Abstract

This project is aimed at security analyst, penetration tester, or anybody interested in exploiting cloud environments to establish vulnerable areas and then secure them.

AWS Penetration Testing with Kali Linux

CIS-4850 Project

Jeffrey Tam, Jerry Petty, Joseph Arias, Mayowa Toyinbo, Bryan Chica

Table of Contents

[Creating AWS Account 3](#_Toc183971128)

[Give external user access to the account 6](#_Toc183971129)

[Assign User permission to an Account 9](#_Toc183971130)

[Accept Invitation from AWS 11](#_Toc183971131)

[Setting up a Pen testing Lab on AWS 13](#_Toc183971132)

[Setup Networking Environment 14](#_Toc183971133)

[Setting up VPC - In Progress 14](#_Toc183971134)

[Planning 14](#_Toc183971135)

[Deploying a VPC 15](#_Toc183971136)

[Creating Subnets – In Progress 18](#_Toc183971137)

[Configuring Firewall (Security Groups) – In Progress 18](#_Toc183971138)

[Setup Vulnerable Servers 20](#_Toc183971139)

[Vulnerable Ubuntu EC2 instance 20](#_Toc183971140)

[Provisioning an Ubuntu EC2 instance 20](#_Toc183971141)

[Putty login 25](#_Toc183971142)

[Install vulnerable service on Ubuntu EC2 instance 28](#_Toc183971143)

[Vulnerable Windows Server instance – In Progress 34](#_Toc183971144)

[Configuring a vulnerable web application on Windows 38](#_Toc183971145)

[Setting Up a Kali Pen testing box on AWS 45](#_Toc183971146)

[Setting up Kali Linux on AWS EC2 45](#_Toc183971147)

[Configuring Kali Linux instance 45](#_Toc183971148)

[Setting up Guacamole for remote access – SKIP 46](#_Toc183971149)

[Exploitation on the Cloud Using Kali Linux – Screenshots Needed 53](#_Toc183971150)

[NMAP 53](#_Toc183971151)

[Configuring and running Nessus 53](#_Toc183971152)

[Install Nessus on Kali 53](#_Toc183971153)

[Exploiting a vulnerable Linux VM 56](#_Toc183971154)

[Nessus scan on Ubuntu Server 56](#_Toc183971155)

[Exploitation on Ubuntu 57](#_Toc183971156)

[Exploiting a vulnerable Windows VM – IN PROGRESS 58](#_Toc183971157)

[Exploitation on Windows 58](#_Toc183971158)

# Creating AWS Account

Select a region closest to you. Preferably US West

A screenshot of a computer

Description automatically generated

Next, we want to create another account to work on our project. In the search box, type **AWS Organizations.** Select the service that fits the name

A screenshot of a computer

Description automatically generated

*By default, your first account will be the management account. Best practice is to use the management account for admins tasks such as account management, policies, billing, and so on. Is best to avoid deploying workload to this account.*

With that being said, we will create a new account dedicated to our project

A screenshot of a computer

Description automatically generated

In **Add an AWS account,** give it a name and assign it an email address -> **Create AWS Account**

*\*\* I recommend using an email alias by using sub addressing with a “****+****” sign after your username, for example:*

*bchica+example@calstatela.edu*

*Incredibly useful when you want to create multiple accounts but not create separate emails for each. The email should be sent to the original inbox.*

*A screenshot of a computer

Description automatically generated*

You will now see you newly create OU (Account)

A screenshot of a computer

Description automatically generated

# Give external user access to the account

In the AWS portal, search **IAM Identity Center.**  Select that service

A screenshot of a computer

Description automatically generated

In the **IAM Identity Center** dashboard, select the **Users** in the left pane -> click **Add User** button

A red line on a white background

Description automatically generated

In **Step 1,** we will only fill out the **Primary Information** section. Have the **Password** be sent via email for this user to setup. You skip optional boxes and hit **Next**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Step 2** is to add users to an AWS Managed Group, that will be skipped. Click **Next**

**A screenshot of a computer

Description automatically generated**

In **Step 3,** you will verify that all the information is correct. Then hit **Add user**

