Industrial Training Project

Report

at

**Weapons Electronics**

**Systems**

**Engineering Establishment**

**(WESEE)**

Presented by:

**Rohan Anand**

**SRM University (KTR)**

**RA1411008010252**

**Acknowledgement**

I would like to thank WESEE, New Delhi, for providing me this brilliant opportunity to undertake my Winter Internship for a period of 3 weeks as an Undergraduate Student looking to gain insight on how things work in the real industry.

I would also like to take this opportunity to express my deep gratitude towards the WESEE Network Security Lab (NSL) managers and employees for their exemplary support, guidance and constant encouragement through the course of this project. They welcomed me with open arms to their group and quickly integrated me into the working office. Without their support, I would not have gained as much knowledge as I have now and I would not enjoy working in the office as much as I did.

I am extremely grateful for their cooperation during the period of my assignment. Despite their busy schedules, they spared their time to impart useful knowledge related to how to work in an office, as well as new technologies.

I also thoroughly enjoyed their daily talks in which once a day every employee would come and share new about new technologies and their uses in the industry.

**About WESEE**

WESEE (Weapons Electronics Systems Engineering Establishment) is an Indian Defence Service Organization that researches and develops new technologies to improve the functioning of the Indian Defence Services by applying new technologies to valid fields of applications.

The fields in which WESEE is involved in include, but are not limited to are :-

* Submarine Warfare
* Surface Warfare
* Surface to Air Warfare
* Secure Systems Communication
* Network Security
* Cryptanalysis

… and many more

This organization is dedicated to ensure that the Indian Defence Services are not in any compromising situation due to lack of technological solutions and ensures that India does not lag behind in defence technologies with the rest of the world.

WESEE was also recently the first defence service organisation to achieve CMMI Maturity Level 3 Rating for software development and maintenance projects. This has demonstrated their ability to deliver world class systems complying with international standards.

**Centralized Log**

**Collection**

**and**

**Analysis**

**Problem Statement**

Now a days, many organizations run multiple servers at the same time. These servers could be used for multiple purposes such as :-

* Web Server
* Network Filter Server
* Antivirus Server
* FTP Server

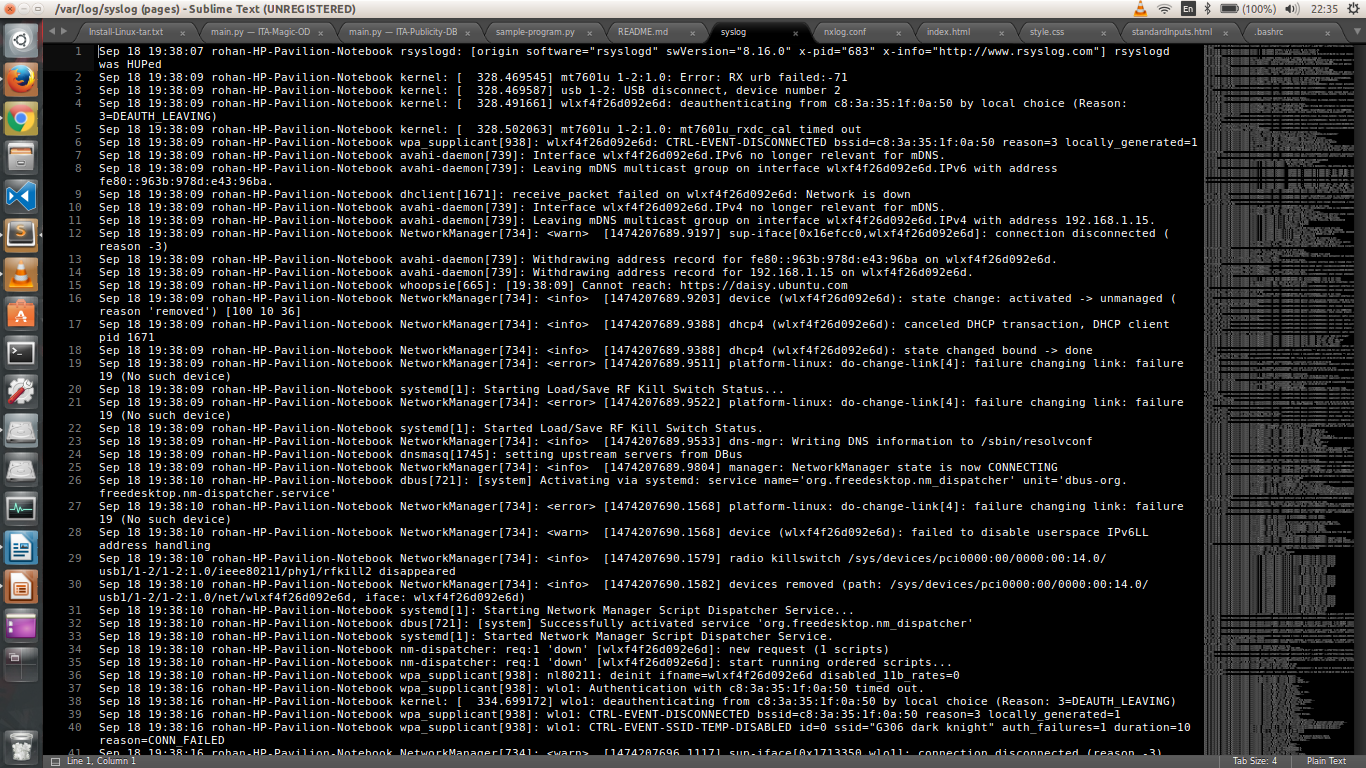
As these servers are being used for different applications, usually they will be using different Operating Systems to run their services as in certain operating systems, it is easier to set up some services over others.

