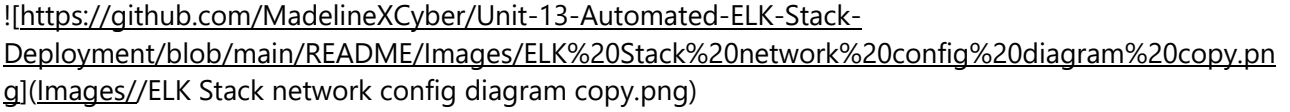
Automated ELK Stack Deployment

This READ ME document contains the following details:

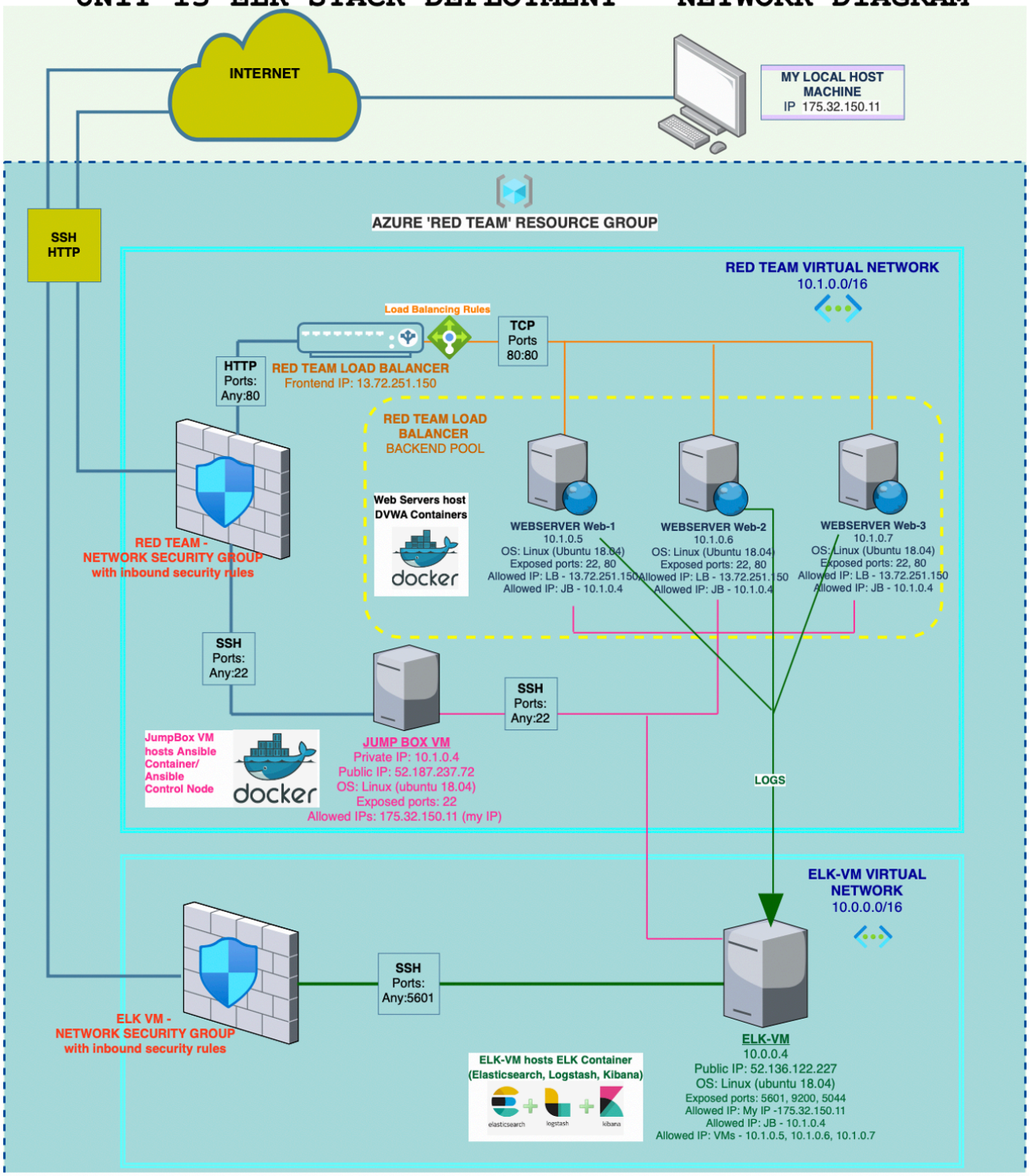
- Description of the Topology
- Access Policies
- ELK Configuration
    - Beats in Use
    - Machines Being Monitored
- How to Use the Ansible Build

Network configuration

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



UNIT 13 ELK STACK DEPLOYMENT – NETWORK DIAGRAM



SECURITY RULES

RED TEAM VIRTUAL NETWORK - 10.1.0.0/16

Allow JumpBox VM (10.1.0.4) full access to internal Vnet using SSH connection on Port 22
Allow SSH traffic on Port 22 from local host station with IP 175.32.150.11 to Jump Box VM (10.1.0.4)
Allow HTTP traffic on Port 80 from local host station with IP 175.32.150.11 to Load Balancer (13.72.251.150)
Allow HTTP traffic from Load Balancer (13.72.251.150) to internal Vnet using TCP Port 80:80

ELK-VM VIRTUAL NETWORK - 10.0.0.0/16

Allow SSH traffic on Port 5601 from local host station with IP 175.32.150.11 to ELK-VM (10.0.0.4)

LOAD BALANCING RULES

Load Balancer to communicate with backend internal network using Port 80

All files 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 yml files may be used to install only certain pieces of it, such as Filebeat.

