Design Spec for ELK VM and Beats

The original ELK vm at 192.168.0.91 developed maintenance/configuration issues, most of which are admittedly due to lack of knowledge about its configuration. Updating to the latest application versions and going through the steps of configuring a new ELK server (and beats, for shipping data) will be a beneficial practice--especially if we document its creation/configuration, as this document attempts to do.

VM setup:

|  |  |
| --- | --- |
| Kronos | |
| IP | 192.168.0.92 |
| User | elk-root |
| Password | Blackout123 |
| Interfaces | |
| ens160 | 192.168.0.94 |
| ens192 | Listens to mirror port of switch at 192.168.0.87 |

The new vm, Kronos, has all required components/configs installed as services: elasticsearch, logstash, kibana, softflowd, nginx. As a result, all services should come up on reboot within about a minute (logstash can take ~30s). Debuggin can be done with the systemctl status command, e.g. ‘*sudo systemctl status logstash’*. Other service configuration can be done with systemctl as well.

The vm receives these primary types of data:

* Netflows: These are captured from the Ethernet switch (192.168.0.87) mirror port, port 48
* Sysmon data: sysmon event data
* Winlog: Windows event data, for windows hosts on our system
* Snort: IDS data
* Bro: Bro IDS data

[Network diagram]

Nginx

Nginx proxies to kibana and elasticsearch using the following configuration:

1. sudo apt install nginx
2. echo "admin:$(openssl passwd -apr1 YourStrongPassword)" | sudo tee -a /etc/nginx/htpasswd.kibana
3. sudo rm /etc/nginx/sites-enabled/default
4. sudo nano /etc/nginx/sites-available/kibana, and add the following code:



1. sudo ln -s /etc/nginx/sites-available/kibana /etc/nginx/sites-enabled/kibana
2. sudo nginx -t

sudo service nginx restart

Nginx creds are:

1. admin
2. password=Blackout123

Netflow Configuration

Softflowd receives flow data via the switch mirror port. The switch is at 192.168.0.87, but note that this is the ip of the switch itself; the mirror port is a link-layer port. Softflowd is configured as a service, which just manually calls the following command on boot to generate netflow-v9 data and send it to logstash: ‘usr/sbin/softflowd –v 9 –d –i eth1 –n 192.168.0.94:9999’.

[1]

Data string example for logstash-output.conf file. Note the date format string in the @index field:

*output {*

*elasticsearch {*

*host => localhost*

*cluster => "elasticsearch\_prod"*

*index => "test-%{+YYYY.MM.dd}"*

*}*

*}*

Elasticsearch: localhost:9200

Kibana: localhost:5601

Logstash:

Netflow module listening at localhost:9999

Logstash setup:

1. Install logstash from deb

Winlogbeat setup:

Winlogbeat must be setup for each windows host of interest, and once installed, ships data directly to elasticsearch. This differs from most ELK patterns, where a beat ships data to a logstash ETL object, which then transforms and loads the data into elasticsearch.

**TODO:** In the future it might be desirable to configure winlogbeat to ship data to logstash, using the traditional *beat -> logstash -> elasticsearch* pattern of loading data into elasticsearch. But I actually don’t know of any benefits, except the flexibility of logstash for potentially managing the flow of data. For example, shipping sysmon data (via winlogbeat) has been observed to stall the elasticsearch server by overloading it with data. This is a problem with sysmon (more filtering rules need to be implemented), not elasticsearch, but bringing down the elasticsearch indices for all clients ain’t good.

Logstash:

Installation: installed on Ubuntu 16.04 with ‘sudo apt install logstash’. The state of the logstash process can observed using `sudo systemctl status logstash’, and the server has it set up to run as a service, so it should start at boot.

There are two ways to run logstash **[IMPORTANT]**:

1. Manually, passing a pointer to the directory containing the beats configs:‘/usr/share/logstash/bin/logstash –f /etc/logstash/conf.d’
2. As a service: The documentation on running logstash as a service is poor, given that elastic and kibana run as services without modification. With logstash, I had to change the group of all files/sub-folders in /etc/logstash:
   1. ‘sudo chown :logstash /etc/logstash/\*’
   2. ‘sudo chown :logstash /etc/logstash/conf.d/\*’

Then logstash magically works as a service. The problem is that installing from the apt repositories installs all the configs in /etc/logstash as root, but in order to see/access them (?) they must belong to the ‘logstash’ group. The documentation is all over the place on this issue, because it has changed depending on Ubuntu version and logstash version and directory layout.