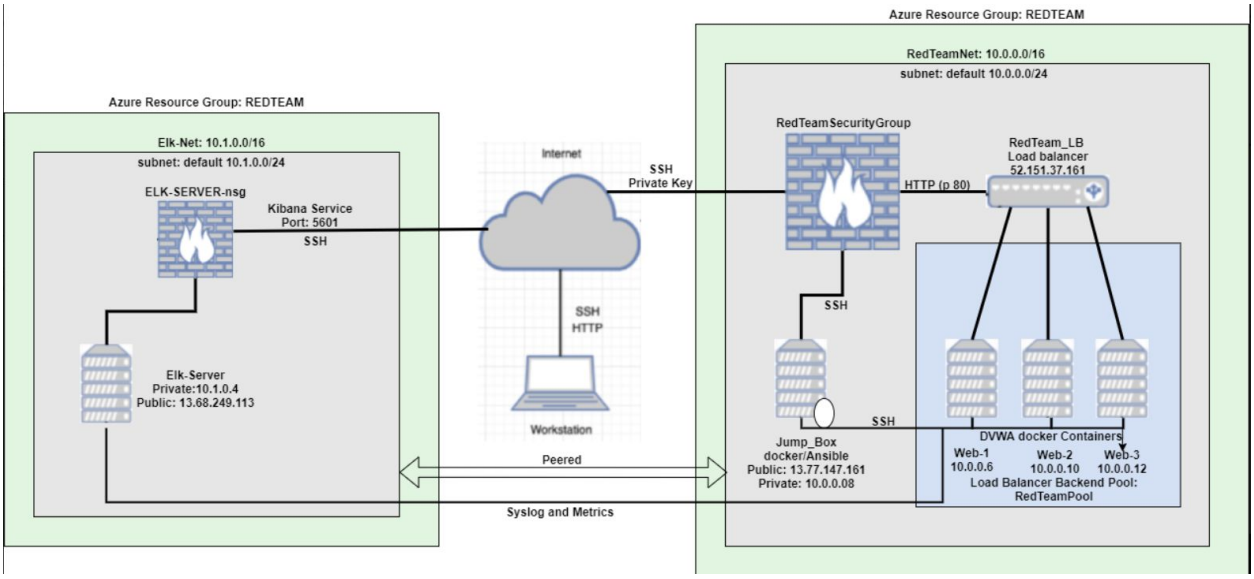
Automated ELK Stack Deployment

Purpose of this document:

This document contains a description of intended topology and the process for establishing an ELK server to monitor and provide telemetry for the web servers already established.

The files in this repository were used to configure the network depicted below.

[RedTeam Network Diagram]



The files included in this repository have been tested and used to generate a live ELK deployment on Azure. They can be used to either recreate the entire deployment pictured above. Alternatively, select portions of the ansible playbook files may be used to install only certain pieces of it, such as Filebeat.

Filename: install-elk.yml

```
########################################
---
- name: Configure Elk VM with Docker
  hosts: elkservers
  remote_user: sysadmin
  become: true
  tasks:
    # Use apt module
    - name: Install docker.io
      apt:
        update_cache: yes
        force_apt_get: yes
        name: docker.io
        state: present

      # Use apt module
    - name: Install python3-pip
      apt:
        force_apt_get: yes
        name: python3-pip
        state: present

      # Use pip module (It will default to
pip3)
    - name: Install Docker module
      pip:
        name: docker
        state: present

      # Use command module
    - name: Increase virtual memory
      command: sysctl -w
vm.max_map_count=262144

      # Use sysctl module
    - name: Use more memory
      sysctl:
        name: vm.max_map_count
        value: "262144"
        state: present
        reload: yes

      # Use docker_container module
    - name: download and launch a docker
elk container
      docker_container:
        name: elk
        image: sebp/elk:761
        state: started
        restart_policy: always
        # Please list the ports that ELK runs
on
        published_ports:
              - 5601:5601
              - 9200:9200
              - 5044:5044
########################################
```

Description of the Topology

The main purpose of this network is to have a load-balanced and monitored instance of DVWA, the D*mn Vulnerable Web Application. This allows having an image with known vulnerabilities to allow practicing pen-testing.

