

Cybersecurity Training Project

Created by: Anthony Latoni

January 2025

Overview

Gain hands on experience using practical cybersecurity tools. The operating system I will be using is proxmox, where I will be able to create dedicated virtual machines (VM's) and work on server management. I will be creating a contained network using a pfsense firewall using an .iso on a dedicated VM. I will create a segmented network using multiple virtual local area networks (VLAN's) in which each VM could reside on a separate network.

Operating Systems (OS) and security tools being used:

- Kali Linux: A Debian-based Linux distribution designed for penetration testing and security auditing. This will be our main machine
- Ubuntu Linux Desktop: A user-friendly Linux distribution suitable for personal computing and general-purpose use.
- Ubuntu Linux Server: A robust, open-source Linux distribution designed for server environments and enterprise use.
- Windows Active Directory: A directory service for managing and organizing network resources and user authentication in Windows-based networks.
- Windows 10: A personal desktop operating system by Microsoft, known for its user-friendly interface and advanced features.
- Windows 11: The latest version of Microsoft's operating system, offering enhanced productivity features, performance improvements, and a new user interface.
- Docker.io: A platform for developing, shipping, and running applications in lightweight containers, ensuring consistency across environments.
- Portainer.io: A management UI for Docker environments that simplifies container management and deployment.
- Nessus Vulnerability Scanner: A widely used tool for identifying vulnerabilities, misconfigurations, and security issues in systems and networks.
- Security Onion: A free and open-source Linux distribution for intrusion detection, network security monitoring, and log management.

- The Hive: An open-source Security Incident Response Platform (SIRP) designed for managing and responding to cybersecurity incidents.
- Cortex: An open-source platform for security operations automation, focusing on analyzing and responding to security incidents.
- Wazuh: An open-source security monitoring platform that provides intrusion detection, vulnerability detection, and log analysis.
- Metasploit: A penetration testing framework used to find, exploit, and validate vulnerabilities in systems and networks.
- bWAPP: A deliberately insecure web application designed to help security professionals practice exploiting web vulnerabilities.
- VULNHUB: A platform offering downloadable virtual machines with intentionally vulnerable configurations for security learning and testing.
- DVWA: The Damn Vulnerable Web Application, a PHP/MySQL web application designed for practicing web security skills and penetration testing.

Potential Cloud Environment's to Use:

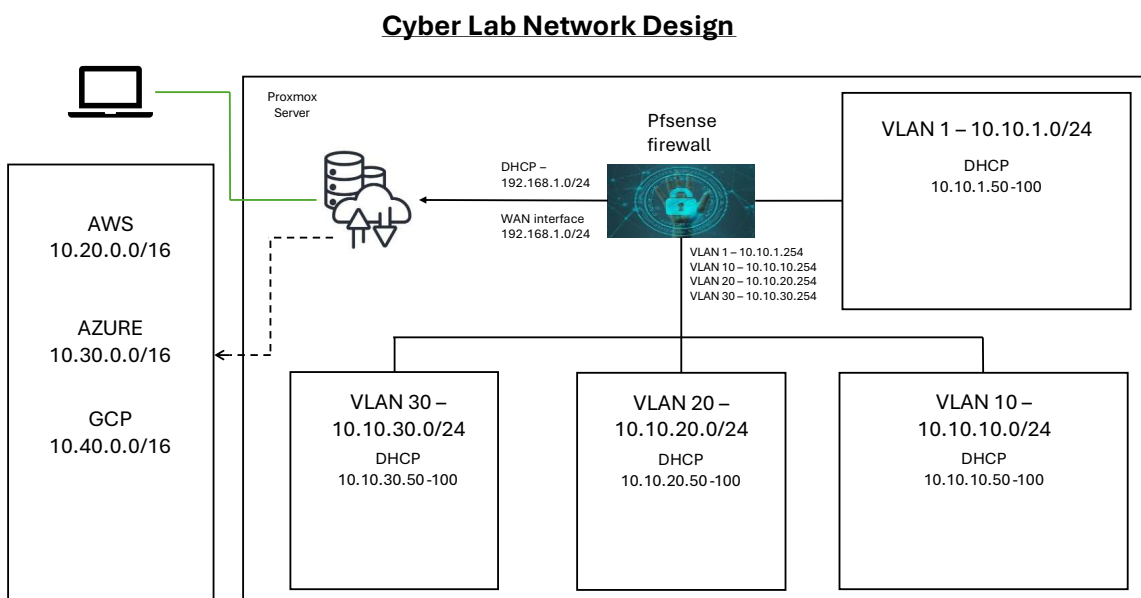
- AWS (Amazon Web Services): A comprehensive cloud computing platform offering a wide range of on-demand services for computing, storage, networking, and more.
- Azure: Microsoft's cloud computing service, providing a variety of cloud solutions for building, deploying, and managing applications and services.
- Google Cloud: Google's cloud platform offering scalable infrastructure, machine learning, and data analytics services for businesses and developers.

Network Design

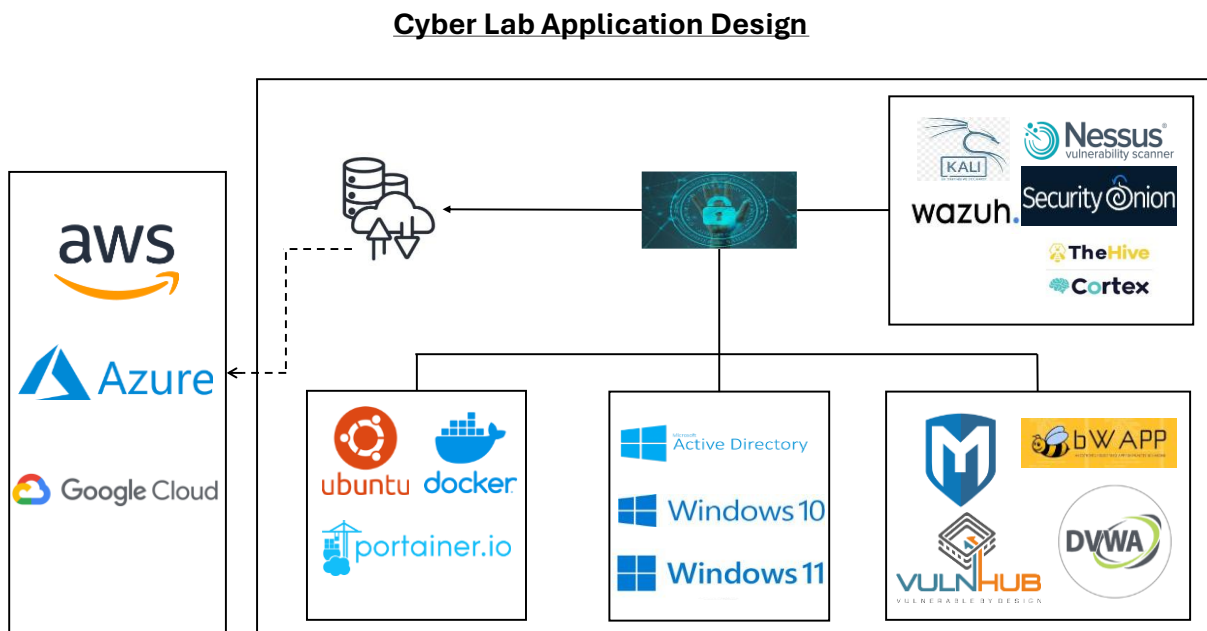
Using a proxmox I am able to create a bridge network dedicated for my contained environment. Using the pfsense firewall I will be able to segment this network into 4 VLAN's. Our WAN interface will be as normal, allowing us to reach out to the internet, while our LAN interface will create this segmented and separate network allowing us to perform our tasks without affecting our home network. All networks will have DHCP and we will have 4 VLANS:

- VLAN1: 10.10.1.0/24
 - DHCP: 10.10.1.50-100
- VLAN10: 10.10.10.0/24
 - DHCP: 10.10.10.50-100
- VLAN20: 10.10.20.0/24
 - DHCP: 10.10.30.50-100
- VLAN30: 10.10.30.0/24
 - DHCP: 10.10.30.50-100

In order to access this server remotely there will be a wireguard vpn set up. Below is a picture of the schematic of the network.



Based on the network schematic, the below picture shows all applications to be used and their dedicated VLAN's they will be a part of.




Phase 1

The first phase involved installing proxmox onto a server, set up an ubuntu server VM to host wireguard in a docker container using portainer, and then create our containerized network. After the previous steps are complete, we can move onto setting up our first three VM's, kali linux, ubuntu server, and docker.