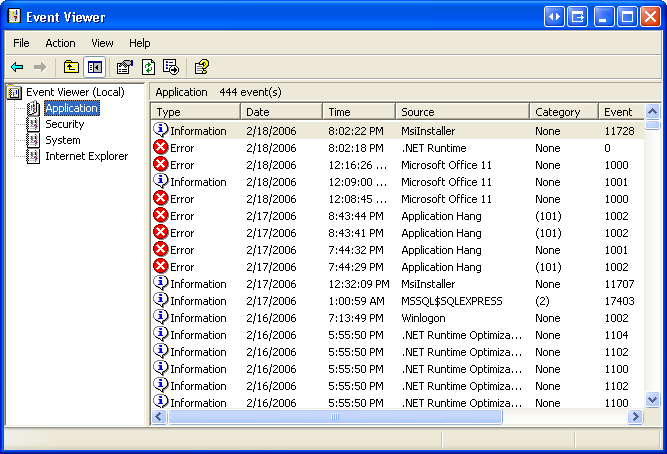
For example, It is easier to implement an antivirus server in Windows Server as most products offer better services and compatibility for Windows when compared to Linux. However, Ubuntu or other Linux Distros will be used for Web and FTP applications as it is easier to customize them in Linux for the organization's own requirements.

This means that an organization will have multiple servers running both Linux and Windows on them.

Now each machine generates its own logs. However all of these logs are available in different formats. For example, most linux logs are stored in a syslog text format, while windows uses its own proprietary .evtx format which can only be viewed natively by the Windows Event Manager.

Given below is an example of a syslog and an evtx file

*Syslog File Example*



*Windows Event Viewer*

As we can see from this, these two are very different formats. Also, the syslogs are in a format that is not easily read by a user. And while the windows event viewer gives data in a much more presentable format, it is impossible for one to analyze all of those logs.

Also since these logs are from separate systems, you would have to physically or remotely login to each individual terminal separately which is tedious and time consuming. Also there is no mechanism to search or visualize any of this data in a meaningful way.

This is a problem in an industry where one would like to be able to view all these logs simultaneously and visualize this data so as to make meaningful decisions based on the data given.

**The Solution**

The above problem can be broken down into 3 parts

* How to view all of these logs together
* How to read and format these logs of different formats
* How to analyze and visualize this data in a meaningful way

The first problem can be solved by centralizing all of the logs. Each server can send their logs to one centralized log server for collection and analysis. This means that all of the logs are now located in one machine.