The playbooks used for this deployment are as follows:

- Ansible configuration file
- Ansible hosts file
- Configure ELK VM with docker
- Configure Web VM with Docker
- Filebeat configuration file
- FIlebeat playbook
- Metricbeat configuration file
- Metricbeat playbook

Description of the Topology

The main purpose of this network is to expose a load-balanced and monitored instance of DVWA, the D*mn Vulnerable Web Application. The DVWA site allows the cybersecurity industry to develop, learn and test security tools and skills in a legal environment.

The network topology above includes load balancing which ensures that the application will be highly available, in addition to restricting traffic flow and access to the network.

- A **Load Balancer** is used to harden the network by protecting its availability and adding resilience to the overall system. By incorporating a Load Balancer into our network architecture, all incoming traffic – in our case HTTP requests – is initially routed to a single point at the Load Balancer's external frontend, before being redistributed to our 3 internal web servers (Web-1, Web-2 and Web-3) in the backend pool. As the purpose of Load Balancers is to manage network traffic and divide it between the backend servers based on traffic flow, this helps to ensure maximum reliability and uptime of the network and provides critical redundancy to the system. Load Balancers also undertake network 'health checks' and have the ability to incorporate specific security rules, both important measures in providing additional safeguards and security to the network.

- This network also includes a **Jump Box** VM, an administration server which acts as an intermediary or SSH host to the internal network, once again by managing and controlling access to the internal network. An additional advantage of the Jump Box is that it is an intelligent device and can also be used as a control panel to perform critical functions such as system configurations and updates. In order to set up this particular network, the Jump Box was used to install Docker containers on our Web-VMs and then run an Ansible playbook to configure the Web-VMs with DVWA container images. Our Jump Box VM was also used to setup and configure a VM as an ELK server (to run an ELK Stack container) using Ansible.

Incorporating an ELK server into the network allows users to easily monitor the vulnerable VMs for changes to the network activity and system logs.

This is achieved using ELK Stack, a powerful, open-source tool used to store, search, analyse and visualise many different forms of data. ELK is an acronym for the 3 components which make up Elastic Stack - Elasticsearch, Logstash and Kibana.

- ELASTICSEARCH is a powerful tool which allows the user to store, search and analyse data. It has the ability to handle huge volumes of data in almost real-time ie. milliseconds.

- LOGSTASH is a data processing pipeline that collects log data from different sources, converting different log data into a uniform format if necessary. It is used to feed data to Elasticsearch.

- KIBANA is a tool used to visualise data indexed in Elasticsearch. The user can generate a variety of charts, graphs, maps and metrics using Kibana's complex dashboard.

Due to the significant amount of information potentially contained in the Elasticsearch log database, a tool known as 'Beats' is now available as part of the ELK Stack suite to allow collection of specific data and information. There are 8 official Beats in total, two of which are used in this deployment – Filebeat and Metricbeat (see also 'Target Machines & Beats' below).

- **Filebeat** is used to monitor specific log files or locations, as specified by the user. Filebeat collates and organises the requested data, which is then forwarded to Elasticsearch or Logstash for indexing. Filebeat watches for changes by monitoring the file system and specific logs. As it is specific to a particular machine, filebeat must be installed on each individual VM/server to be monitored.

- **Metricbeat** collects and records the metrics of a machine from the operating system and services running on the server. These metrics allow the user to assess such things as the health of a network, as well as monitoring for signs of suspicious activity, for example CPU usage and uptime. As with filebeat, metricbeat is specific to a particular machine and must be installed on each individual VM/server which is being monitored.

Our final network topology consists of a Jump Box VM, 3 Web Servers and an ELK-VM. The configuration details of each machine may be found below.

Name	Function	IP Address	Operating System
Jump Box VM	Gateway, intelligence Ansible control node	Public: 52.187.237.72 Private: 10.1.0.4	Linux (Ubuntu 18.04)
Web-1	Internal web server DVWA container	Public: Load balancer public IP Private: 10.1.0.5	Linux (Ubuntu 18.04)
Web-2	Internal web server DVWA container	Public: Load balancer public IP Private: 10.1.0.6	Linux (Ubuntu 18.04)
Web-3	Internal web server DVWA container	Public: Load balancer public IP Private: 10.1.0.7	Linux (Ubuntu 18.04)
ELK-VM	Log server ELK Stack container	Public: 40.87.108.196 Private: 10.0.0.4	Linux (Ubuntu 18.04)

Access Policies

The machines on the internal network are not exposed to the public Internet.