Proxmox:

A proxmox .iso can be obtained from their website as it is an opensource platform at the following URL, <https://www.proxmox.com/en/downloads>.


Proxmox VE 8.3 ISO Installer			
	Version	File Size	Last Updated
	8.3-1	1.45 GB	November 21, 2024
	SHA256SUM b5c2d10d6492d2d763e648bc8562d0f77a90c39fac3a664e676e795735198b45		
			Download
			Torrent

You can use an application such as Rufus.exe to take the iso and load it onto a USB and create a bootable USB. The server should have a SSD/m.2 drive dedicated to the OS and booting and there should be separate HDDs for storage. The OS will be installed on the SSD or m.2, for the rest of this document our boot drive will be an SSD. Once the installation has completed, we will access the server from the IP address given at the end of installation. You will create a password to log into your server and we will need to initialize our disks. If you have more than one disk you can create a pool for redundancy. I have three, 4TB drives that will allow me to access 8TB of storage and have one hotspare.

Ubuntu Server:

We can now start creating our virtual machines. I began with an ubuntu server that can access my home network outside of our containerized network so I can deploy wireguard. This will allow me to access my server remotely and continue working when I'm not on my home network. With the ubuntu server I dedicated 1 core, 2gb or RAM, and 10gb of storage for this machine. I downloaded the ubuntu server .iso from there website found at URL,

<https://ubuntu.com/download/server>.



Ubuntu 24.04.1 LTS


The latest LTS version of Ubuntu Server. LTS stands for long-term support — which means five years of free security and maintenance updates, extended to 10 years with [Ubuntu Pro](#).


[Download 24.04.1 LTS](#) 2.6GB


[Alternative downloads >](#)


[Alternative architectures >](#)


We need to upload this .iso to our server in order to load it. That is done by going to the local storage pool, and clicking on ISO images, and the upload.


 local (pve)


 local-lvm (pve)


 pool1 (pve)

 Summary

 Backups

 ISO Images

 CT Templates

 Permissions

[Upload](#) [Download from URL](#) | [Remove](#)

Name
kali-linux-2024.3-installer-amd64.iso
netgate-installer-v1.0-RC-amd64-20240919-1435.iso
ubuntu-24.04.1-desktop-amd64.iso
ubuntu-24.04.1-live-server-amd64.iso

As seen, I have multiple images uploaded for this first part. We can then click on create VM and then we can enter in an id, where I used 100 for my first machine, select the ubuntu server .iso and set the specs stated previously.



This is what the button will look like on your screen. Once the settings are entered you can start the VM and enter the console. Continue through the graphical install and once the ubuntu server has finished installation we can go ahead and begin getting docker installed.

Docker:

I headed over to the documentation at URL, <https://docs.docker.com/engine/install/ubuntu/>.

When you scroll down you will find a series of commands. Before we can run these commands, we need to access our new server. You can do so by using terminal or putty in order to ssh into your server. The address will be given at the end of the server's installation. Since it is on your home network it should be something similar to 192.168.1.xxx. Once you SSH and log in, you will be able to enter these series of commands:

(if the commands encounter an error type "sudo" before each line)

1. for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove \$pkg; done
2. # Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

Add the repository to Apt sources:
echo \
"deb [arch=\$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

```
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update
```

3. `sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`
4. `sudo docker run hello-world`

You should now see a long text with a line stating something similar to “hello world your container is now running”. To simplify docker further we will install portainer.io on top of it to give it a graphical interface.


Portainer.io:

To install portainer.io we can follow the docs at this URL, <https://docs.portainer.io/start/install-ce/server/docker/linux>. We will be installing this on the same server, so continue to be SSH'd into it. Again, the following commands will be entered in order, and if there is an error type “sudo” before each command:

1. `docker volume create portainer_data`
2. `docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:2.21.4`

THAT’S IT, so simple. Successfully installed portainer. You are able to type “docker ps” and you will see you portainer container active.

To log into portainer we will need to enter in the address bar on our local machine followed by the port. It will look something like <https://192.168.1.xxx:9443>. We will be taken to our login page where we can set our password.



Log in to your account

Welcome back! Please enter your details

Username


Enter your username

Password

Enter your password

Login

We can press live connect to go into our local server, click on containers, and it is here you will see our hello world container and portainer. The hello world container can be deleted.



local Up 2024-12-02 17:43:47

Standalone 27.3.1 /var/run/docker.sock

Group: Unassigned No tags Local

0 stacks


2 containers 2 0 0 0

1 volume 3 images 2 CPU

2.1 GB RAM

Live connect

Disconnected



2

Containers

2 running
0 stopped
0 healthy
0 unhealthy

Now we can move on to installing wireguard!

Wireguard VPN:

On the top right we are going to deploy our container. There is documentation on this found at this URL, <https://docs.linuxserver.io/images/docker-wireguard/#site-to-site-vpn>.

We will follow the docker CLI inputs for our portainer.

You can name your container and under image we will use, lscr.io/linuxserver/wireguard:latest, which will grab the latest image when it launches. Continue to enter under variables the puid and the pgid of the user on your server, normally it is set to 1000. Set a number for peers, this will be how many connections you want to make and this can always be updated later. SSH back into the linux server on the side and set a path so that way you have a wireguard directory. If you are on the node (~) you can enter pwd and should see something like /home/user. This is where we want to enter mkdir wireguard to create that directory and we can cd into wireguard and enter pwd to see /home/user/wireguard. Under volumes we will use that path for the volume and the path for container will be /config. And there will be a second volume with path to host and container being the same at /lib/modules. For the port we will set that to UDP 51820 on incoming and outgoing, this is our safest option. Here is the rest of the documentation to try and research and try to get the right entries:

```
docker run -d \  
  --name=wireguard \  
  --cap-add=NET_ADMIN \  
  --cap-add=SYS_MODULE `#optional` \  
  -e PUID=1000 \  
  -e PGID=1000 \  
  -e TZ=Etc/UTC \  

```

```
-e SERVERURL=wireguard.domain.com `#optional` \  
-e SERVERPORT=51820 `#optional` \  
-e PEERS=1 `#optional` \  
-e PEERDNS=auto `#optional` \  
-e INTERNAL_SUBNET=10.13.13.0 `#optional` \  
-e ALLOWEDIPS=0.0.0.0/0 `#optional` \  
-e PERSISTENTKEEPALIVE_PEERS= `#optional` \  
-e LOG_CONFS=true `#optional` \  
-p 51820:51820/udp \  
-v /path/to/wireguard/config:/config \  
-v /lib/modules:/lib/modules `#optional` \  
--sysctl="net.ipv4.conf.all.src_valid_mark=1" \  
--restart unless-stopped \  
  
lscr.io/linuxserver/wireguard:latest
```

Once you are confident this is complete you can deploy the container and you will then need to port forward. I logged into my home router and created a rule to port forward from the server IP over port 51820. This is how you will get back into your home network. You can grab your config files for the peers off of the server in the directory that you created for wireguard. Once you have those files head on over to wireguard and download the program on their website to your laptop or mobile device and you can upload your config file. Once complete make sure you are not on your home Wi-Fi and try to connect to your server. All finished!

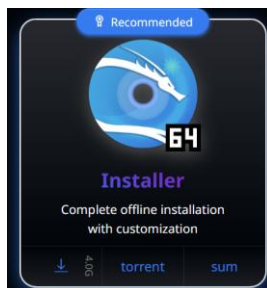
NordVPN [UPDATE]:

When outside of my house on data wireguard was functioning properly. When I was outside of my home but on another wifi router, I was experiencing double NAT conflicts and DNS conflicts so I wasn't able to resolve my IP address of the server at home, but I was able to surf the web. I had migrated my system still on the ubuntu server and used a series of linux commands to install NordVPN. With these commands I was able to set up my server as the "location" point to connect to and essentially vpn'd back to my home network and use the server as access to the local network and route throughout my network. The commands I used were:

- **sh <(curl -sSf <https://downloads.nordcdn.com/apps/linux/install.sh>)**
- **sudo nordvpn login --apikey**
- **sudo nordvpn set meshnet on**
- **sudo nordvpn meshnet peers local devicename**
- **sudo nordvpn meshnet peers routing devicename**

Based on the previous steps I can now VPN back into my home network so I can continue to work! Continuing on with the rest of phase one I created the firewall, a kali machine, and another ubuntu server for docker and portainer. Let us begin by uploading our missing .ios files. I needed to upload a Kali Linux .iso and pfsense .iso both found at the following URLs:

- Kali: <https://www.kali.org/get-kali/#kali-installer-images>



- PfSense: <https://www.pfsense.org/download/>



Phase 2

First, we will start with our firewall and Kali machine in order to access the firewall interface. Then we will install our docker instance and portainer instance the same way we previously did but on our new network and a new and separate ubuntu server VM. To begin the next steps, we need to check out network interface inside of proxmox. Currently I have one NIC port active and this port acts as our WAN. When proxmox was created it created a bridge interface called vmbr0 which acts as a network device for the server to access the web. I need to create a second interface by adding a new hardware device and this will be called vmbr1 and labeled LAB LAN. This second interface will act as the virtual LAN interface for our fw.

pfsense:

When I obtained the download image, I created a new machine titled prod-fw and an id of 201. I decided to start in the 200 range for all machines inside of my project network. I gave the fw 1 core with 4gb of RAM. When going through the install steps I began with setting the WAN and LAN devices. Enth0 was set to WAN which will grab a 192.168.1.xxx IP in order to talk to the internet and allow our LAN to reach the web as well. Enth1 was set to my LAN network where I gave that a static IP of 10.10.1.254. After the basic settings were enabled and the install was

finished, I was able to reach the dashboard, but first I needed to get my Kali machine up in order to access the interface since this is now under my contained network.

