it using the below command
   iotsys3@iotsys3-Precision-Tower-3420: *dpkg -i logstash-6.2.2.deb*

3. Now start the service using below command
   iotsys3@iotsys3-Precision-Tower-3420: sudo service logstash start

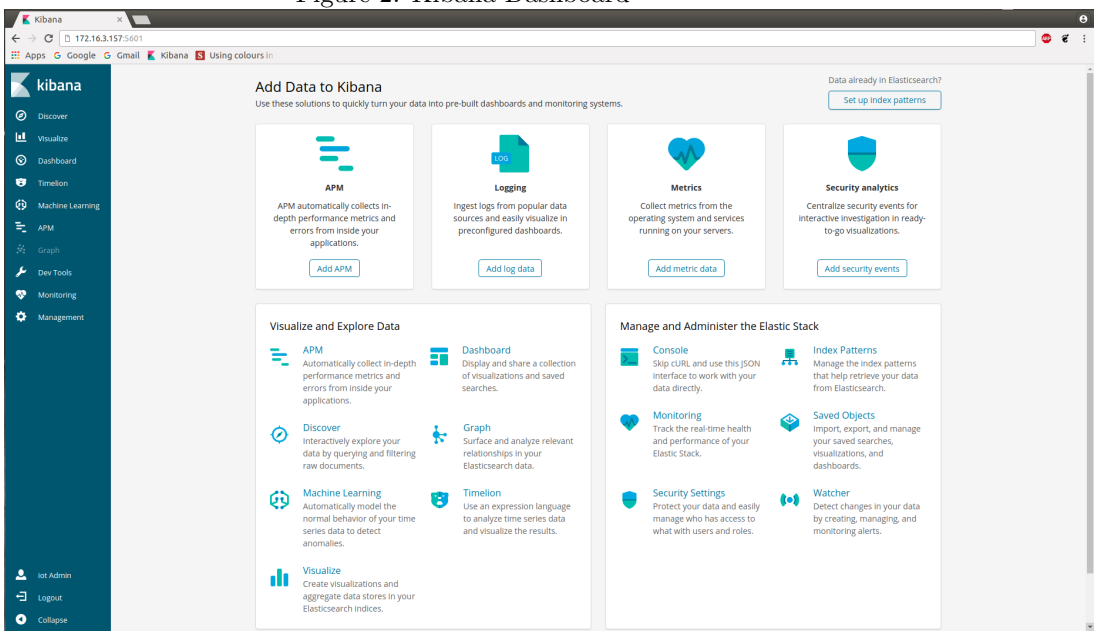
2.3 Kibana

1. Download the Kibana debian installation file from below link
   https://www.elastic.co/downloads/kibana

2. Install it using the below command
   iotsys3@iotsys3-Precision-Tower-3420: *dpkg -i kibana-6.2.2.deb*

3. Now start the service using below command
   iotsys3@iotsys3-Precision-Tower-3420: *sudo service kibana start*

Figure 2: Kibana Dashboard



Open the link (*http://localhost:5601*) in the browser and check for the Kibana Dashboard which looks like above.

2.4 Cowrie Honeypot

Cowrie honeypot is installed in a separate machine with public IP which is connected to the internet.It can be installed by following the steps given below.

2.4.1 Install dependencies

iotsys3@iotsys3-Precision-Tower-3420: sudo apt-get install git python-virtualenv libssl-dev libffi-dev build-essential libpython-dev python2.7-minimal authbind

2.4.2 Create a user account

iotsys3@iotsys3-Precision-Tower-3420: sudo adduser –disabled-password cowrie
iotsys3@iotsys3-Precision-Tower-3420: sudo su - cowrie

2.4.3 Checkout the code

iotsys3@iotsys3-Precision-Tower-3420: git clone http://github.com/micheloosterhof/cowrie
iotsys3@iotsys3-Precision-Tower-3420: cd cowrie

2.4.4 Setup Virtual Environment

iotsys3@iotsys3-Precision-Tower-3420: pwd
iotsys3@iotsys3-Precision-Tower-3420: virtualenv cowrie-env

2.4.5 Port redirection

iotsys3@iotsys3-Precision-Tower-3420: sudo iptables -t nat -A PREROUTING -p tcp –dport 22 -j REDIRECT –to-port 2222

In the same folder, create a copy of *cowrie.cfg.dist* and modify the file according to your needs like changing the name, SSH port number etc.