You should now see the new user

A screenshot of a computer

Description automatically generated

This will also send the user an invite via email

## Assign User permission to an Account

Go to **AWS accounts,** select an account you want to add users to -> **Assign users or groups**

A red line on a white background

Description automatically generated

In **Users** tab, select the user account(s) you want to add to the account

A screenshot of a computer

Description automatically generated

Give the user/groups the proper permission for the account

A screenshot of a computer

Description automatically generated

Verify the users and the permissions you are assigning. Click **Submit** button

A screenshot of a computer

Description automatically generated

## Accept Invitation from AWS

You will receive the invite via email. Click **Accept Invitation**

**A screenshot of a computer

Description automatically generated**

This will direct you to the AWS site, where you will begin by setting up a password

**A screenshot of a login screen

Description automatically generated**

Once you set the new password. You will be required to log in with your username and new password

A screenshot of a login box

Description automatically generated

Once you login, if they see this window, you have not been given permission to open an account. Please contact your administrator for access. (Whoever sent you the invitation)

A screenshot of a computer

Description automatically generated

Once you are given proper permission to an account, you will now be able to see the account and the Role(permissions) in **AWS access portal**

Click the **Role** to enter the account **AWS Console**

# Setting up a Pen testing Lab on AWS

This repository will help set up a vulnerable-by-design environment in the cloud to minimize the risk involved while learning all about cloud penetration testing and ethical hacking. It will be a step-by-step guide for setting up pen testing environments within AWS, performing reconnaissance to identify vulnerable services using a variety of tools, finding misconfigurations and insecure configurations for various components, and how vulnerabilities can be used to gain further access.

We will use gen AI tools, like ChatGPT, to accelerate the preparation of IaC templates and configurations.

**Why should we build a penetration testing lab in the cloud?**

As a security professional, building pen testing labs can help us practice our skills safely in an isolated environment.

Aimed at helping those who don't have direct access to targets for penetration testing set up a vulnerable lab environment within AWS. This lab will allow testers to practice various exploitation techniques using Metasploit and scanning and vulnerability assessment using multiple tools within Kali Linux.

# Setup Networking Environment

## Setting up VPC - In Progress

Planning

How many Availability Zones will our VPC use?

* Always start with 3 because it will work, have a spare for future AZ’s. That is 4 subnets. Can be used for different tiers:
  + Web
  + Apps
  + DB
  + Spare
* Each tier will have its own subnets, a different subnet for each tier per AZ. That is 16 total subnets.

We will only have 2 subnet. A public and private subnet

* + Region
    - US-West-1
  + VPC CIDR:
    - 10.48.0.0/20
  + 2 Subnets (Public and Private)
    - 10.48.50.0/24 (Private)
      * Usable Host Range:
        + 10.48.50.1 - 10.48.20.254
      * Broadcast
        + 10.48.50.255
    - 10.48.100.0/24 (Public)
      * Usable Host Range:
        + 10.18.100.1 – 10.48.100.254
      * Broadcast
        + 10.48.100.255
  + Elastic IP Address
  + NAT Gateway
  + Internet Gateway
  + DNS (Route53)
    - **enableDNSHostNames**
      * Gives public DNS names to instances with public IPs
    - **enableDNSSupport**
      * Enables DNS resolution in VPC
    - These are 2 things you need to check if you are having DNS issues

Using and IGW

1. Create IGW
2. Attach IGW to VPC
3. Create custom RT
4. Associate RT
5. Default Routes => IGW
6. Subnet allocate Public IPv4

## Deploying a VPC

1. Make sure your region is set to **us-west-1**

A screenshot of a computer

Description automatically generated

1. In the AWS console, search for **VPC**

A screenshot of a computer

Description automatically generated

1. From the **VPC dashboard,** go to **Your VPCs -> Create VPC**

A screen shot of a computer

Description automatically generated

1. In **Create VPC,** the following settings were chosen:

* Resources to create: **VPC Only**
  + **VPC and more** creates a VPC, subnets, NAT gateways, and VPC endpoints
* Name tag: **CI4850-vpc1**
  + This is the VPC name
* IPv4 CIDR block: **IPv4 CIDR manual input**
  + There is no need for us to setup and IP address manager (IPAM) pool
* IPv4 CIDR: **10.48.0.0/20**
  + We won’t have a big environment for that many hosts. Just did enough for two /24 subnets
* IPv6 CIDR block: **No IPv6 CIDR block**
  + No need for IPv6
* Tenancy: **Default**
  + We don’t have the enough to run on dedicated hardware