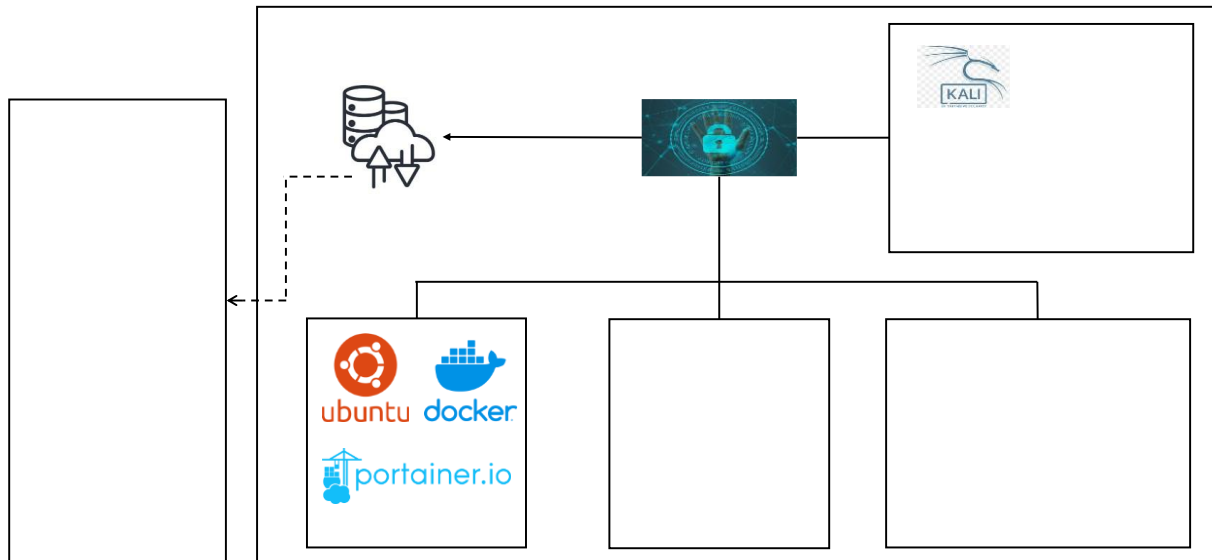
Kali:

When creating this machine, I gave it 2 cores and 8gb of RAM. Specifically for this device we do not need to set a VLAN id because this machine will be on our main VLAN1. Once the install finished, I was able to log into the machine and launch firefox where I can access the address of my firewall at 10.10.1.254. Once here I logged into the fw and changed the admin account default password for security. Based on my network schematic I configured my 3 other VLANs, VLAN10/20/30, and then made sure to enable dhcp and edit my IP ranges to match the schematic as well. Now that the fw is up and running I can set up the final ubuntu server machine for phase 1.

Ubuntu Server:

The ubuntu server will house my new docker instance and portainer as well. When setting up this machine I dedicated 2 cores and 2 sockets, and 8gb of RAM. Based on my network schematic this machine needs to fall into the VLAN30 range; therefore, I tagged the VLAN ID on setup to 30. This was confirmed successful at the end of install when the IP given for the server was 10.10.30.50. From my kali machine I was able to ping the ubuntu server to verify connect ability. After, I SSH'd into the server I continued to deploy my new instances of docker and then portainer same as the directions used before.

Cyber Lab: Phase 2



Phase 3

Within my docker machine I wanted to start to deploy some containers. The containers would include bwapp, dvwa, and webgoat. Along with those containers I also deployed a Metasploitable 2 VM. Beginning with the Metasploitable 2 machine I used the command, `wget http://downloads.metasploit.com/data/metasploitable/metasploitable-linux-2.0.0.zip`, to download the installer zip. I then used the command, `unzip metasploitable-linux-2.0.0.zip` to unzip the file to my images folder local to the proxmox storage.

Metasploitable 2:

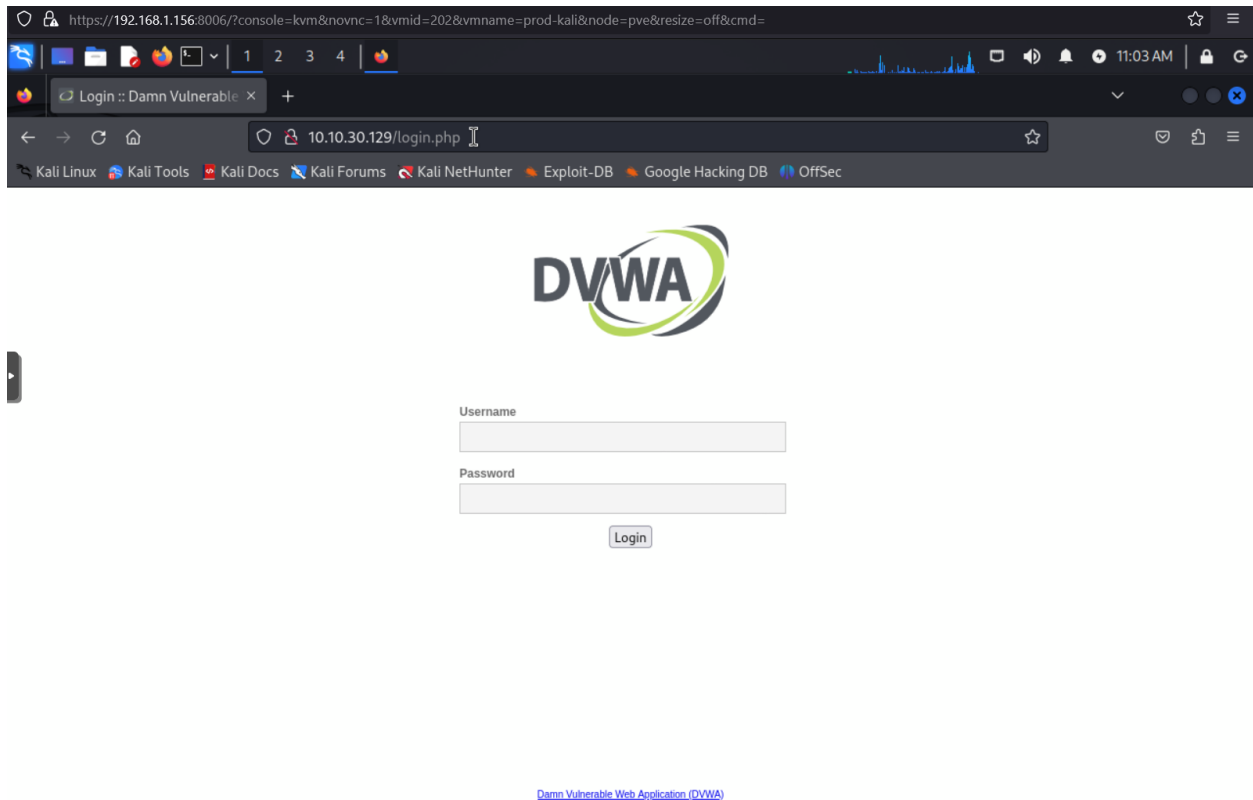
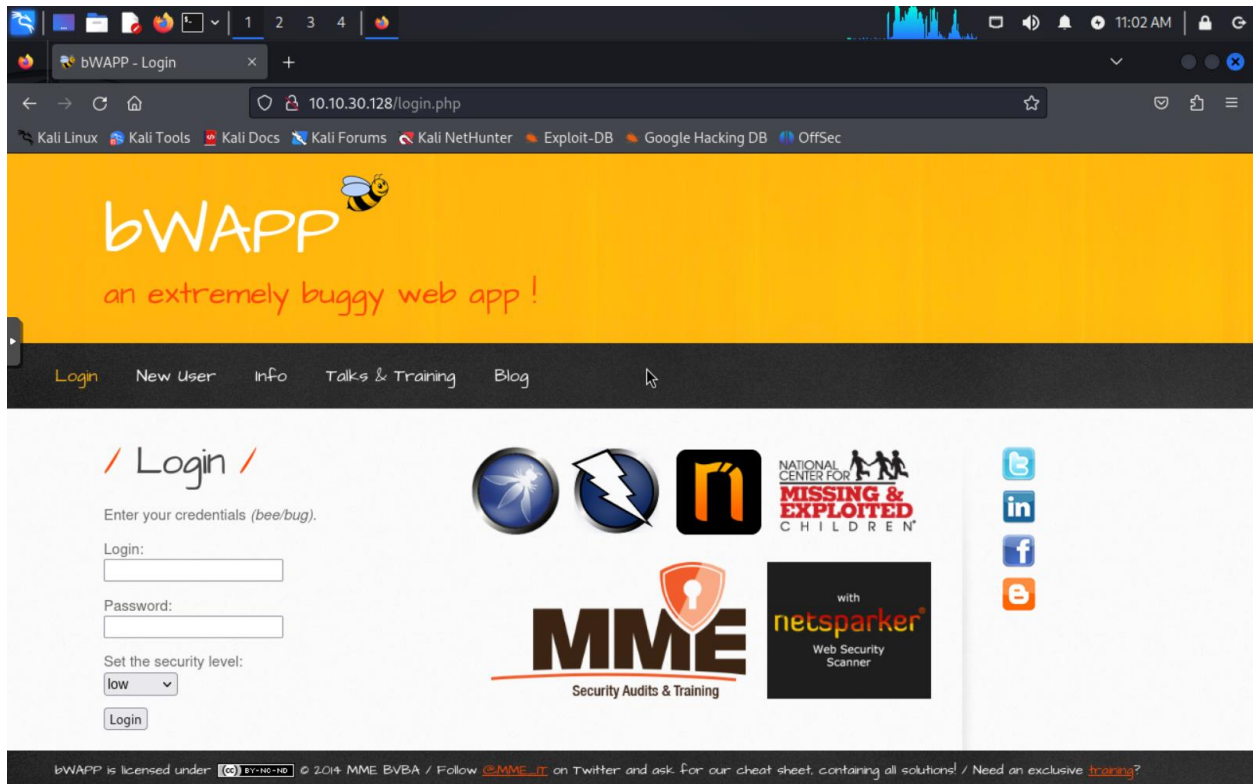
I had created an empty template machine that will later get this installer image, but kept in mind this machine needed to be on my VLAN10 so I made sure to tag that machine. This machine was Once it finished unzipping, I then converted the VMDK file to a QCOW2 file. The command was `qemu-img convert -f vmdk Metasploitable.vmdk -O qcow2 Metasploitable.qcow2`. Once we had the qcow2 file I then navigated over to the VM's 204.conf file and entered the location of our