Only the Jump Box VM and Load Balancer machines can accept connections from the Internet. Access to these machines is only allowed from the following IP address:

- My local host machine with public (*dynamic) IP: 175.32.150.11

Machines within the network can only be accessed by the Jump Box.

- The Jump Box VM can access the ELK VM through the internal network.
- My local host machine can access the ELK VM using its external IP.

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

Name	Publicly Accessible	Allowed IP Addresses
Jump Box VM	Yes	My host machine: IP 175.32.150.11 Web-1: 10.1.0.5 Web-2: 10.1.0.6 Web-3: 10.1.0.7 ELK-VM: 10.0.0.4
Web-1	No	Jump Box VM: 10.1.0.4 Load Balancer: 13.72.251.150
Web-2	No	Jump Box VM: 10.1.0.4 Load Balancer: 13.72.251.150
Web-3	No	Jump Box VM: 10.1.0.4 Load Balancer: 13.72.251.150

Name	Publicly Accessible	Allowed IP Addresses
ELK-VM	Yes	My host machine: 175.32.150.11 Web-1: 10.1.0.5 Web-2: 10.1.0.6 Web-3: 10.1.0.7 Jump Box VM: 10.1.0.4

Elk Configuration

Ansible was used to automate configuration of the ELK machine. No configuration was performed manually, which is advantageous because...

- Automated configuration streamlines and simplifies network and system configurations as it allows us to execute complex and multiple commands/scripts in one command.
- Automated configuration allows us to configure multiple servers/machines identically, and simultaneously.
- There is less room for human error using automation. This is particularly important when configuring multiple machines which require identical configuration.
- An automated process is much easier to use and less time consuming than configuration through a manual process, which generally requires configuration one machine at a time.

We used the following playbook to configure the ELK machine:

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

      # Use apt module
    - name: Install pip3
      apt:
        force_apt_get: yes
        name: python3-pip
        state: present

      # Use pip module
    - name: Install Docker python module
      pip:
        name: docker
        state: present

      # 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
        published_ports:
          - 5601:5601
          - 9200:9200
          - 5044:5044

      # Use systemd module
    - name: Enable service docker on boot
      systemd:
        name: docker
        enabled: yes

```

The playbook used implements the following tasks:

- Install the docker package, docker.io, python3-pip (the package-management system written in Python which is used to install and manage software packages) and docker.
- Configure the target machine to use more virtual memory when running the ELK container
- Install the docker module using python3-pip.
- Download and launch the docker ELK container, sebp/elk:761. The image should be start using three specific port mappings: 5601:5601, 9200:9200 and 5044:5044.
- Use the systemd module to configure automatic restart of the docker service when the machine reboots.

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