Load balancing ensures that the application will be highly available, in addition to restricting access to the network. Load balancing also makes sure tha the load is shared by all server machines in the group. It also provides for some level of DoS protection by diverting the load to another machine if one of hte machines is inundated with the DoS attack.

The jump-box in the group does not provide any service to the outside world but instead, it is used as a control machine to configure and provision the web machines and the ELK-server.

Integrating an ELK server allows users to easily monitor the vulnerable VMs for changes to the changes and monitor system logs and metrics such as CPU load, and possible attacks such as sudo escalation failures and SSH login attempts.

The configuration details of each machine may be found below. *Note: Use the Markdown Table Generator to add/remove values from the table*.

Machine Name	Function	IP address	Operating System
JumpBox	Gateway	52.156.78.1	Linux
Web-1	DVWA	10.0.0.6	Linux
Web-2	DVWA	10.0.0.10	Linux
Web-3	DVWA	10.0.0.12	Linux
ELK-Server	ELK Server	Pub:40.117.212.165 Priv: 10.1.0.4	Linux

Access Policies

The machines on the internal network are not exposed to the public Internet. They get their traffic through the Load Balancer. These machines can also be accessed from the Jump-box for configuration and management purposes.

Only the ELK-server machine can accept connections from the Internet. Access to this machine is only allowed from the following IP addresses: **104.220.125.86**

A summary of the access policies in place can be found in the table below.

Name	Publicly Accessible	Allowed IP Addresses
Jump Box	Yes	104.220.125.86
ELK-Server	Yes	104.220.125.86
Web1 (DVWA 1)	No	10.0.0.1-254
Web2 (DVWA 2)	No	10.0.0.1-254
Web3 (DVWA 3)	No	10.0.0.1-254

Elk Configuration

Ansible was used to automate the configuration of the ELK machine. No configuration was performed manually, which is advantageous because it makes the configuration repeatable, reusable, and eliminates human error in the configuration process.

To run the configuration you need to SSH into the jumpbox (configure the jumpbox, when created, with the proper public key from your machine). From there you need to check which docker containers are running using:

sudo docker container list -a

To get a list of the dockers available.

Start and attach to the proper docker using:
sudo docker start <docker name>
sudo docker attach <docker name>

From there you will need to make sure that the /etc/ansible/hosts files is updated with the lines describing the [elkservers] and indicated the internal IP address and add the lines identifying the ansible interpreter as in the

/etc/ansible/hosts file:

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group
 [elkservers]
10.1.0.4 ansible_python_interpreter=/usr/bin/python3

 [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110
10.0.0.6 ansible_python_interpreter=/usr/bin/python3
10.0.0.10 ansible_python_interpreter=/usr/bin/python3
10.0.0.12 ansible_python_interpreter=/usr/bin/python3
# If you have multiple hosts following a pattern you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com
```

Then from the docker container run the command:
ansible-playbook /etc/ansible/install-elk.yml

```
root@9fcf8e946936:~# ansible-playbook /etc/ansible/install-elk.yml

PLAY [Configure Elk VM with Docker] ********************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [10.1.0.4]

TASK [Install docker.io] *******************************************************************************
ok: [10.1.0.4]

TASK [Install python3-pip] *****************************************************************************
ok: [10.1.0.4]

TASK [Install Docker module] ***************************************************************************
ok: [10.1.0.4]

TASK [Increase virtual memory] *************************************************************************
changed: [10.1.0.4]

TASK [Use more memory] *********************************************************************************
ok: [10.1.0.4]

TASK [download and launch a docker elk container] ******************************************************
ok: [10.1.0.4]

PLAY RECAP *********************************************************************************************
10.1.0.4                   : ok=7    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=
0
```

The playbook implements the following tasks:

- Connects securely to the ELK server and gathers status on what is installed
- Installs docker.io
- Installs python3 pip
- Installs docker. This is needed so Ansible can then utilize that module to control docker containers
- Increases available memory. This is needed to allow for always restarting the ELK server
- Last thing it does is start the ELK server and enables it to communicate over the established ports for ELK Kibana.

The following screenshot displays the result of running `docker ps` after successfully configuring the ELK instance.