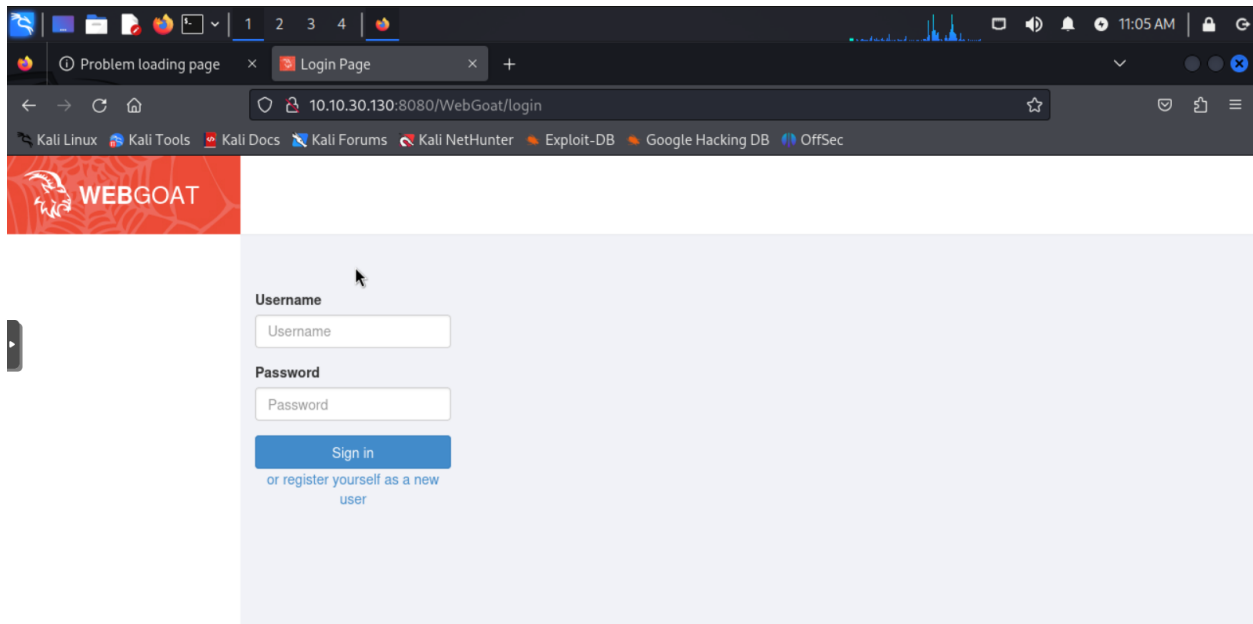
qcow2 image so that way once the VM started it grabbed this file as the installer image. Once the install finished, I used the command, ip address, to verify we were given the right subnet. I then went on my Kali machine and went over to the IP 10.10.10.50 which was the IP given to the metasploitable 2 machine where we reached the landing page. I then moved on to my three docker containers.

DVWA, bWapp, Webgoat:

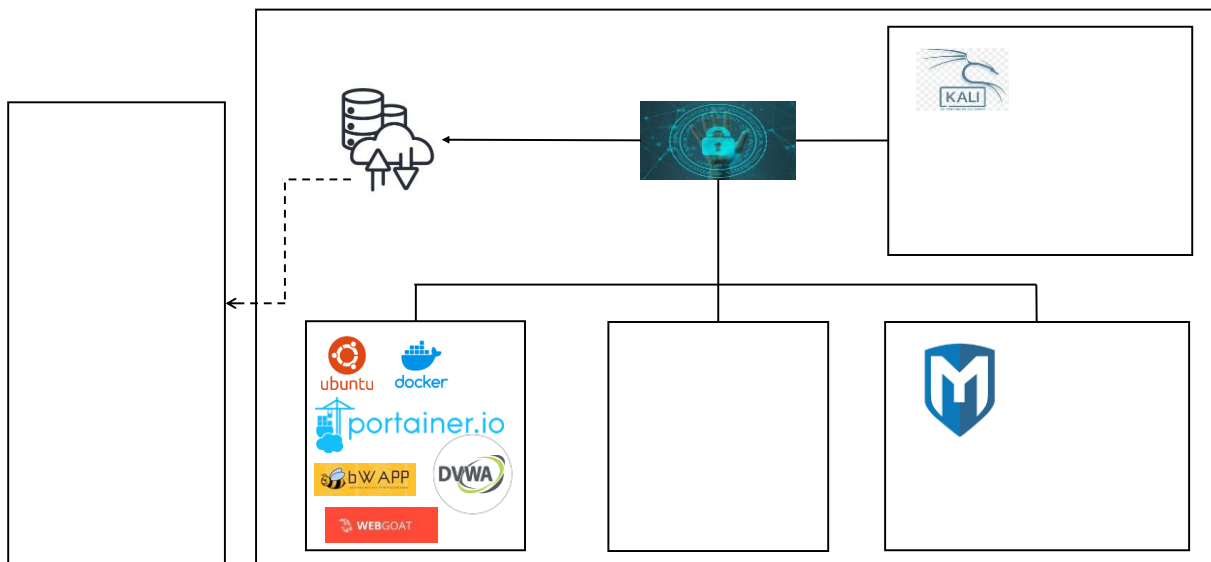
On my kali machine I went over to portainer and I began building my containers. Before deployment I created a network within portainer so each new container would grab a new IP address instead of using the same IP address and exposing different ports. Under network settings I added two networks; one network was a config file for the other network. The first network was titled VLAN30-config and under this my interface was a macvlan. Under this I used my main subnet of 10.10.30.0/24 and then segmented this network. I used an IP calculator online to see what a 10.10.30.150/27 would give me and it showed me that I would have access to addresses .128 - .158. I then created another network called VLAN30 as the creation VLAN that will distribute the addresses and used the config container that was just created as its base rule set. I then created the three separate containers using the image installer commands below and setting the network to our new VLAN30.