```
azadmin@ELK-VM:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                                                                  NAMES
50fe9bd50b7d        sebp/elk:761        "/usr/local/bin/star…"   3 minutes ago       Up 3 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
azadmin@ELK-VM:~$
```

Target Machines & Beats

This ELK server is configured to monitor the following machines:

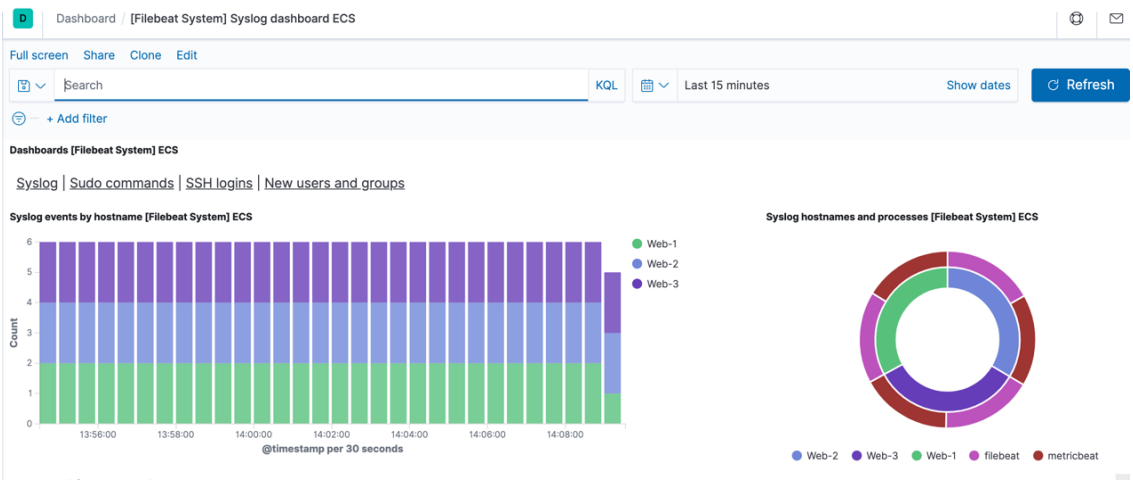
- Web-1: 10.1.0.5
- Web-2: 10.1.0.6
- Web-3: 10.1.0.7

We have installed the following Beats on these machines:

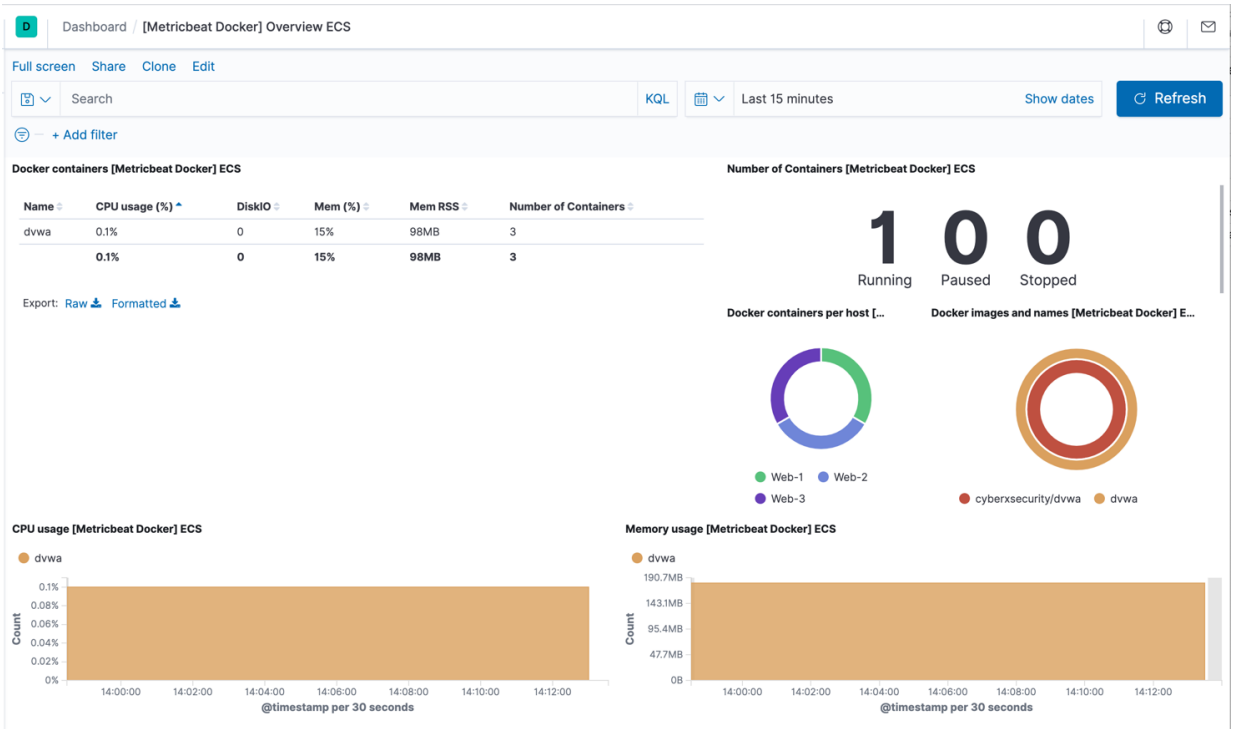
- Filebeat
- Metricbeat.

These Beats allow us to collect the following information from each machine:

- **Filebeat** is used to monitor specific log files or locations, as specified by the user. Filebeat collates and organises this data, which is then forwarded to Elasticsearch or Logstash for indexing. Filebeat watches for changes in data by monitoring the file system and specific logs – see sample of system log activity below. As it is specific to a particular machine, Filebeat must be installed on each individual VM/server to be monitored.