```
sysadmin@ELK-SERVER:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
                    NAMES
b21a58a96547        sebp/elk:761        "/usr/local/bin/star…"   2 weeks ago         Up 4 minutes        0.0.0.0:5044->5044/tcp, 0.0.0.0:5601->5601/tcp, 0.0.0.0:9200->9200
/tcp, 9300/tcp   elk
sysadmin@ELK-SERVER:~$
```

Target Machines & Beats

This ELK server is configured to monitor the following machines:

Web1 (DVWA 1)	10.0.0.6
Web2 (DVWA 2)	10.0.0.10
Web3 (DVWA 3)	10.0.0.12

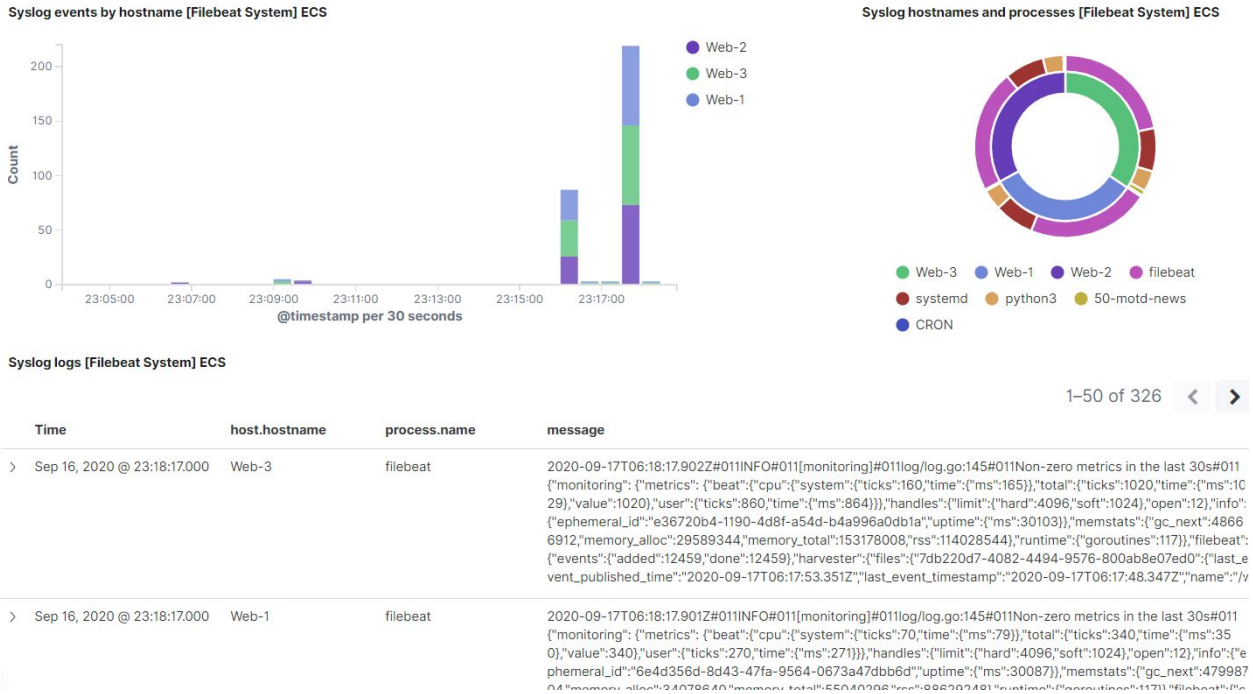
We have installed the following Beats on these machines using the following curl command:

curl -L -0 https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.6.1-amd64.deb

which is included in the ansible playbook file: **filebeat-playbook.yml:**

To check the installation, from the public machine navigate to:
http://[40.117.212.165]:5601/app/kibana the IP address being that of the ELK Server.

These Beats allow us to collect the following information from each machine: the logs and metrics from our unique environments and document them with essential metadata from hosts, container platforms like Docker and Kubernetes,



Using the Playbook

In order to use the playbook, you will need to have an Ansible control node already configured. In our case it was already created and is in fact the jumpbox that was configured already.

SSH into the control node and follow the steps below:

- Copy the **filebeat-playbook.yml** file to **/etc/ansible** directory.

- Update the **/etc/filebeat/filebeat-config.yml** file to include:
  Hosts:[“10.1.0.4:9200”]
  Username: “elastic”
  Password: “changme” , this password can be changed later.

- Run the playbook, and navigate to alread open to check that the installation worked as expected.