2.4.6 Starting Cowrie

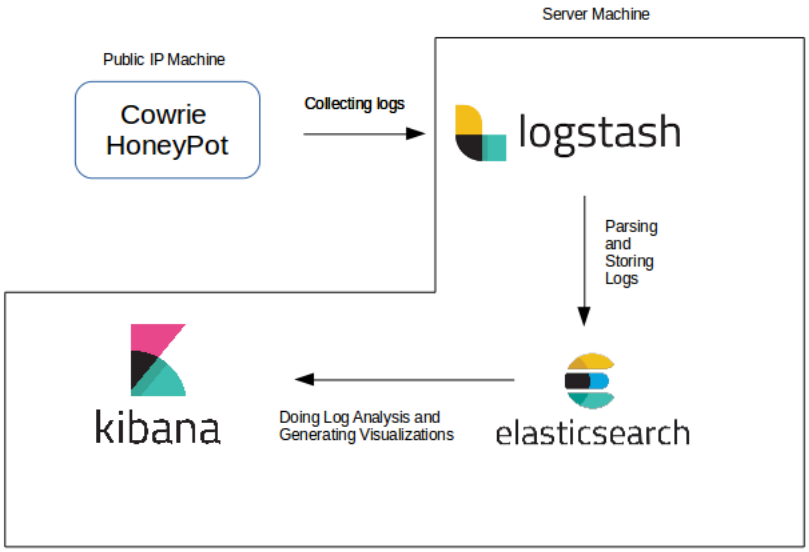
iotsys3@iotsys3-Precision-Tower-3420: bin/cowrie start

Cowrie will be up and running on port 22 which will monitor all SSH logins and commands executed by the attackers or intruders.

3 Working Model

All the tools listed above are integrated to form a complete monitoring solution which will looks like below.

Figure 3: Monitoring System



Cowrie will be generating the logs in the form of JSON format which will be available in public IP machine as specified in the figure-3. In order to stream the logs from the public IP machine to Server machine, filebeat can be used which is an open source tool. Unfortunately we did not get to use that tool as there was no physical connection between public IP machine and server machine due to security reasons.

So all the logs are copied from public IP machine to server machine on daily basis. These log files are supplied as input to Logstash which will parse, add metadata and push them to Elastic Search database. Logstash works on config files. Each config file has three parameters namely input, filter and output. Our config file is given below.

```
input {
  file {
    path => "/home/iotsys3/Documents/Cowrie_Logs/*"
    start_position => "beginning"
    ignore_older => 0
  }
}
filter {
```

```
9
10    json {
11              source => message
12          }
13
14          date {
15              match => [ "timestamp", "ISO8601" ]
16          }
17
18          if [src_ip]  {
19
20              dns {
21                  reverse => [ "src_host", "src_ip" ]
22                  action => "append"
23              }
24
25              geoip {
26                  source => "src_ip"  # With the src_ip field
27                  target => "geoip"   # Add the geoip one
28                  # Using the database we previously saved
29                  database => "/home/iotsys3/Downloads/GeoLite2-
      City_20180206/GeoLite2-City.mmdb"
30                  add_field => [ "[geoip][coordinates]", "%{[geoip][
      longitude]}" ]
31                  add_field => [ "[geoip][coordinates]", "%{[geoip][
      latitude]}"  ]
32              }
33
34              # Get the ASN code as well
35              geoip {
36                  source => "src_ip"
37                  database => "/home/iotsys3/Downloads/GeoLite2-
      ASN_20180206/GeoLite2-ASN.mmdb"
38              }
39
40              mutate {
41                  convert => [ "[geoip][coordinates]", "float" ]
42              }
43          }
44
45  }
46  output {
47
48    elasticsearch {
49              hosts => ["localhost:9200"]
50              sniffing => true
51              manage_template => false
52              index => "logstash-cowrie-%{+YYYY.MM.dd}"
53            user => "elastic"
54              password => "63TxZjWQNkB4tArR7hri"
55          }
56
57    stdout {
58              codec => rubydebug
59          }
```

```
60 }
```

Listing 1: Cowrie Config File

1. Input

   It contains a parameter file which has options path, start-position and ignore-older. Path is where log files are present and start-position indicates from which point Logstash should start reading the logs from.