A screenshot of a computer

Description automatically generated

1. Once the VPC gets created, you should get a green banner that says the VPC was created successfully, the info you set should be displayed in **Details.**

A screenshot of a computer

Description automatically generated

* 1. To give a public DNS names to out instances with public IPs, go to **Actions -> Edit VPC Settings**

A screenshot of a computer

Description automatically generated

* 1. Check **Enable DNS hostnames** (this is disabled by default)

A screenshot of a computer

Description automatically generated

### Creating Subnets – In Progress

## Configuring Firewall (Security Groups) – In Progress

Each EC2 will be protected by its own virtual firewall, known as **Security Groups** in AWS. It manages access to the EC2 instance by controlling inbound and outbound traffic.

* The EC2s that is running in a public subnet. So the Kali box will allow **SSH** connections for remote access
* Private SG will allow **RDP (**port 3389) and **SSH (**port 22) connections.
  + To allow traffic from public subnet, we will add an inbound rule that allows traffic coming from the CIDR we assigned the public subnet
  + For testing purpose, we will allows HTTP and HTTPS, ports 80 and 443, respectively.
  + Leave outbound rule as default

# Setup Vulnerable Servers

## Vulnerable Ubuntu EC2 instance

This section focuses on setting up a vulnerable Linux virtual machine (VM) as well as a Windows VM on AWS and putting it on the same network as the Kali instance.

**Technical requirements:**

* Damn Vulnerable Web Application
* Very Secure File Transfer Protocol Daemon (vsftpd) version 2.3.4

### Provisioning an Ubuntu EC2 instance

The vulnerable instance of Ubuntu will contain a single vulnerable FTP service, as well as some other services.

Let’s provision an Ubuntu instance that will be running a vulnerable operating system, Ubuntu 18.04

1. Search **EC2** in AWS console

A screenshot of a computer

Description automatically generated

1. Click **Launch instance** on EC2 dashboard

A screenshot of a computer

Description automatically generated

1. Start off with giving the instance a Name, then hit **Browse more AMIs** to search for the Ubuntu image

A screenshot of a computer

Description automatically generated

1. Search for **Ubuntu 20.04 LTS – Focal** image under the **AWS Marketplace AMIs** tab -> Click the **Select** button. You might get a pop-up window that will require you to subscribe to the AMI. Click **Subscribe Now.**

A screenshot of a computer

Description automatically generated

A screenshot of a web page

Description automatically generated

1. **Instance type** –> Select **t2.micro**. Verify is **free tier eligible** before selecting.

A screenshot of a computer

Description automatically generated

1. **Key pair (login)** –> select **Create your own key pair**
   1. To **Create key pair**, give it a **name.** Select **RSA** key type and save it in a **.ppk** format. (This key will also be used for the Kali box, we will use a .pem key for our Windows server)

We must do this to authenticate to the EC2 using a key pair authentication system

Download the key pair and store it securely. You will not be able to access the EC2 once you lose it

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Network settings**:
   1. **VPC**
      1. Select **cis4850-vpc1**
   2. **Subnet**
      1. Since we will allow traffic from the internet, put it under **cis4850-public-subnet**
   3. **Auto-assign public IP ->** select **Enable**
   4. **Firewall (security groups) -> Select existing security group**
      1. In the dropdown for **Common security groups,** select the existing SG, **pentest-lab-public-sg**
         1. The only rule allows port 22 traffic from the internet