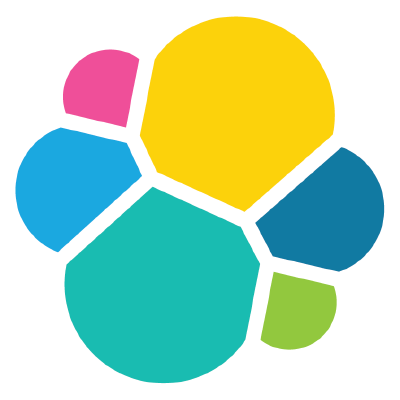
The second problem can be solved by parsing the logs using separate parsing rules for each specific file type. This can be done by using parsing tools.

The third problem requires that the data is indexed first by a powerful engine for storing and sorting these logs. Only then can the data be visualized by a UI designed for this purpose.

Now the problem comes to how to design these tools. Luckily the Open Source community has already built most of these tools and they can now be customized and stacked together to form our own custom Centralized Log Server stack. So, we drastically reduce the time of this project by not building from scratch and improvising already existing technologies for our purposes. In the next section we shall talk about these technologies.

**Technologies Used**

To tackle the problem, we use 4 open source technologies and stack them together for our needs :



**Elasticsearch :** This is the engine which will be used to store and index the logs we receive from logstash.



**Logstash :** The server component of our stack that processes incoming logs from a log forwarder such as Nxlog or Filebeat. For our stack we used Nxlog.



**Kibana :** Web interface for searching and visualizing logs, which will be proxied through Nginx or any other web server.

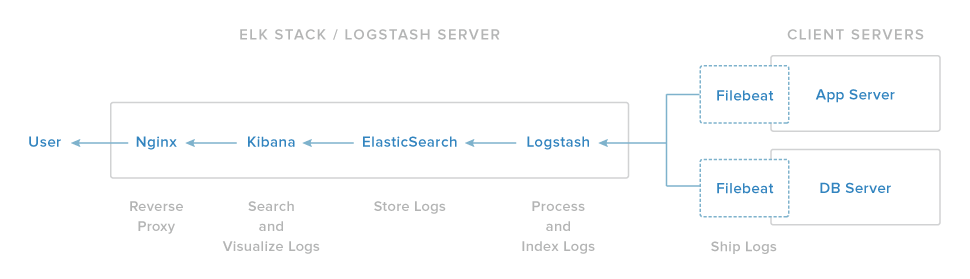


**Nxlog :** Universal log collector and forwarder supporting different platforms log sources and protocols.

Using all of these together we form the ELK Stack (**E**lasticsearch **L**ogstash **K**ibana)

**Implementation**

We will install the first three components on a single server, which we will refer to as our ELK Server. Nxlog will be installed on all of the client servers that we want to gather logs for, which we will refer to collectively as our Client Servers.



In our setup, the ELK server components are installed on an mobile workstation running Ubuntu 14.04 LTS. Since Elasticsearch and Logstash use the Java Virtual Machine (JVM) this systems also had JDK8 configured.

We now have to individually install and configure each service onto the ELK Server.

**Elasticsearch**

The configuration file for elasticsearch once installed is located in

/etc/elasticsearch/elasticsearch.yml

Over here you can change your network host id which you wish to use as well as the ports you would like to use. Given below is a part of the sample configuration.

network.host: localhost

transport.tcp.port: 9300

transport.tcp.compress: true

http.port: 9200

http.max\_content\_length: 100mb

http.enabled: false

**Nginx**

Nginx is the web server we were using. This web server must be configured to fetch the elasticsearch results. The configuration file is located at

/etc/nginx/sites-available/default

Here is a sample configuration

server {

listen \*:80 ;

server\_name localhost;

root /var/www/kibana3;

access\_log /var/log/nginx/kibana.myhost.org.access.log;

location / {

root /var/www/kibana3;

index index.html index.htm;

}

location ~ ^/\_aliases$ {

proxy\_pass http://127.0.0.1:9200;

proxy\_read\_timeout 90;

}

location ~ ^/.\*/\_aliases$ {

proxy\_pass http://127.0.0.1:9200;

proxy\_read\_timeout 90;

}

location ~ ^/\_nodes$ {

proxy\_pass http://127.0.0.1:9200;

proxy\_read\_timeout 90;

}

location ~ ^/.\*/\_search$ {

proxy\_pass http://127.0.0.1:9200;

proxy\_read\_timeout 90;

