**Microsoft Azure**

portal.azure.com

**Create – Resource Group**

Add

Subscription detail

Resource Group – any name you would like

Resource Region – keep resources in same region – (until you want to create another region)

Click next until “Validation Pass”

Click “Create”

**Create - Virtual Network**

Search – Virtual Network

Click add

Resource Group – click down arrow – select “Resource Group Created”

Instance Detail – “Name”

Region – select region – (same as Resource Group)

Next until “Validation Passed”

Click “Create”

**Create - Network Security Group**

Search - Network Security Group

Resource Group – click down arrow – select “Resource Group Created”

Instance Detail – “Name”

Resource Region – keep resources in same region – (until you want to create another region)

Click next until “Validation Pass”

Click “Create”

**Inbound Security Rules**

d-click RedTeam-SG

click – under Settings / Inbound Security Rules

**SSH\_Access**

click Add

Source: IP Address

Source IP addresses/CIDR ranges: 76.168.248.146

Source port ranges: \* (any)

Destination: Any

Destination port ranges: 22

Protocol: TCP

Action: Allow

Priority: 4076

Name: SSH\_Access

The default firewall rule is deny any any – which denys any protocol on any port

This creates the inbound rule for ssh through the firewall.

* The less restrictive rules must have a lower priority than the more restrictive rules.
* If a more restrictive rule come before a less restrictive rule, the less restrictive rule will never be applied.

**Inbound\_SSH**

click – under Settings / Inbound Security Rules

click Add

Source: IP Address

Source IP addresses/CIDR ranges: 10.0.0.5

Source port ranges: \* (any)

Destination: Virtual Network

Destination port ranges: 22

Protocol: Any

Action: Allow

Priority: 4056

Name: Inbound\_SSH

**Port\_80**

click – under Settings / Inbound Security Rules

click Add

Source: IP Address

Source IP addresses/CIDR ranges: 76.168.248.146

Source port ranges: \* (any)

Destination: Virtual Network

Destination port ranges: 80

Protocol: Any

Action: Allow

Priority: 4036

Name: Port\_80

Descrption: HTTP Access

**Generate SSH Public Key**

From the remote computer.

Launch Git-Bash

type ssh-keygen

type cat ~/.ssh/id\_rsa.pub

copy public key to clipboard

This key will be used in the virtual machines

**In Azure create the Jumpbox VM**

The Jumpbox is created to be a central location administration computer.

All other virtual machines are configured from this VM.

Click on Search and select Virtual Machine

Click Add

Subscription: Default

Resource Group: Group you created

Virtual Machine Name: JumpBox

Region: Keep All Machines in the same Region

Availability Options: No Infrastructure

Image: Ubuntu Server 18.04 LTS-Gren1

Size: Standard B1s 7.95m

Admin Account

Auth Type: SSH Public Key

User Name: azureuser

SSH Public Key Source: Use existing public key

SSH Public Key: Paste from clipboard

Public Inbound Rules: Allow selected ports

Select in bound Ports: SSH (22)

next

Disc Size: Default

Disc Type: Standard HDD

Encryption: Default

Network

Virtual Network

Subnet

Public IP

Review and Create

Once created and the jumpbox is running, click on the Jumpbox, take note of the Public IP Address: 23.99.190.121

**Create Two Web Servers**

Click on Search and select Virtual Machine

Click Add

Subscription: Default

Resource Group: Group you created

Virtual Machine Name: Web-1 / Web-2

Region: Keep All Machines in the same Region

Availability Options: No Infrastructure

Image: Ubuntu Server 18.04 LTS-Gren1

Size: Standard B1s 7.95m

Admin Account

Auth Type: SSH Public Key

User Name: azureuser

SSH Public Key Source: Use existing public key

SSH Public Key: remote computer ssh public key (this will be changed to the ansible ssh public key later)

Public Inbound Rules: Allow selected ports

Select in bound Ports: SSH (22)

next

Disc Size: Default

Disc Type: Standard HDD

Encryption: Default

Network

Virtual Network

Subnet

Public IP

Review and Create

**Install the Docker Container**

**SSH into the JumpBox**

SSH into the jumpbox:

From the remote computer open a bash session:

ssh [azureuser@23.99.190.121](mailto:azureuser@23.99.190.121)

When prompted to continue type: yes

**At the Jump Box command prompt:**

sudo su

enter the password for the azureuser (or administrator username/password)

apt-get update

apt install docker.io

systemctl status docker

**Install Docker**

docker pull cyberxsecurity/ubuntu:bionic

docker run -ti bionic/ubuntu bash

List all containers created on the JumpBox

docker container list -a

Container name will be at the end of the screen display

docker container start <container name>

docker passwords

Activate a shell on the container

docker attach <container name>

Generate an ssh key from the ansible container

ssh-key gen

cat ~/.ssh/.pub

Copy the public key to a clipboard and paste into notepad under *ansible ssh.*

**Edit the Ansible.cfg File**

cd /etc/ansible

ls

nano ansible.cfg

Once the ansible.cfg is open in nano press “ctrl w” type “remote\_user”, enter to search.