- vulnerables/web-dvwa
- raesene/bwapp
- webgoat/Webgoat





Cyber Lab: Phase 3



Phase 4

The next two machines brought into production were Wazuh and Nessus.

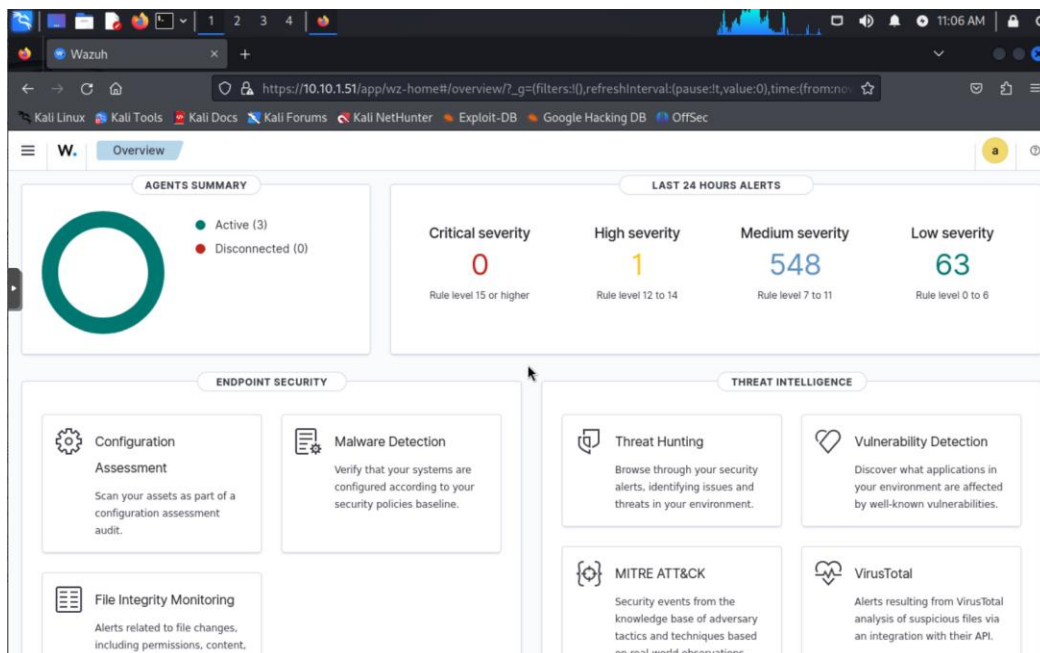
Wazuh.:

The Wazuh server was created using 100gb of storage because I wanted to be able to store more logs. The Wazuh server also holds 4gb of RAM and 4 CPU cores as well. Since this machine is on my VLAN1 I didn't need to tag the VLAN when setting up the VM. To deploy the server, I went onto my kali machine SSH'd into the new server. I needed to use the installer assistant and that was done with the following commands:

- `curl -sO https://packages.wazuh.com/4.9/wazuh-install.sh && sudo bash ./wazuh-install.sh -a`

The commands for the installer assistant were found at,

<https://documentation.wazuh.com/current/quickstart.html>. After the installer finished, I got a username and password. I went on my Kali machine and logged into the Wazuh dashboard using the server IP and the new username and password.



I then started deploying the agents to my devices. The devices that I installed the agent on so far was my pfSense firewall, Kali machine, and my Docker container machine. For the Linux machines I used the following commands to install the agent:

- `curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644 /usr/share/keyrings/wazuh.gpg`
- `echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list`
- `apt-get update`
- `WAZUH_MANAGER="IP OF SERVER" apt-get install wazuh-agent`
- `systemctl daemon-reload`
- `systemctl enable wazuh-agent`
- `systemctl start wazuh-agent`

Using the server IP will have the agent talk to the server manager and allow me to oversee changes on my servers. When the Docker agent was installed, I had to activate the Docker listener on the dashboard. First, I ran this command on my Docker server, “`pip3 install docker==7.1.0 urllib3==2.2.2 requests==2.32.2`”. Second, on the Wazuh dashboard by going to the settings I updated the config file to include:

```
<wodle name="docker-listener">
```

```
<disabled>no</disabled>
```

```
</wodle>
```

I then just had to save and click restart manager for the settings to apply. I then went over to the docker listener settings in the dashboard and made sure they turned on. When this was complete, I now had my Kali machine and Docker machines activated. I needed to move on to my pfsense firewall which was more task intensive. In order to get the agent on the firewall I needed to ssh into the root user of the firewall. Once in I ran the following commands:

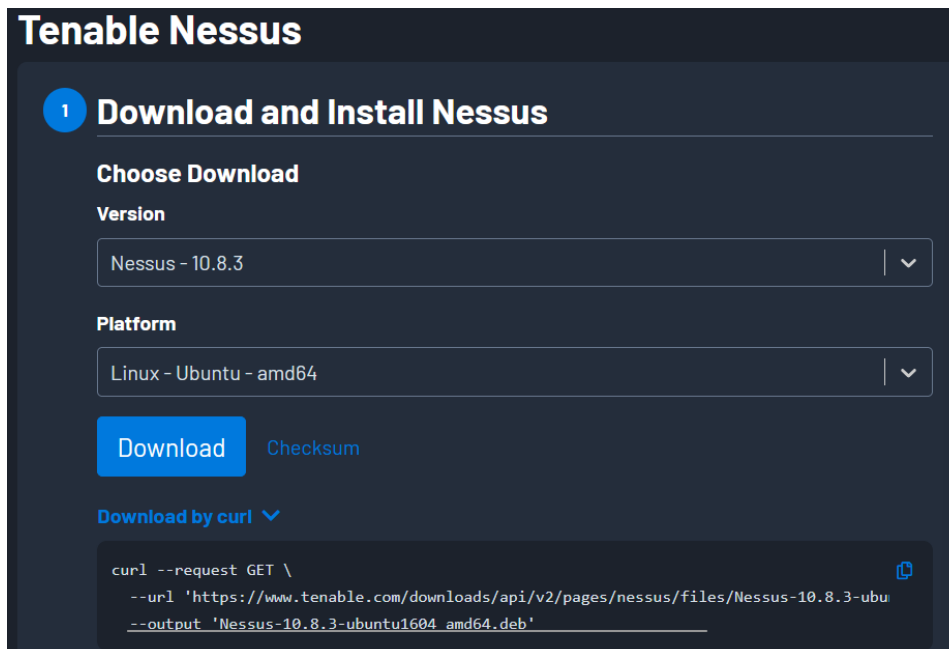
- `cd /usr/local/etc/pkg/repos/`
- `vi pfsense.conf`
- `FreeBSD: { enabled: yes }` (changed a line on the file and then used esc, :wq! To save the file)
- `vi FreeBSD.conf`
- `FreeBSD: { enabled: yes }` (changed a line on the file and then used esc, :wq! To save the file)
- `pkg update`
- `pkg search wazuh-agent`
- `pkg install wazuh-agent-VERSION-RETURNED-FROM-SEARCH`
- `cd /etc/localtime /var/ossec/etc`
- `vi ossec.conf`
- `<server>`
`<address>Address of server</address>`
`</server>`
- `sysrc wazuh_agent_enable="YES"`
- `ln -s /usr/local/etc/rc.d/wazuh-agent /usr/local/etc/rc.d/wazuh-agent.sh`
- `service wazuh-agent start`

I now had all three agents communicating with the Wazuh server manager. On the dashboard I initiated new firewall logs to see more into the pfsense. The custom rule is:

```
<group name="pfsense,">
  <rule id="87701" level="5" overwrite="yes">
    <if_sid>87700</if_sid>
    <action>block</action>
    <description>pfsense firewall drop event.</description>
    <group>firewall_block,pci_dss_1.4,gpg13_4.12,hipaa_164.312.a.1,nist_800_53_SC
  </rule>
</group>
```

Nessus:

The Nessus VM was given 4 CPU, 4gb of RAM, and 40GB of storage space. Since this machine is on my VLAN1 I didn't need to tag the VLAN when setting up the VM. At the tenable website found at this link, <https://www.tenable.com/downloads/nessus?loginAttempted=true>, I chose the most recent Nessus version and the Linux ubuntu amd64 install. I then clicked on the drop down to get the commands for "download by curl"



Tenable Nessus

1 Download and Install Nessus

Choose Download

Version

Nessus - 10.8.3

Platform

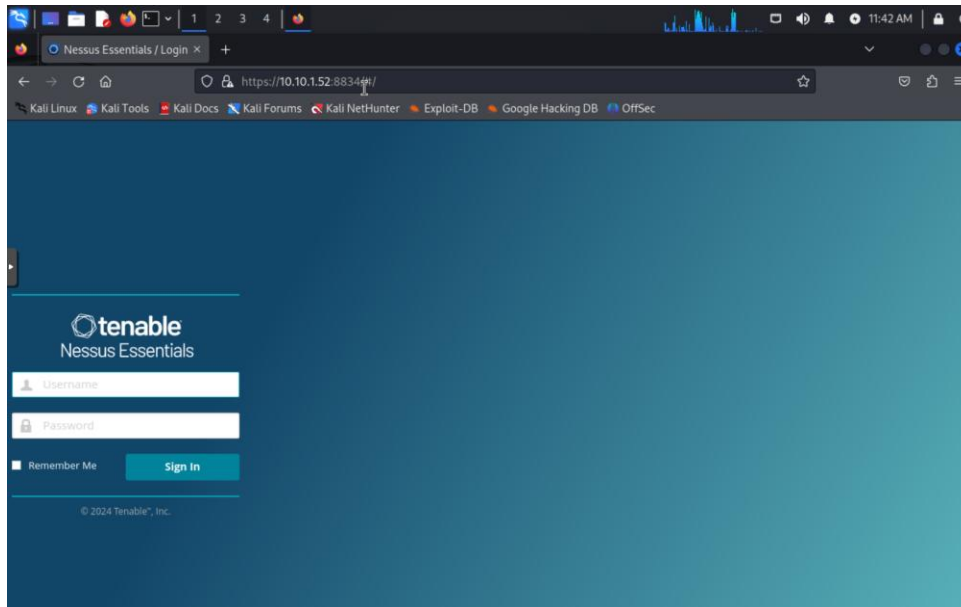
Linux - Ubuntu - amd64

Download [Checksum](#)

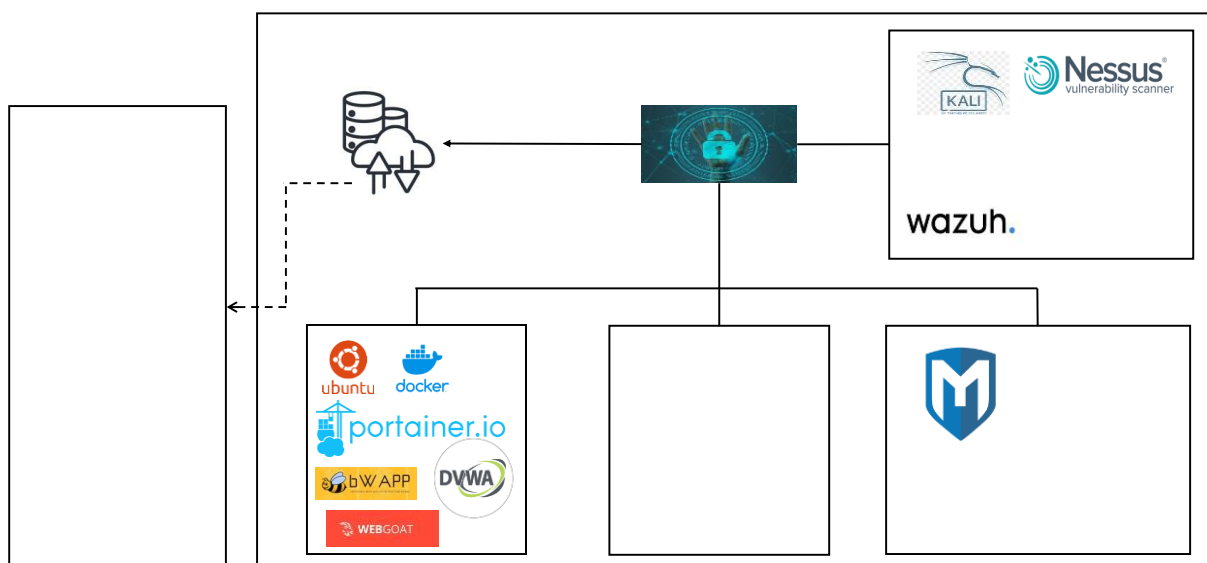
Download by curl

```
curl --request GET \
--url 'https://www.tenable.com/downloads/api/v2/pages/nessus/files/Nessus-10.8.3-ubu
--output 'Nessus-10.8.3-ubuntu1604_amd64.deb'
```

When entering the curl command, the download for Nessus began and I waited for that to finish. I entered the following command to start the Nessus dashboard, “systemctl start nessusd”. I then went over to my Kali machine and logged into the dashboard using the IP address of the server and port 8834. I then created an account using my email which gave my product and activation code and let the plugins begin to install to the server.



Cyber Lab: Phase 4



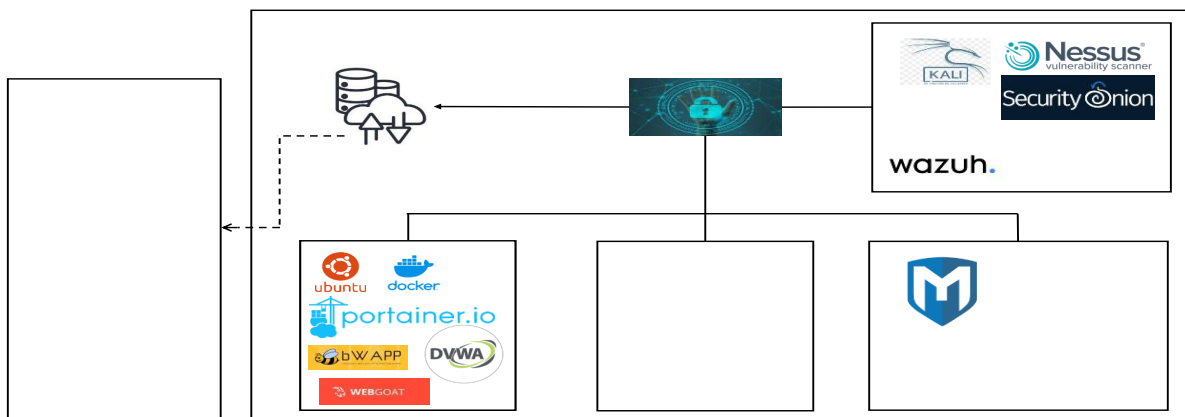
Phase 5

Security Onion:

The Security Onion VM is an intensive standalone program and was given 4 CPU, 16gb of RAM, and 200gb of storage. That iso was downloaded and then uploaded to my proxmox server and set up on my default VLAN so it will not need any tag. Once the iso was installed with the basic steps of accepting everything in the install, I was able to reach the dashboard from my kali machine.