}

location ~ ^/.\*/\_mapping {

proxy\_pass http://127.0.0.1:9200;

proxy\_read\_timeout 90;

}

# Password protected end points

location ~ ^/kibana-int/dashboard/.\*$ {

proxy\_pass http://127.0.0.1:9200;

proxy\_read\_timeout 90;

limit\_except GET {

proxy\_pass http://127.0.0.1:9200;

auth\_basic "Restricted";

auth\_basic\_user\_file /etc/nginx/conf.d/kibana.myhost.org.htpasswd;

}

}

location ~ ^/kibana-int/temp.\*$ {

proxy\_pass http://127.0.0.1:9200;

proxy\_read\_timeout 90;

limit\_except GET {

proxy\_pass http://127.0.0.1:9200;

auth\_basic "Restricted";

auth\_basic\_user\_file /etc/nginx/conf.d/kibana.myhost.org.htpasswd;

}

}

}

We have configured Nginx to get the results from elasticsearch engine onto our kibana UI which will be hosted on the web server.

**Logstash**

We now proceed to install Logstash. Logstash configuration files are in the JSON-format, and reside in /etc/logstash/conf.d. The configuration consists of three sections: inputs, filters, and outputs.

Here is a sample configuration file for syslog parsing as well as windows event logs parsing

input {

file {

path => [ "/var/log/\*.log", "/var/log/messages", "/var/log/syslog" ]

type => "syslog"

}

tcp {

type => "WindowsEventLog"

port => 3515

codec => "line"

}

}

filter {

if [type] == "syslog" {

grok {

match => { "message" => "%{SYSLOGTIMESTAMP:syslog\_timestamp} %{SYSLOGHOST:syslog\_hostname} %{DATA:syslog\_program}(?:\[%{POSINT:syslog\_pid}\])?: %{GREEDYDATA:syslog\_message}" }

add\_field => [ "received\_at", "%{@timestamp}" ]

add\_field => [ "received\_from", "%{host}" ]

}

syslog\_pri { }

date {

match => [ "syslog\_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]

}

}

if [type] == "WindowsEventLog" {

json {

source => "message"

}

if [SourceModuleName] == "eventlog" {

mutate {

replace => [ "message", "%{Message}" ]

}

mutate {

remove\_field => [ "Message" ]

}

mutate {

lowercase => [ "EventType", "FileName", "Hostname", "Severity" ]

}

mutate {

rename => [ "Hostname", "source\_host" ]

}

mutate {

gsub => ["source\_host","\.example\.com",""]

}

date {

match => [ "EventTime", "YYYY-MM-dd HH:mm:ss" ]

}

mutate {

rename => [ "Severity", "eventlog\_severity" ]

rename => [ "SeverityValue", "eventlog\_severity\_code" ]

rename => [ "Channel", "eventlog\_channel" ]

rename => [ "SourceName", "eventlog\_program" ]

rename => [ "SourceModuleName", "nxlog\_input" ]

rename => [ "Category", "eventlog\_category" ]

rename => [ "EventID", "eventlog\_id" ]

rename => [ "RecordNumber", "eventlog\_record\_number" ]

rename => [ "ProcessID", "eventlog\_pid" ]

}

if [SubjectUserName] =~ "." {

mutate {

replace => [ "AccountName", "%{SubjectUserName}" ]

}

}

if [TargetUserName] =~ "." {

mutate {

replace => [ "AccountName", "%{TargetUserName}" ]

}

}

if [FileName] =~ "." {

mutate {

replace => [ "eventlog\_channel", "%{FileName}" ]

}

}

mutate {

lowercase => [ "AccountName", "eventlog\_channel" ]

}

mutate {

remove => [ "SourceModuleType", "EventTimeWritten", "EventReceivedTime", "EventType" ]

}

}

}

}

output {

elasticsearch { host => localhost }

stdout { codec => rubydebug }

}

This is one of the most important files in our system. This is responsible for parsing the logs to the correct format we require.

**Nxlog**

Nxlog is our log forwarder which shall forward all the logs to the desired server. This should be configured in each of the client server and should point to our centralized log server.

Given is a sample configuration for windows.