A screenshot of a computer

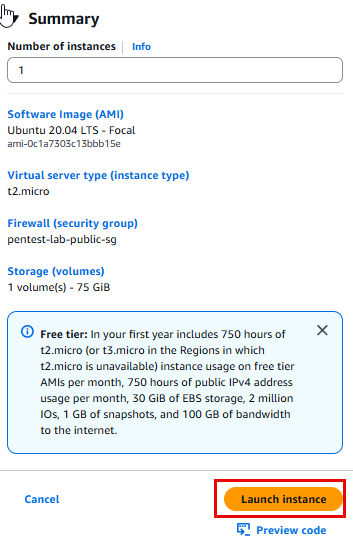
Description automatically generated

1. **Configure storage** -> Increase the volume size to **75 GiB**

A screenshot of a computer

Description automatically generated

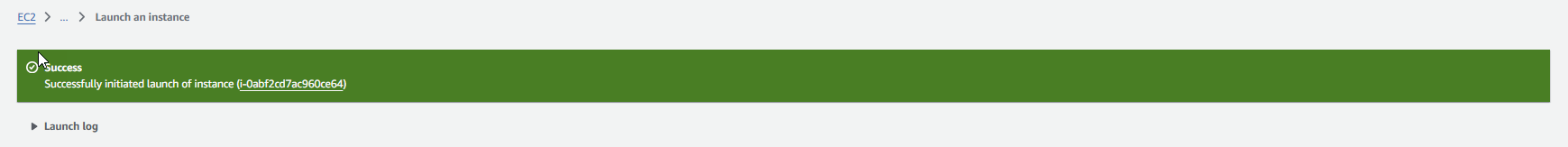
1. Keep the default settings in **Advanced Detail.**  All is left is **Launch instance**



After launching the instance, you will be subscribing to the **AMI,** the EC2 build will initiate. You will see a green **Success** banner once it is completed.

A screenshot of a computer

Description automatically generated



Last step, verify the EC2 Ubuntu instance was created by looking at the **Instances** in the **EC2 dashboard.**

**A screenshot of a computer

Description automatically generated**

### Putty login

1. Install **PuTTY** on your local machine

[PuTTY: a free SSH and Telnet client](https://www.chiark.greenend.org.uk/~sgtatham/putty/)

1. Once installed, launch **Putty**
2. Go to **Connection -> SSH ->** expand **Auth,** then go to **Connection** and, next to the field named **Private key file for authentication**, click on **Browse**. Point PuTTY to the .ppk file was created



1. Now in **Session,** in the **Host Name** field, enter the hostname, **ubuntu@<<your public ip>>.** Leave the port at 22 -> Click on **Open** to start the SSH session

A computer screen shot of a computer

Description automatically generated

1. During the first time that you login into the instance, you will receive the following alert. Click on **Yes** to continue. You will be authenticated to the Ubuntu instance.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

### Install vulnerable service on Ubuntu EC2 instance

Once connected, run the following command these commands to update the repository listing and all the packages installed on this Ubuntu instance. Type **Y** when prompted to continue with the installation.

sudo apt-get update && sudo apt-get dist-upgrade

A screenshot of a computer screen

Description automatically generated

On this Ubuntu host, we will install a vulnerable version of an FTP server, **vsftpd**

* Version 2.3.4 of this FTP software was found to be backdoored
* Thebackdoored version of **vsftpd 2.3.4** is archived on GitHub. We shall be using that code to install the vulnerable software. To start with, we need to clone the git repository:

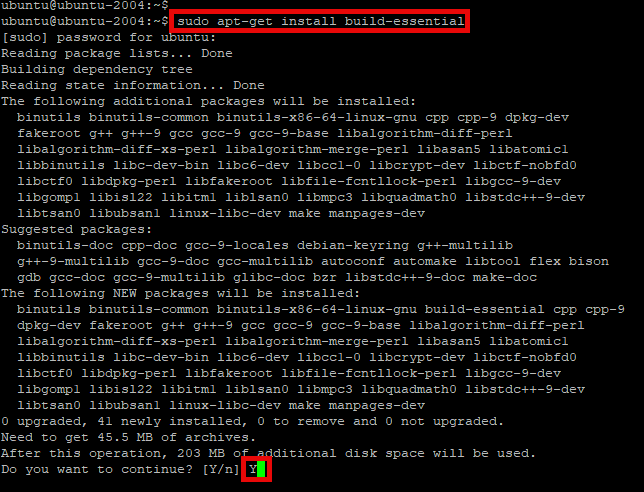
git clone https://github.com/nikdubois/vsftpd-2.3.4-infected.git