Cyber Lab: Phase 5

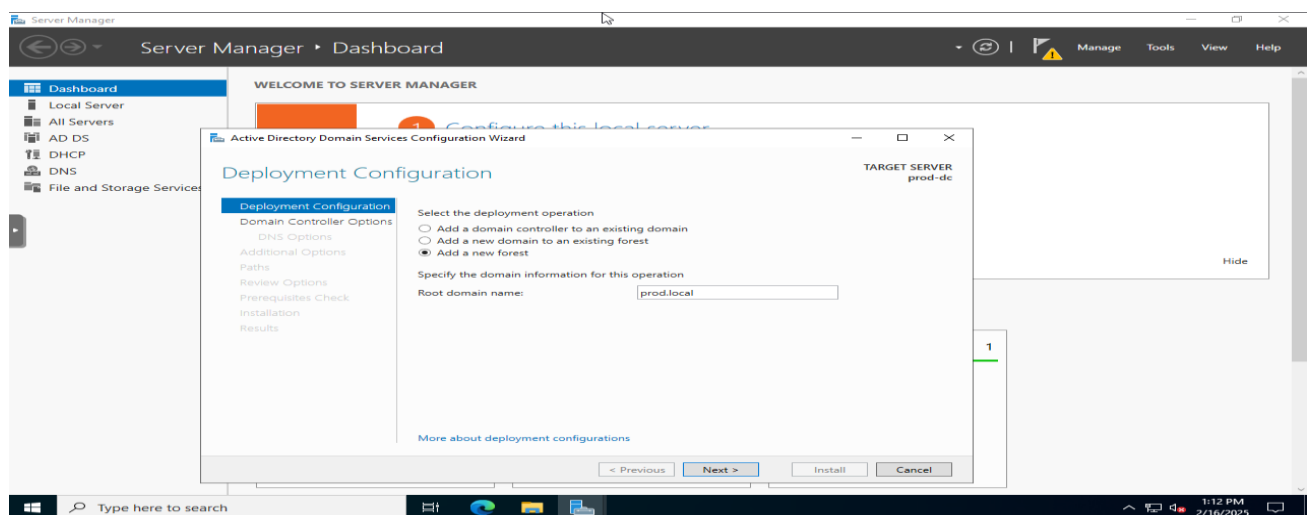


Phase 6

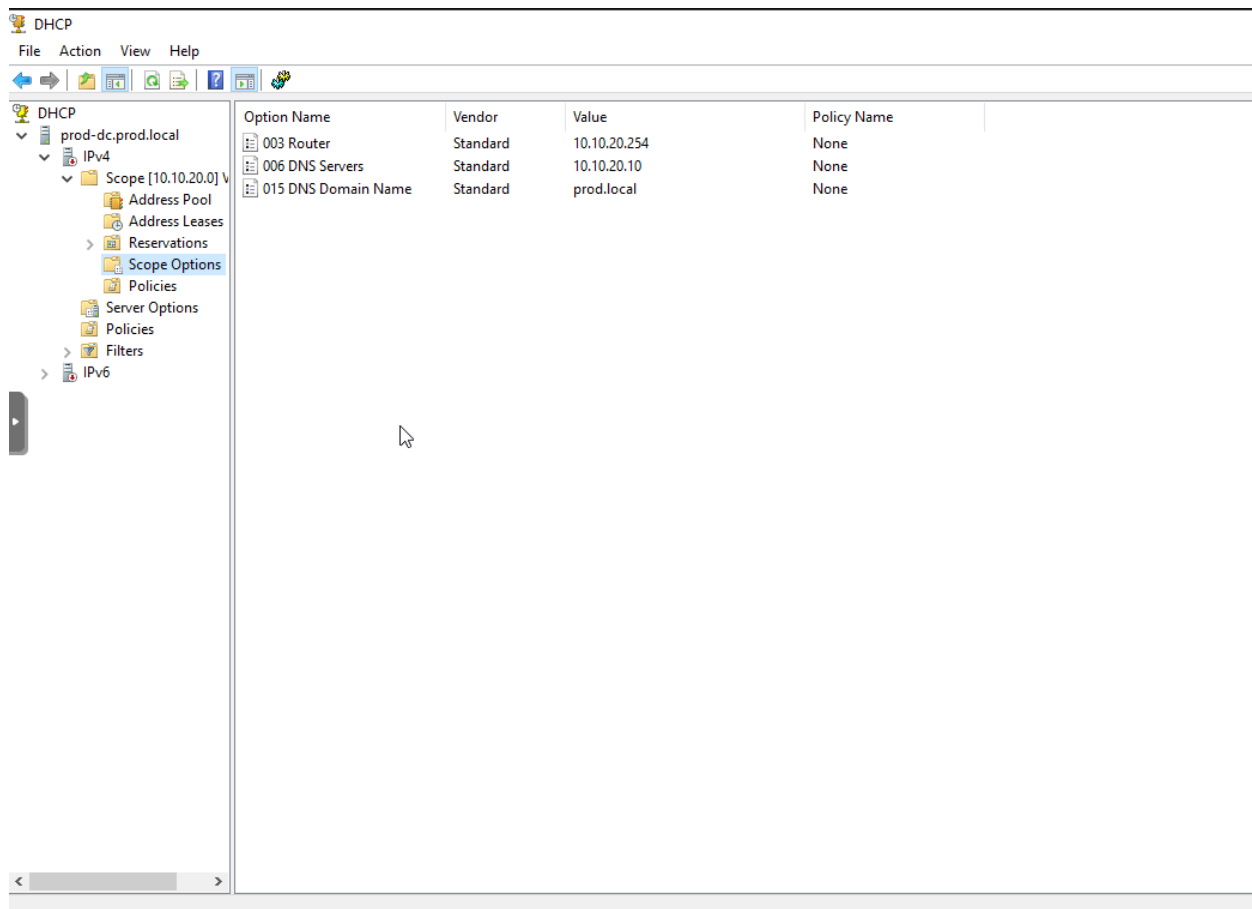
Windows Server 2022:

The windows server is going to be set up as a domain controller for new windows machines to join. To begin I needed to go and get an evaluation version of the server iso from Microsoft. This will allow for 180 days of a continuous trial to continue my project. The iso can be found here, <https://www.microsoft.com/en-gb/evalcenter/evaluate-windows-server-2022>. I also needed to get Windows VirtIO Drivers in order for proxmox to work properly with the windows installation. That iso can be found here, https://pve.proxmox.com/wiki/Windows_VirtIO_Drivers.

When in proxmox I was able to successfully load the two iso's to the new vm. When trying to install the windows server it will not show any drives for where to install... you will need to browse files and choose the virtio file, drop down to scsi, and you will see amd64. When selected the driver for red hat will appear and you can install and then see the drive to install server 2022. When installed you can sign in and I have chosen to add roles and features for active directory services, dhcp, and dns. When deploying the domain controller we will want to make this our primary controller and create a new forest like this:



On the next screen you are able to set any password you want and then click next until you are on the install page. It will do prerequisite checks and then you are able to perform the install. Once complete the computer will restart. I then went into the active directory users and created a user for myself and an admin user also for myself. After I then created a group called sharedaccess for users to have access to a shared folder. After creating the local users and groups I went over to my pfSense firewall and I deactivated DHCP for my VLAN20 to move it over to the WIN 2022 server. After it was deactivated, I went onto the DHCP application of the server and I right clicked on the IPv4 to set up a new scope and enter all proper information to begin a new range and have an active DHCP for when clients join the domain.



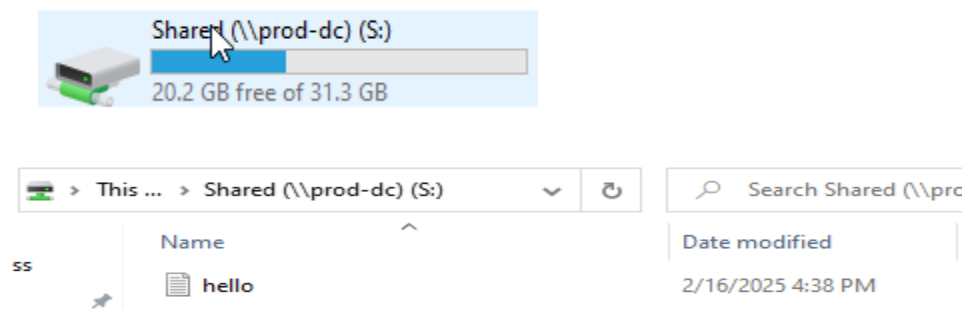
After all is done, I then will create a folder on the C drive and give access to read and write to the users and have the GPO set the folder to automatically map for the users. I made a new group

policy and targeted it to my shared folder access security group so when a user is apart of that group, when they log in they will have the shared folder mapped to their sdrive automatically. Since the domain controller is configured, we can move on to installing a Win10 PC and connect it to out domain with our user account we created prior.

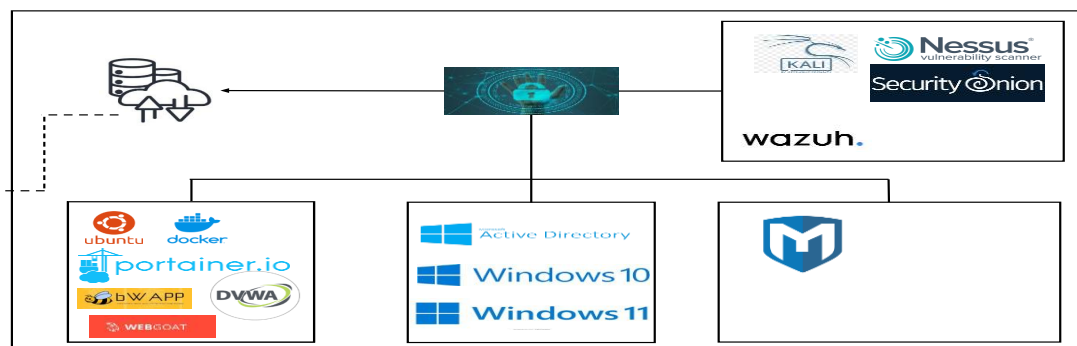
Windows 10:

The windows pc is going to be installed similarly to the serve. You will need to add both .iso files and be able to install the driver the same way so we can find our install location just how I did with the 2022 server. Now that the install for win 10 has completed I am able to check that the pc is getting its IP from the new DHCP server, and it is. I will then go to settings and join the domain using my admin account. After joining the domain, I am able to restart the computer and sign in as my user account. After signing in I confirmed that the shared folder is established and that I am able to see the test document that had been added.