## Please set the ROOT to the folder your nxlog was installed into,

## otherwise it will not start.

#define ROOT C:\Program Files\nxlog

define ROOT C:\Program Files (x86)\nxlog

Moduledir %ROOT%\modules

CacheDir %ROOT%\data

Pidfile %ROOT%\data\nxlog.pid

SpoolDir %ROOT%\data

LogFile %ROOT%\data\nxlog.log

<Extension json>

Module xm\_json

</Extension>

# Nxlog internal logs

<Input internal>

Module im\_internal

Exec $EventReceivedTime = integer($EventReceivedTime) / 1000000; to\_json();

</Input>

# Windows Event Log

<Input eventlog>

# Uncomment im\_msvistalog for Windows Vista/2008 and later

Module im\_msvistalog

# Uncomment im\_mseventlog for Windows XP/2000/2003

# Module im\_mseventlog

Exec $EventReceivedTime = integer($EventReceivedTime) / 1000000; to\_json();

</Input>

<Output out>

Module om\_tcp

Host 192.168.1.126

Port 3515

</Output>

<Route 1>

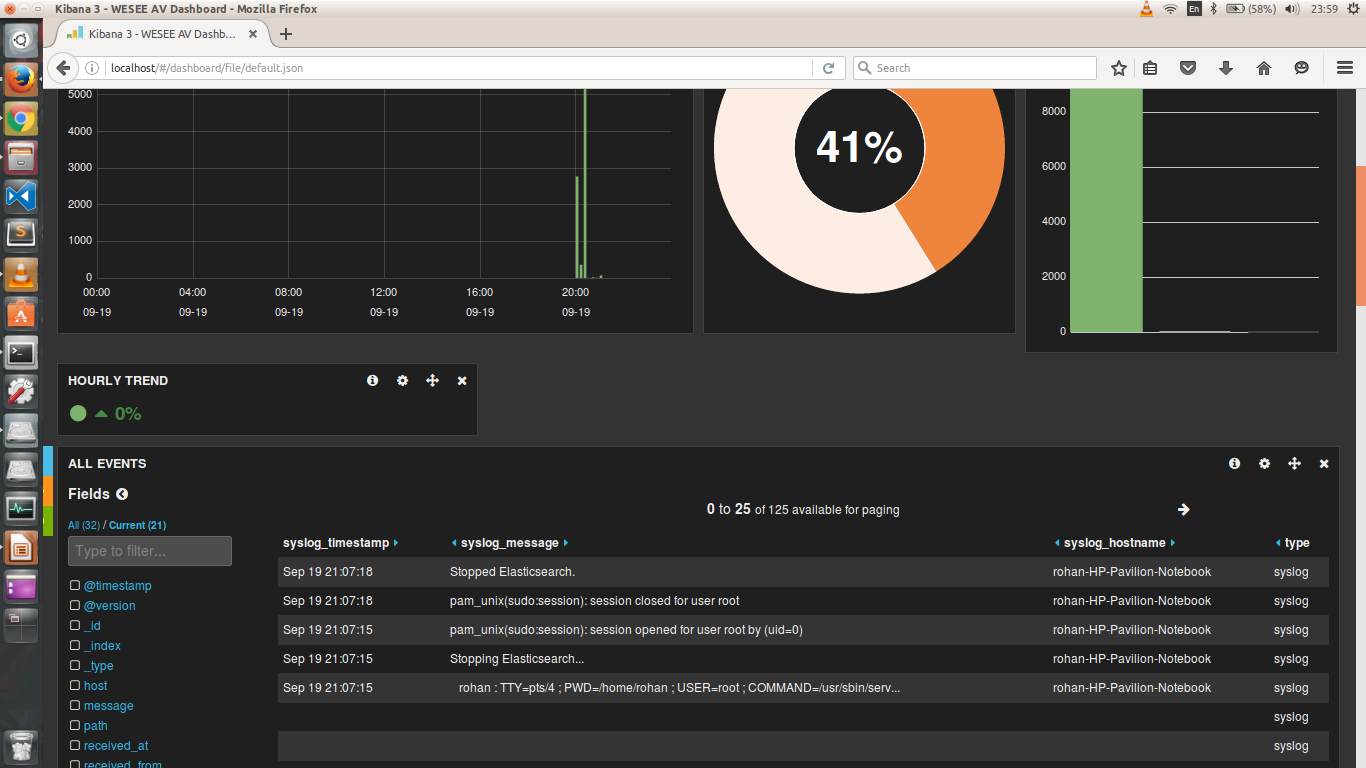
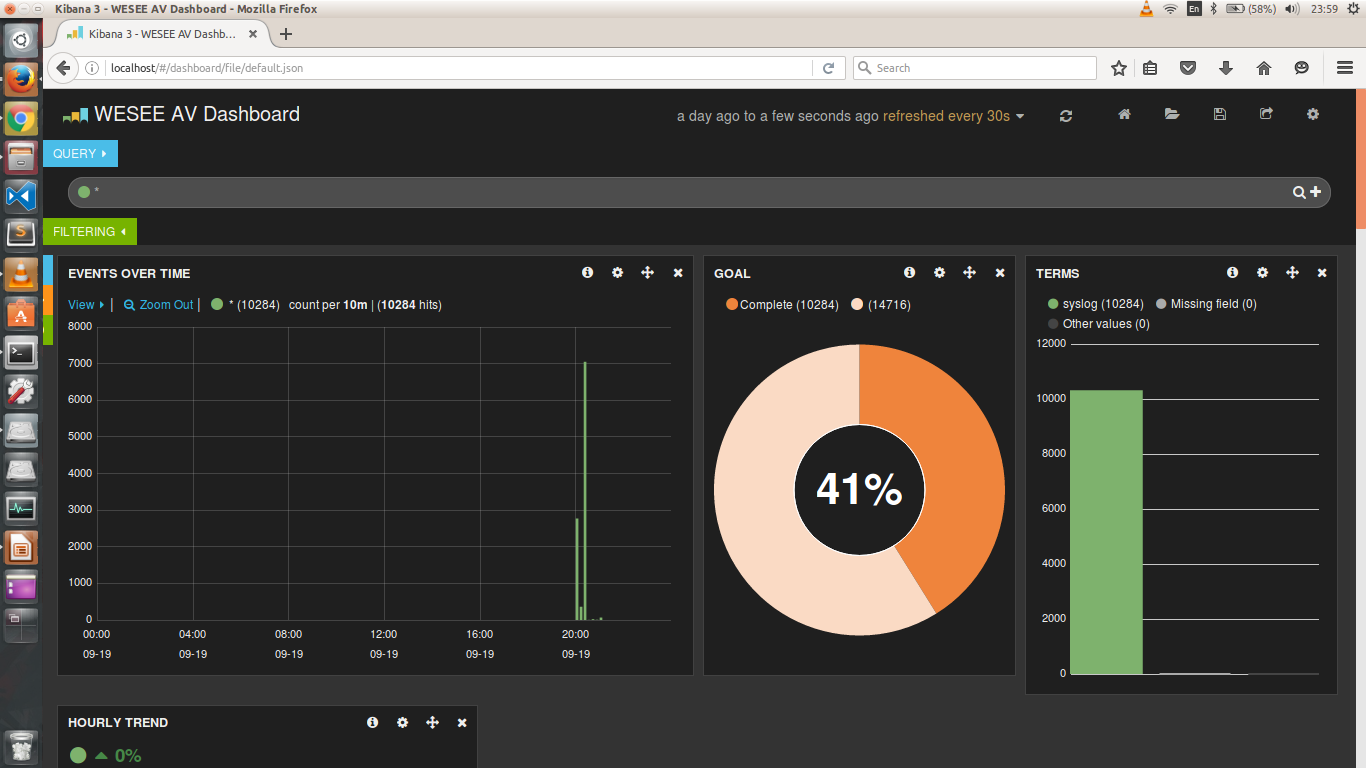
Path internal, eventlog => out

</Route>

Note that this configuration must be setup to point to the Central Server and must be installed in each Client Server. This is the log forwarder.

**Conclusion**

Now we have our ELK server up and running we should be able to visualize logs on our web server. We can customize kibana to give us different views of our system. Elasticsearch allows you to search these logs and isolate the ones that you wish to view.



￼