- **Metricbeat** collects and records the metrics of a machine from the operating system and services running on the server, for example CPU and memory usage and container information (see below). These metrics allow the user to assess such things as the health of a network, as well as monitoring for signs of suspicious activity. As with Filebeat, Metricbeat is specific to a particular machine and must be installed on each individual VM/server which is being monitored.



Using the Playbook

In order to use the playbook, you will need to have an Ansible control node already configured. Assuming you have such a control node provisioned:

SSH into the control node and follow the steps below:

- Copy the filebeat-playbook.yml file to the /etc/ansible/roles folder.

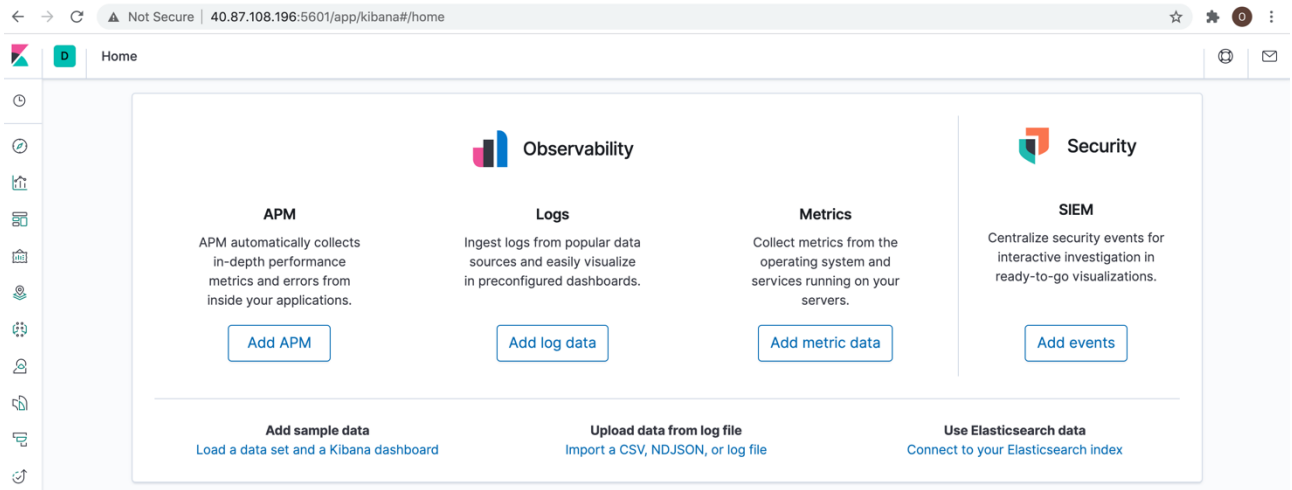
- Update the filebeat-config.yml file to include the ELK-VM IP details at lines 1106 and 1806, as follows:
    - Configure Elasticsearch output at line 1106: **hosts: ["10.0.0.4:9200"]**
    - Kibana endpoint configuration at line 1806: **host: "10.0.0.4:5601"**

- Run the playbook.