2. Filter

   Filter is where entire processing happens in logstash. It has many options out of which json option is used as the logs are in JSON format. Some more information is added to the already present logs like country , Continent , latitude and longitude information which are specified in geoip option. Finally latitude and longitude are converted into float variables for plotting on map.

3. Output

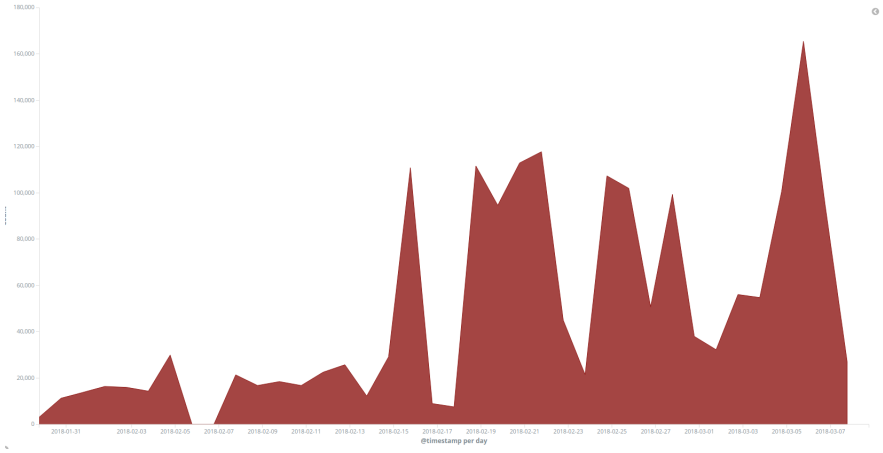
   It specifies options as to where the data should go after the processing is done. There are many options out of which we are using elastc search where data is stored in json format which will be in key value format.

Data collected in the elastic search will be picked up by the Kibana by default as specified in the Kibana config files. Using the data visualizations are created for analyzing and monitoring the regular changes in data. Monitoring system was deployed for 40 days and collected nearly **2 Million Records**.

Using Kibana different visualizations are created which are listed below.

1. Daily Attacks Count Area Graph An area graph was plotted against the count of attacks on daily basis. X-axis represents the date and Y-axis represents the count of attacks.It can observed that, attacks kept on increasing at the end.

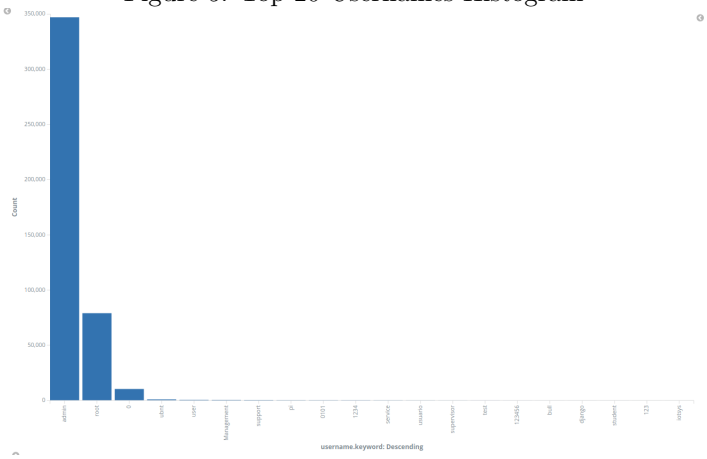
Figure 4: Daily Attacks Count Area Graph



2. Top 20 Usernames Histogram

A Histogram was used to represent the top 20 usernames that have been used by the attackers and intruders.

Figure 5: Top 20 Usernames Histogram



From the graph, it can be observed that username "admin", "root and "ubnt" are mostly used by attackers.