Uncomment the line and add your user name: (azureuser or ansible)

exit and save

**Edit the hosts file**

nano hosts

“ctrl w” type “webservers”, enter to search.

Add Web servers IP Addresses and ansible\_python\_interpreter=/usr/bin/python3

#

# List the IP Addresses of your webservers

# You should have at least 2 IP addresses

[webservers]

10.0.0.4 ansible\_python\_interpreter=/usr/bin/python3

10.0.0.5 ansible\_python\_interpreter=/usr/bin/python3

#

**On Azure Network**

**Edit the User Password to the Web Servers**

Open Web-1

scroll down Support+ troubleshooting

Select Reset Password

Select Reset SSH Public Key

Username: Enter the user name placed in the ansible.cfg file (azureuser or ansible)

SSH public key: Paste the public key code generated from the docker container.

**Repeat this process for Web-2 and Web-3**

**Create an ansible playbook**

**On the JumpBox**

*connect to an ansible container.*

docker container list -a

*start the container*

docker start <container name>

*launch a shell*

docker attach <container name>

**Edit a YAML playbook**

nano /etc/ansible/pentest.yml

---

* name: Config Web VM with Docker

hosts: web

become: true

tasks:

---

**Install docker.io and python3-pip**

* name docker.io

apt:

update\_cache: yes

name: docker.io

state: present

* name: Install pip3

apt:

force\_apt\_get: yes

name: python3-pip

state: present

**Install docker with the ansible pip module**

* name: Install Python Docker Module

pip:

name: docker

state: present

**Install cyberxsecurity/dvwa container**

* name: download and launch a docker webcontainer

docker container:

name: dvwa

image: cyberxsecurity/dvwa

state: started

restart\_policy: always

published\_ports: 80:80

SEE My\_YAML\_Playbook – readme

**Run the Ansible Playbook**

ansible-playbook pentest.yml

**Create Load Balancer in front of the web server VM’s**

Search load balancer

Click Add

Assign a static IP Address

Create

Add a Health Probe

Name: Web-80

Protocol: TCP

Port: 80

Interval: 5

Unhealthy threshold: 2

Used by: Inbound-HTTP

**Create Backend Pool add Web servers**

**Virtual machine IP Configuration Availability set**

web-1 ipconfig1 (10.0.0.4) WEBSERVERS

web-2 ipconfig1 (10.0.0.5) WEBSERVERS

**Create Load Balancing Rule to forward port 80 to Red Team Virtual Network**

Name: InBound\_HTTP

IP version: IPv4

Frontend IP Address: null (load balancer)

Protocol: TCP

Port: 80

Backend Port: 80

Backend Pool: Web-Backend-Pool (3 Virtual Machines)

Health Probe: WEB-80 (TCP:80)

Session Persistence: Client IP and Protocol

Idle timeout: 4 min

Floating IP: Disabled

**Create a Security Group rule to allow traffic from port 80 internet too internal VNet.**

Source: IP Addresses

Source: External IPv4 IP Address

Source port ranges: \* (any)

Destination: VirtualNetwork

Destination port ranges: 80

Protocol: Any

Action: Any

Priority: 4036

Name: Port\_80

Description: HTTP Access from external ANY

**Create another Web Server**

Click on Search and select Virtual Machine

Click Add

Subscription: Default

Resource Group: Group you created

Virtual Machine Name: Web-3

Region: Keep All Machines in the same Region

Availability Options: No Infrastructure

Image: Ubuntu Server 18.04 LTS-Gren1

Size: Standard B1s 7.95m

Admin Account

Auth Type: SSH Public Key

User Name: azureuser

SSH Public Key Source: Use existing public key

SSH Public Key: Paste the public key generated from the ansible container

Inbound Rules: Allow selected ports

Select in bound Ports: SSH (22)

next

Disc Size: Default

Disc Type: Standard HDD

Encryption: Default

Network

Virtual Network

Subnet

Public IP

Review and Create

**On the Web-1 and Web-2 servers**

Reset Password

Select: Reset SSH Public Key

Username: azureuser or ansible

SSH Public Key: Paste the public key generated from the ansible container

Update.