```yaml
---
- name: Installing and Launch Filebeat
  hosts: webservers
  become: yes
  tasks:
    # Use command module
  - name: Download filebeat .deb file
    command: curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.4.0-amd64.deb

    # Use command module
  - name: Install filebeat .deb
    command: dpkg -i filebeat-7.4.0-amd64.deb

    # Use copy module
  - name: Drop in filebeat.yml
    copy:
      src: /etc/ansible/files/filebeat-config.yml
      dest: /etc/filebeat/filebeat.yml

    # Use command module
  - name: Enable and Configure System Module
    command: filebeat modules enable system

    # Use command module
  - name: Setup filebeat
    command: filebeat setup

    # Use command module
  - name: Start filebeat service
    command: service filebeat start

    # Use systemd module
  - name: Enable service filebeat on boot
    systemd:
      name: filebeat
      enabled: yes

```

Navigate to the Filebeat installation page on the ELK server GUI using the ELK-VM public IP **(http://:** 40.87.108.196**/app/kibana)** to check that the installation worked as expected.

Take a screenshot of the result.



Note – to install Metricbeat, follow a similar process to the above

Bonus

As a **Bonus**, provide the specific commands the user will need to run to download the playbook, update the files, etc.

Connect from local host machine to the JumpBox VM using SSH on port 22. Once connected to the JumpBox VM, check sudo permissions.	myterminal:~$ ssh azadmin@52.187.237.72 azadmin@JumpBox2:~$ sudo -l
Install Docker onto the Jumpbox VM.	azadmin@JumpBox2:~$ sudo apt update azadmin@JumpBox2:~$ sudo apt install docker.io
Once Docker is installed, pull the cyberxsecurity/ansible container onto the Jumpbox VM.	azadmin@JumpBox2:~$ sudo docker pull cyberxsecurity/ansible.
Launch the Ansible container in a bash shell and connect to it.	azadmin@JumpBox2:~$ docker run -ti cyberxsecurity/ansible:latest bash
One it has been successfully launched, exit the container.	root@79af822c5787:~# exit
Create a new Network Security Group Rule for the RedTeam which allows the JumpBox full access to the Vnet	
Find the previously installed cyberxsecurity/ansible container and connect with it. *Note – the image for the cyberxsecurity/ansible container is cool_saha*	azadmin@JumpBox2:~$ sudo docker container list -a azadmin@JumpBox2:~$ docker run -it cyberxsecurity/ansible /bin/bash
Generate a new SSH public/private key pair from inside the Ansible container and reset the VM passwords with the new public key.	root@79af822c5787:~# ssh-keygen root@79af822c5787:~# cat .ssh/id_rsa.pub root@79af822c5787:~# cp .ssh/id_rsa.pub
Test connection from the Ansible container to the Web-VMs using ping. Access the Web-VMs from the Ansible container using SSH.	Web-1: root@79af822c5787:~# ping 10.1.0.5 root@79af822c5787:~# ssh azadmin@10.1.0.5 Web-2: root@79af822c5787:~# ping 10.1.0.6 root@79af822c5787:~# ssh azadmin@10.1.0.6
Locate the Ansible hosts file	root@79af822c5787:~# ls /etc/ansible/ …hosts…
Update the Ansible hosts file to include IPs for the Web-VMs. Note – the python line needs to be included with each IP: ansible_python_interpreter=/usr/bin/python3	root@79af822c5787:~# nano /etc/ansible/hosts Uncomment the [webservers] header line Add the Web-VM IPs: 10.1.0.5 ansible_python_interpreter=/usr/bin/python3 10.1.0.6 ansible_python_interpreter=/usr/bin/python3

	Save changes and exit the nano file: ^C > Y > enter
Locate the Ansible config file	root@79af822c5787:~# ls /etc/ansible/ …ansible.config…
Update the remote_user in the Ansible config file to include azadmin, the admin username for the JumpBox and Web VMs.	root@79af822c5787:~# nano /etc/ansible/ansible.cfg Uncomment the remote_user line and replace root with azadmin: remote_user = azadmin Save changes and exit the nano file: ^C > Y > enter
Check updates to the hosts and config files by testing connections to the VMs from the Ansible container.	root@79af822c5787:~# ansible all -m ping
Create an Ansible playbook to install Docker and configure the Web-VMs with the DVWA web app. - Use apt module to install docker.io and python3- - Update the cache - Use the Ansible pip module to install docker - Install the cyberxsecurity/dvwa container. Use port 80 on the container to port 80 on the host. - Set the restart policy so that the container always restarts with the VM. - Use the systemd module to restart the docker service when the machine reboots. *NB. To check syntax of YAML files, use YAMLlint: www.yamllint.com*	root@79af822c5787:~# nano /etc/ansible/config-WebVMs.yml