A screen shot of a computer

Description automatically generated

Next, we need to install packages for setting up a primary build environment. To do this, we run the following:

sudo apt-get install build-essential



Now, we cd into the **vsftpd** folder to build it from source. But, before doing that, we need to make a small change to the **Makefile**

cd vsftpd-2.3.4-infected/

nano Makefile

*A screenshot of a computer screen

Description automatically generated*

The ***-lcrypt*** value needs to be added to the link flag:

A screen shot of a computer

Description automatically generated

Once done, save the file and just run:

Make

*A screen shot of a computer screen

Description automatically generated*

If all goes well, we should see a **vsftpd** binary in the same folder. Verify by running:

ls -lha vsftpd

A screenshot of a computer

Description automatically generated

Next, we need to set up some prereqs before installing **vsftpd**.

Add a user called ***nobody*** and a folder called ***empty***. To do that, run the following commands:

useradd nobody

mkdir /usr/share/empty

A screenshot of a computer screen

Description automatically generated

If the user ***nobody*** already exists, verify by running:

cut -d: -f1 /etc/passwd | grep nobody

A black screen with red and green text

Description automatically generated

Next, verify the folder **empty** was created:

ls -l /usr/share | grep empty

A screen shot of a computer screen

Description automatically generated

Once done, we can run the installation by executing the following commands:

sudo cp vsftpd /usr/local/sbin/vsftpd

sudo cp vsftpd.8 /usr/local/man/man8

sudo cp vsftpd.conf.5 /usr/local/man/man5

sudo cp vsftpd.conf /etc

A screen shot of a computer

Description automatically generated

Once that's done, we need to execute the **vsftpd** binary to confirm whether we can connect to the localhost:

/usr/local/sbin/vsftpd &

ftp localhost

A screen shot of a computer

Description automatically generated

Nexy step is to set up anonymous access to the FTP server. To do this, we need to run the following commands:

mkdir /var/ftp/

useradd -d /var/ftp ftp

chown root:root /var/ftp

chmod og-w /var/ftp

A screen shot of a computer screen

Description automatically generated

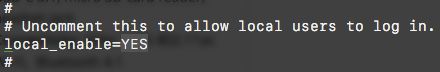
Lastly, enable local login to the **vsftpd** server by making the following changes to ***vsftpd.conf***`. Do this by running:

nano /etc/vsftpd.conf

Uncomment **local\_enable**. It should look like the second picture

A screenshot of a computer

Description automatically generated



## Vulnerable Windows Server instance – In Progress

Here we set up an attack vector through a Windows server that's running a vulnerable web application.

In this lab, we will be using a **Windows Server 2016** instance from the AWS Marketplace:

The provisioning steps are much more identical to what we used to set up the Linux instance earlier. Make sure the VPC settings are like what we used for the previous instance. This will later allow us to configure the VMs to be on the same VPC and Subnet. The only difference will be the **key types**, make sure to select .pem format.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

On the **EC2 dashboard,** check the new windows distance -> **Connect**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Download the **RDP shortcut** to your desktop

A screenshot of a computer

Description automatically generated

A computer monitor and a symbol

Description automatically generated

On that same page, you are provided the IP and username to connect with. Now click **Get password**

A screenshot of a message

Description automatically generated

On get **Windows Password** -> **Upload the private key** downloaded earlier -> hit on **Decrypt Password**. This will provide us with the password that we can use to RDP into our Windows server instance.

A screen shot of a computer

Description automatically generated

Now double-click the **Remote Desktop shortcut** and connect with username **Administrator,** and password that was decrypted

A computer error message

Description automatically generated

### Configuring a vulnerable web application on Windows

Once we are logged in, the next step is to set up XAMPP on the Windows server so we can host a vulnerable website on the server.

But before we proceed, we need to install the latest version of Firefox on the server, since Internet Explorer version that comes packaged with Windows Server 2016 is old and doesn't support some website configurations. Use IE browser to download.