**Add Web-3 to Load Balancer Backend-Pool**

**Virtual machine IP Configuration Availability set**

web-1 ipconfig1 (10.0.0.4) WEBSERVERS

web-2 ipconfig1 (10.0.0.5) WEBSERVERS

web-3 ipconfig1 (10.0.0.6) WEBSERVERS

**SSH into Jumpbox**

ssh [azureuser@23.99.190.121](mailto:azureuser@23.99.190.121)

List all containers created on the JumpBox

docker container list -a

Container name will be at the end of the screen display

docker container start <container name>

docker passwords

Activate a shell on the container

docker attach <container name>

**Edit the hosts file**

nano /etc/ansible/hosts

“ctrl w” type “webservers”, enter to search.

Add Web-3 servers IP Addresses and ansible\_python\_interpreter=/usr/bin/python3

#

# List the IP Addresses of your webservers

# You should have at least 2 IP addresses

[webservers]

10.0.0.4 ansible\_python\_interpreter=/usr/bin/python3

10.0.0.5 ansible\_python\_interpreter=/usr/bin/python3

10.0.0.6 ansible\_python\_interpreter=/usr/bin/python3

**Ping the web servers from the JumpBox container**

‘ansible all -m ping’ (Ignore ‘[DEPRECIATION WARNINGS]’)

**Run the ansible playbook to configure the new Web-3 VM**

ansible-playbook pentest.yml

**Creating the ELK VM**

**Create - Virtual Network**

Search – Virtual Network

Click add

Resource Group – click down arrow – select “Resource Group Created”

Instance Detail – “ELK-NET

Region – select region – (Select a Different Region ‘Azure Limitations’

IP Addresses

IPv4 Address Space: 10.2.0.0 /16

Next until “Validation Passed”

Click “Create”

**Create Peer link between vNets:**

Select: ELK-NET

Settings: Peerings

Click “Add”

Name: ELK-to-RED

Virtual Network: Red-Team vNet

Connection Name: Red-to-ELK

**Add Peering: ELK-Net**

Name of Perring ELK-Net to RedNet: ELK-to-RED

Vitual Deployment Model: Resource Manager

Subscription: Azure Subscription 1

Virtual Network: Rednet (Red-Team)

Name of Peering RedNet to ELK

Allow VN access from ELKNet to RedNet: Enabled

Allow VN access from RedNet to ELKNet: Enabled

Allow Forwarded Traffic from ELKNet to RedNet: Disabled

Allow Forwarded Traffic from RedNet to ELKNet: Disabled

**Create A New VM**

Click on Search and select Virtual Machine

Click Add

Subscription: Default

Resource Group: Group you created

Virtual Machine Name: ELK-Server

Region: Place in a different Region

Availability Options: No Infrastructure

Image: Ubuntu Server 18.04 LTS

Admin Account

Auth Type: SSH Public Key

User Name: azureuser

SSH Public Key Source: Use existing public key

SSH Public Key: Paste the public key generated from the ansible container

Inbound Rules: Allow selected ports

Select in bound Ports: SSH (22)

next

Disc Size: Default

Disc Type: Standard HDD

Encryption: Default

Network

Virtual Network: ELK-Net

Subnet: Default

Public IP: (new) ELK-Server-IP

NIC Network Security Group: Basic

Public inbound ports: Allow Selected

Select Inbound Ports: SSH (22)

Accelerated Network: Off

Load Balancing: No

Review and Create

**Downloading and Configuring the ELK Container**

Add VM to the ansible hosts file

SSH into the JumpBox

Connect to the ansible container.

Edit the hosts file:

nano /etc/ansible/hosts

# List the IP address of your ELK server

# There should only be one IP address

[elkservers]

10.1.0.4 ansible\_python\_interpreter=/usr/bin/python3

save and exit

Edit the ELK YAML Playbook

nano /etc/ansible/install-elk.yml

---

- name: Configure Elk VM with Docker

hosts: elkservers

remote\_user: elk

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

Save and exit

SSH from the ansible container to the ELK Server to ensure connectivity

Exit from the ELK Server

install docker on the ELK Server.

ansible-playbook install-elk.yml

SSH from the ansible container to the ELK Server

sudo docker ps

Identity and Access Management:

Verify the ELK Stack Server can be loaded.

Launch a web browser;

<http://[elk> server ip] :5601/app/kibana