Run the Ansible pentest.yml playbook.	root@79af822c5787:~# ansible-playbook /etc/ansible/pentest.yml
Set up a new ELK-STACK VM in Azure in the existing Resource Group using a new region and separate Vnet.	
In order to complete setup, connect to the JumpBox from terminal on the host machine and then start the existing Ansible container to access the public SSH key.	myterminal:~$ ssh azadmin@52.187.237.72 azadmin@JumpBox2:~$ docker start cool_saha azadmin@JumpBox2:~$ docker attach cool_saha root@79af822c5787:~# cat .ssh/id_rsa.pub root@79af822c5787:~# cp .ssh/id_rsa.pub
Update the Ansible hosts file to include the new ELK-VM. Create a separate group heading, [elk].	root@79af822c5787:~# nano /etc/ansible/hosts Add the ELK-VM IP underneath a new ELK group heading:

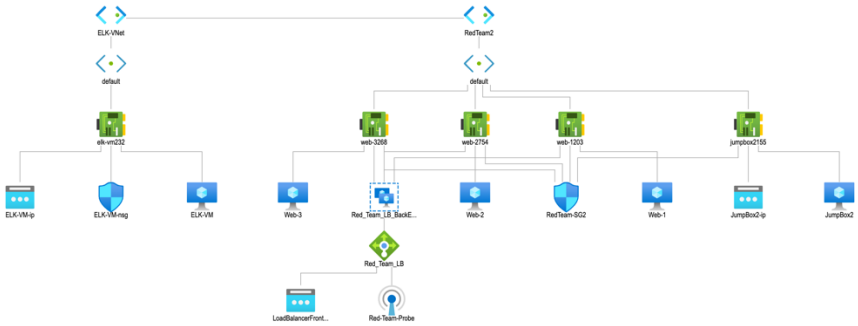
Add the IP for the new ELK-VM: 10.0.0.4. Include the python line: ansible_python_interpreter=/usr/bin/python3	[elk] 10.0.0.4 ansible_python_interpreter=/usr/bin/python3 Save changes and exit the nano file: ^C > Y > enter
Create an Ansible playbook in YAML to configure the new ELK-VM server. - This playbook needs to specify the applicable group (ie. elk. - In order to run the ELK container virtual memory needs to be increased. - Install docker.io and python3-pip and docker. - After Docker is installed, download and run the sebp/elk:761 container. - The container should bee started with the following ports: 5601:5601 9200:9200 5044:5044 se port 80 on the container to port 80 on the host. - Use the systemd module to restart the docker service when the machine reboots. *NB. To check syntax of YAML files, use YAMLlint: www.yamllint.com*	root@79af822c5787:~# nano /etc/ansible/install-elk.yml Playbook: install-elk.yml
Run the Ansible install-elk.yml playbook.	root@79af822c5787:~# ansible-playbook /etc/ansible/install-elk.yml
After the playbook has run, SSH to the ELK-VM and double check that the elk-docker container is running. Take a screenshot of the result.	root@79af822c5787:~# ssh azadmin@10.0.0.4 Then run: sudo docker ps Take a screenshot of the result.
Create a new incoming rule for the new Network Security Group which allows TCP traffic over port 5601 from the local host address.	
Test the setup is working correctly by navigating to the Kibana home page using the ELK-VM public IP.	http://40.87.108.196:5601/app/kibana#/home
Navigate back into the ELK-VM and start the docker container to check that the ELK server container is up and running, then exit.	root@79af822c5787:~# ssh azadmin@10.0.0.4 azadmin@ELK-VM:~$ docker container list -a azadmin@ELK-VM:~$ exit
Create a Filebeat configuration file:	azadmin@JumpBox2:~$ docker start cool_saha

- Navigate into the Jump Box - Open the Ansible container - Copy the filebeat-config.yml configuration template using curl into the etc/ansible/ folder	azadmin@JumpBox2:~$ docker attach cool_saha root@79af822c5787:~# curl https://gist.githubusercontent.com/slape/5cc350109583af6cbe577bbcc0710c93/raw/eca603b72586fbe148c11f9c87bf96a63cb25760/Filebeat >> /etc/ansible/filebeat-config.yml
Open the filebeat-config.yml in nano and edit it as follows: - Update line 1106 and replace the IP with the private IP of the ELK machine - Update line 1806 and replace the IP with the private IP of the ELK machine - Save the update configuration file by making a copy to the /etc/ansible/files/ folder	root@79af822c5787:~# nano /etc/ansible/filebeat-config.yml #1106 output.elasticsearch: hosts: ["10.1.0.4:9200"] username: "elastic" password: "changeme" #1186 setup.kibana: host: "10.1.0.4:5601" root@79af822c5787:~# cp /etc/ansible/filebeat-config.yml /etc/ansible/files/filebeat-config.yml.
Create a Filebeat installation playbook: Download the .deb file from artifacts.elastic.co.and then install it using the dpkg command.	root@79af822c5787:~# dpkg -i filebeat-7.4.0-amd64.deb
Update the filebeat-playbook.yml and locate it in the etc/ansible/roles/ folder	Playbook: filebeat-playbook.yml
Run the playbook	root@79af822c5787:~# ansible-playbook filebeat-playbook.yml
To check if successfully installed, return to the Kibana homepage and scroll to Step5: Module to 'Check Data'. It should be receiving logs.	

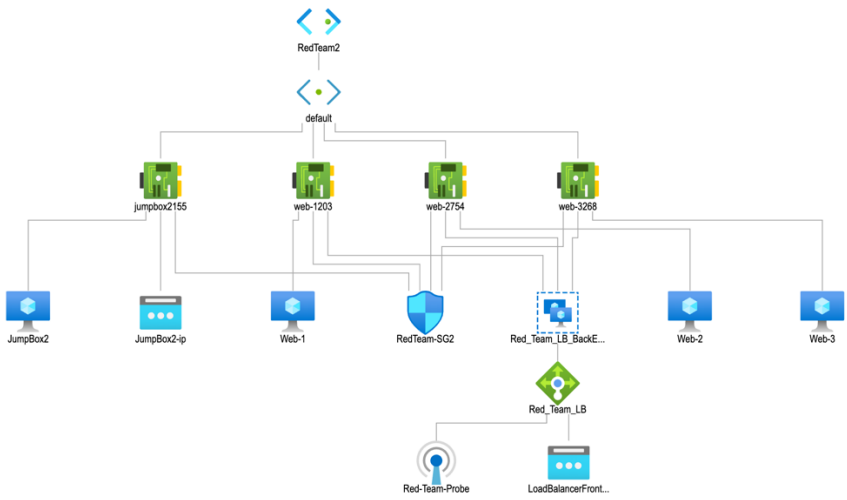
Additional material

DIAGRAMS
Additional diagrams featuring each of the networks in Azure can be found by following these links:

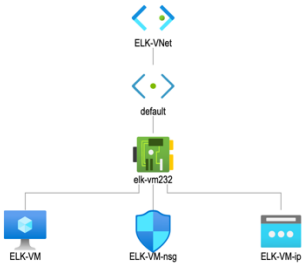
- Azure Network Watcher Topology - Red Team Resource Group.png



- Azure Network Watcher Topology - Red Team VNet.png



- Azure Network Watcher Topology - ELK-VM VNet.png



LINUX COMMANDS

LINUX COMMANDS: WK 3 Lucky Duck

Roulette_dealer_finder_by_time_sh