Network locations (1)



Cyber Lab: Phase 6



Phase 7

The Hive and Cortex:

Using the previously installed docker instance I am able to go to docs.stangebee.com/thehive and follow the docker deployment information to deploy the hive as a docker container. I first cloned the repository using: `git clone https://github.com/StrangeBeeCorp/docker.git`. I then ran the initialization script with command `bash ./scripts/init.sh` inside the testing directory to use the testing hive application. This script will install all application image pre requisites. After that I ran `sudo docker compose up` which then installed thehive, nginx, Cassandra, cortex, and elastic search. After some configuration:

1. Incident Creation

- A security event (e.g., suspicious network traffic from Pi-hole logs) triggers a case in TheHive.

2. Observable Extraction

- TheHive extracts observables (IPs, domains, hashes, emails) from the case.

3. Automated Analysis with Cortex

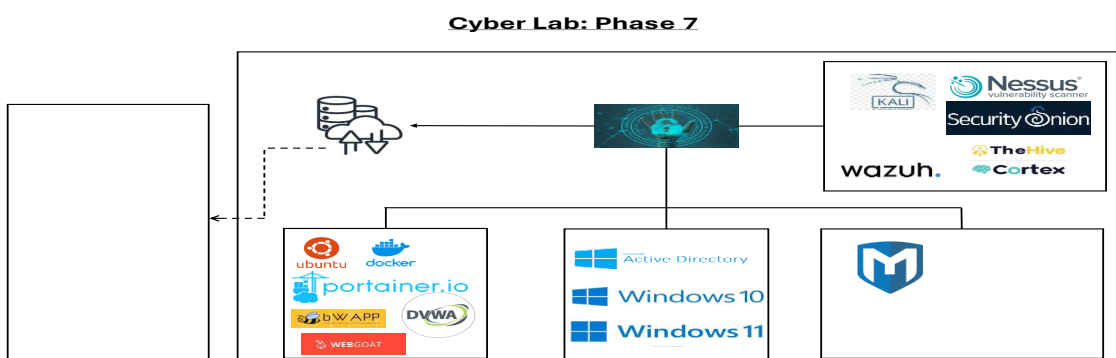
- TheHive sends the observables to Cortex for analysis.
- Cortex runs configured analyzers (e.g., checking if an IP is blacklisted).

4. Enrichment & Investigation

- Cortex returns the results to TheHive.
- You review findings, correlate data, and decide on next steps.

5. Response Actions

- If a threat is confirmed, Cortex can automatically:
 - Block the IP in your firewall.
 - Notify via email or webhook.
 - Log findings into other security tools.



Phase 8

Azure:

Using azure I create a v-net there we'll create a site-to-site VPN between our pfsense firewall and the v-net within Azure that way we can expand to the cloud. In Azure, I have an account and a subscription where I'll create a resource group to house all necessary components. Within this group, I'll set up a VNet with the 10.30.0.0/16 address space, including two subnets: a gateway subnet (10.30.0.0/27) for the Azure VPN Gateway and a VM subnet (10.30.1.0/24) for Windows VMs. To establish the site-to-site VPN, I also need to configure a Local Network Gateway representing my on-prem network. My home network runs on FiOS with pfSense acting as an internal firewall behind my main FiOS firewall, requiring port forwarding to allow VPN traffic. Since my ISP provides a dynamic public IP, I'll use DuckDNS to ensure consistent connectivity. First, we'll set up Azure by creating a resource group, VNet, and subnets. Next, we'll deploy the VNet Gateway and Local Network Gateway. Then, on my external firewall, I'll configure port forwarding for UDP 500 and UDP 4500 and assign a fixed IP to the WAN interface. After that, we'll return to Azure to create the site-to-site VPN connection. Next, we'll configure Dynamic DNS with DuckDNS, obtaining the necessary domain and token. Then, on pfSense, we'll set up Dynamic DNS and site-to-site VPN firewall rules. Finally, we'll go back to Azure and deploy a VM to test connectivity.

Go to azure Microsoft.com and create an azure account where you can then log in and go to resource group and create. I called the tab lab prod in East US time zone because I am in New York, but you can use whatever time zone you are in.

- **Go to Azure:** Visit [Azure Portal](#) and sign in or create a new account.

- **Create a Resource Group:**

- In the **Azure Portal**, search for "**Resource groups**" in the top search bar.
- Click "**Create**" and enter a **Resource Group Name** (e.g., `lab-prod`).
- Choose a **Region** (e.g., **East US** for New York or your preferred region).
- Click "**Review + Create**" and then "**Create**" to finalize.

Create a new virtual network by searching virtual network and then click create. Select your subscription and the resource group we just created. I am going to call my network lab-vnet. The address space will be the 10.30.0.0/16 and then modify the subnet starting address to be 1030.1.0/24. Click next and finalize the creation of the network. Within the lab-vnet, you will click on settings and then subnets. Here is where I was able to add the additional subnet for my gateway. Click +Gateway subnet and save. This will add the default subnet. We can the search virtual network gateway. This gateway will cost money so it will be the bear minimum for testing. The gateway type will be VPN, SKU will be VpnGw1, generation 1, and select our vnet for the virtual network. For the public IP area, we need to create 2 new public IP's so I named them gw-public 1 and 2. I will then review and create.

While the v-net is deploying I logged into my pfsense firewall to grab the public IP address currently being used. Once I grab those numbers, I went over to my FIOS firewall so that I can port forward UDP 500 and 4500 to my pfsense firewall.

I used duck DNS to handle the changing of the dynamic public IP. By going on my Kali machine and to duck DNS I can log in using an account, and the set up a subdomain to point my pfsense firewall to that IP that will be given. I clicked the install tab and then selected pfsense as my operating application. I then logged into my pfsense firewall and went to services and selected dynamic DNS. I selected add and then selected custom and add the update URL given from duck DNS. This gives the domain automatically and the token needed to access it.

The v-net finished after about 40 minutes. I went back to azure and searched local network gateway and selected create. I selected the already created resource group, named the gateway local gw prod, and then selected my endpoint as an FQDN where I can enter the ip address I got from the duck DNS domain. Under address spaces I entered all subnets that are created based on the network configuration from the diagram at the top of this document.

Under virtual network gateway I went to my gateway lab and selected connections. This is where I set up the site-to-site connection and named it VPN to home lab. The virtual and local networks will only have one choice to select as I just created them and then I made my shared key for security. I then clicked create to finish the site-to-site VPN. Under configuration of the VPN, I set my IKE Phase 1 encryption to AES256, integrity to SHA256, and DH Group to DHGroup14. For IKE Phase 2 I set encryption and integrity to AES256 and SHA256, respectively. These settings will be paralleled onto the pfsense firewall.

On pfsense, under VPN, I clicked IPsec, and added p1. I called it azure VPN and set my key exchange to auto. For remote gateway I filled in the azure public IP from the gateway. The identifier is set to the fully qualified domain name that was set on duck DNS. The pre shared key was set on azure so use the same on pfSense. For encryption I paralleled Azure and hit save. I then added p2 and set up the local network to be 10.10.0.0/16 which will cover all the subnets. I then added what the remote network from azures subnet is, which was 10.30.0.0/16. I copied the same encryption again and then was able to save.

While on pfSense I needed to make a firewall rule for IPsec that used any protocol from any IP to reach out to the VPN and make a connection. There will be no blocks here so that way we can fully communicate with azure. By going back onto azure and looking at the vpn we can see that

we have data in and out. We can also go on pfsense and check the status and see that we have established our connection!

I am now able to create a VM on azure that will use my local network created on azure that will start with the IP 10.30.1.1-254 and I am able to ping my other windows machines because the site-to-site VPN allows us to do so! I am able to RDP into my new VM created on azure which means that the site-to-site VPN is fully working.

Cyber Lab: Phase 8