3. Unique Count Of IP's

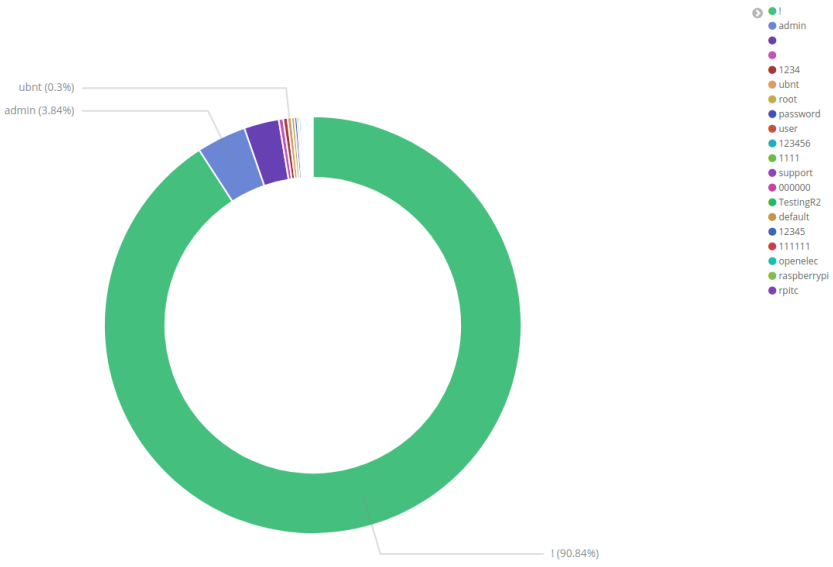
A Kibana Utility graph named Unique Count has been used for counting the Unique Number of IP's till now monitoring system has captured. It has found to be nearly 2000 IP's.

Figure 6: Unique Count of IP's

4. Top 20 Passwords Pie

A pie chart has been drawn against the top 20 passwords used by the attackers and intruders. It can be observed that passwords "ubnt" and "admin" are mostly used.

Figure 7: Top 20 Passwords Pie



5.

Algorithm 1 Modified Naive Bayes With K-Means Clustering

Probability(*IP is Malicious / Commands executed by IP*) =

$$\frac{\textbf{Probability}(\textit{IP is Malicious}) * \textbf{Probability}(\textit{Commands executed by IP / IP is Malicious})}{\textbf{Probability}(\textit{Commands executed by IP})}$$

Note :*As we are working on Unclassified Data,We are assuming below values.*

Probability(*IP is Malicious*) = **1**

Probability(*Commands executed by IP*) = **1**

Probability(*Commands executed by IP / IP is Malicious*) =

$$\frac{\textit{Frequency of Word + 1 (Smoothening Factor)}}{\textit{Total Number of Words + Unique Number of Words}}$$

Note : *Commands executed by each IP is split into words and their frequencies are stored in Database*

Total No of Words $\leftarrow N$

Unique No of Words $\leftarrow dict$

while *IP List* **not** *Empty* **do**

  No of Words $\leftarrow$ *Splitting Commands into words*

  Probability(*IP is Malicious / Commands executed by IP*)

  = **Probability** (*Commands executed by IP / IP is Malicious*)

  = $\prod_{i=1}^{No\ of\ Words}$ **Probability**(*Word*$_i$ *in Command Executed/ IP is Malicious*)

  = $\prod_{i=1}^{No\ of\ Words} \frac{n_i + 1}{N + dict}$

done

Note : *Each IP Final Probability will be stored in Database*

Note : *After Storing the IP's Probabilities in Database,***K-Means** *Clustering algorithm is applied to classify the IP's into two classes as high and Low.*

****

K-Means is given below

****

Input : *Set of Data Points* **D** *and No. of Clusters* **K.**

Output : *Cluster Centers that minimizes the squared error distortion.*

Algorithm :

1. *Pick K Data points randomly from D to form cluster centers.*
2. *Assign each data point to its nearest cluster center by calculating and taking the minimum Euclidean Squared distance metric.*
3. *After all data points are assigned to their clusters,move each cluster center to mean of its assigned data points.*
4. *With new cluster centers, repeat 2 to 3 until there is no convergence.*