```
#!/bin/bash
grep 'AM\|PM' *_Dealer_schedule > roulette.dealer1
awk -F' ' '{print $1, $2, $5, $6}' roulette.dealer1 > roulette.dealer2
grep $1_Dealer roulette.dealer2 > roulette.dealer3
grep schedule:$2 roulette.dealer3 > roulette.dealer4
awk -F' ' '{print $2}' roulette.dealer4 | grep $3 roulette.dealer4 > results.roulette_dealer_finder_by_time
```

Roulette_dealer_finder_by_time_and_game_sh

```
#!/bin/bash
grep 'AM\|PM' *_Dealer_schedule > finder.dealer1
grep $1_Dealer finder.dealer1 > finder.dealer2
grep schedule:$2 finder.dealer2 > finder.dealer3
awk -F' ' '{print $2}' finder.dealer3 | grep $3 finder.dealer3 >
results.roulette_dealer_finder_by_time_and_game
```

LINUX COMMANDS: WK 4 Linux Systems Administration

Command to inspect permissions: **ls -l shadow**

Command to set permissions (if needed): **sudo chmod 600 shadow**

Command to inspect permissions: **ls -l gshadow**

Command to set permissions (if needed): **sudo chmod 600 gshadow**

Command to inspect permissions: **ls -l group**

Command to set permissions (if needed): **sudo chmod 644 group**

Command to inspect permissions: **ls -l passwd**

Command to set permissions (if needed): **sudo chmod 644 passwd**

Command to add each user account (include all five users): **sudo adduser sam**

Command to add admin to the sudo group: **sudo usermod -G sudo admin**

Command to add group: **sudo addgroup engineers**

Command to add users to engineers group (include all four users):
sudo usermod -aG engineers sam
sudo usermod -G engineers joe

Command to create the shared folder: **sudo mkdir /home/engineers**

Command to change ownership of engineer's shared folder to engineer group: **sudo chown sysadmin:engineers /home/engineers**

Command to install Lynis: **sudo apt install lynis**

Command to see documentation and instructions: **man lynis; sudo lynis show help**

Command to run an audit: **sudo lynis audit system**

LINUX COMMANDS: WK 5 Archiving and Logging Data

Command to **extract** the TarDocs.tar archive to the current directory: **tar xvvf TarDocs.tar**

Command to **create** the Javaless_Doc.tar archive from the TarDocs/ directory, while excluding the TarDocs/Documents/Java directory: **tar cvvWf Javaless_Docs.tar --exclude=Java ~/Projects/TarDocs/Documents/**

Command to ensure Java/ is not in the new Javaless_Docs.tar archive: **tar tvf Javaless_Docs.tar | grep Java**

Command to create an incremental archive called logs_backup_tar.gz with only changed files to snapshot.file for the /var/log directory:

```
sudo tar czvf logs_backup_sun.tar.gz --listed-incremental=logs_backup.snar --level=0 /var/log
sudo tar czvf logs_backup_mon.tar.gz --listed-incremental=logs_backup.snar /var/log
sudo tar czvf logs_backup_tues.tar.gz --listed-incremental=logs_backup.snar /var/log
```

Cron job for backing up the /var/log/auth.log file: **0 6 * * 3 tar -czf /var/log/auth_backup.tgz /var/log/auth.log**

Brace expansion command to create the four subdirectories: **sudo mkdir -p ~/backups/{freemem,diskuse,openlist,freedisk}**

System.sh script edits below:

```
#!/bin/bash
free -h > ~/backups/freemem/free_mem.txt
du -h > ~/backups/diskuse/disk_usage.txt
lsof > ~/backups/openlist/open_list.txt
df -h > ~/backups/freedisk/free_disk.txt
```

Command to make the system.sh script executable: **sudo chmod +x system.sh**

Commands to test the script and confirm its execution:

```
sudo ./system.sh
cat ~/backups/freemem/free_mem.txt
cat ~/backups/diskuse/disk_usage.txt
cat ~/backups/openlist/open_list.txt
cat ~/backups/freedisk/free_disk.txt
```

Command to copy system to system-wide cron directory: **sudo cp system.sh /etc/cron.weekly/**

Configure a log rotation scheme that backs up authentication messages to the /var/log/auth.log. Add your config file edits below:

```
/var/log/auth.log {
    missingok
    weekly
    rotate 7
    notifempty
    compress
    delaycompress
    endscript
}
```

Command to verify auditd is active: **systemctl status auditd**

Command to set number of retained logs and maximum log file size. Add the edits made to the configuration file below:

```
num_logs = 7
max_log_file = 35
```

Command using auditd to set rules for /etc/shadow, /etc/passwd and /var/log/auth.log: Add the edits made to the rules file below:

```
-w /etc/shadow -p wra -k hashpass_audit
-w /etc/passwd -p wra -k userpass_audit
-w /var/log/auth.log -p wra -k authlog_audit
```

Command to restart auditd:**sudo systemctl restart auditd**

Command to list all auditd rules: **sudo auditctl -l**

Command to produce an audit report for all user authentications: **sudo aureport -au**

Create a user with sudo useradd attacker and produce an audit report that lists account modifications: **sudo aureport -m**

Command to use auditd to watch /var/log/cron: **sudo auditctl -w /var/log/cron**

Command to verify auditd rules: **sudo auditctl -l**

Command to return journalctl messages with priorities from emergency to error since the current system boot: **sudo journalctl -p 3 -b**

Command to check the disk usage of the system journal unit since the most recent boot: **sudo journalctl -u systemd-journald -b**

Command to remove all archived journal files except the most recent two: **sudo journalctl --vacuum-files=2**

Command to filter all log messages with priority levels between zero and two, and save output to /home/sysadmin/Priority_High.txt: **sudo journalctl -p 2 >> /home/sysadmin/Priority_High.txt**

Command to automate the last command in a daily cronjob. Add the edits made to the crontab file below: **0 0 * * * sudo journalctl -p 2 >> /home/sysadmin/Priority_High.txt**