To download XAMPP, just access https://www.apachefriends.org/download.html and download the version that's built for Windows:

A screenshot of a computer

Description automatically generated

Once you download the executable, **Run as administrator**

A screenshot of a computer

Description automatically generated

Hit **Next** all the way through

A screenshot of a computer

Description automatically generated

To begin with, let's clear up the XAMPP hosting folder by accessing C:\xampp\htdocs.

Create a new folder called **\_bak** and cut and paste all the existing files into that folder.

A screenshot of a computer

Description automatically generated

Now, let's download the vulnerable website's source code. For this, we will use one of the many vulnerable PHP samples that are available on GitHub: [***https://github.com/ShinDarth/sql-injection-demo/***](https://github.com/ShinDarth/sql-injection-demo/)

A screenshot of a computer

Description automatically generated

Extract the zipped folder, then copy the contents of the **sql-injection-demo** folder into the **C:\xampp\htdocs** folder. If done correctly, this is what the file structure should look like:

Once completed, the next step is to create a database for the application and import the data into it.

First, open the **XAMPP Control Panel,** start the **Apache, MySQL,** and **Tomcat** services.

A screenshot of a computer program

Description automatically generated

Once the **PID** and **Ports** appears, you need to access the phpMyAdmin interface, which is accessible at http://127.0.0.1/phpmyadmin. Once here, select the New option under Recent. Here we create a new database called sqli:

A screenshot of a computer

Description automatically generated

Next, to import data into the newly created database, we go into the Import tab and browse to the database.sql file that we just extracted into the htdocs folder:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Once we click on go we will see a success message. Now, if we browse to http://127.0.0.1 in our browser, we will be able to access the vulnerable website:

We now have successfully configured a vulnerable web application on the Windows server! The next step will be to set up the networking rules within our VPC so that the vulnerable hosts are accessible from the other EC2 instances.

**### Configuring security groups within the lab**

In this section we configure network so that our web application isn't accessible to outsiders and, at the same time, so that the other lab machines can communicate with each other.

We had originally set all of the EC2 instances to be on the same VPC. This is how EC2 instances on the same subnet are able communicate with each other through internal IP addresses. However, AWS doesn't want to allow all 4,096 addresses on the same VPC to be communicating with each other. As a result, the default security groups don't allow communication between EC2 instances.

To allow connectivity from the Ubuntu instance to the Windows instance (you can repeat these steps for the Kali instance that will be set up in the next chapter), the first step is to get the Private IP address of the Ubuntu host:

Next, we need to modify the security group rules for the first Windows instance

Click **\*\*Security Group\*\*** in the summary pane

Click on the Edit button and add the rule allowing all traffic from the Kali Linux instance:

Once done, just save this configuration. To confirm that Kali can now communicate with the Windows server, let's run a curl command to see if the site is accessible:

curl -vL 172.31.26.219

Make sure to replace the IP address with your IP address for Windows. If all is well, there should be a bunch of JavaScript in response:

# Setting Up a Kali Pen testing box on AWS

Let’s now focus on creating an Amazon EC2 instance, setting it up with a Kali Linux Amazon Machine Image (AMI) that is available on the Amazon Marketplace, and configuring remote access to this host through a variety of means.

This means that a penetration tester can quickly set up a Kali Linux instance on the Amazon Cloud and access it at any time for any kind of penetration test.

Here we will focus on creating an Amazon EC2 instance, setting it up with a Kali Linux AMI, and configuring remote access to this host in a variety of ways. Once set up, a penetration tester can remotely access a Virtual Private Cloud (VPC) belonging to an AWS account and perform pentests within that VPC and on any remote hosts using Kali.

**Technical requirements:**

- AWS EC2 instance

- Kali Linux AMI

- Apache Guacamole (https://guacamole.apache.org)

- SSH client and a browser

Setting up Kali Linux on AWS EC2

We will start by accessing the Kali Linux AMI on the AWS Marketplace:

A screenshot of a computer

Description automatically generated

### Configuring Kali Linux instance

1. Use the same VPC as we did for the other servers in the pentesting lab

2. We set up the Security Group in such a way that unauthorized outsiders would not have access to the instances. However, in this case, we need to allow remote access to our Kali instance. Therefore, we need to forward the SSH and the Guacamole remote access port to a new Security Group

3. We can use the same key pair that was created during the setup of the lab environment

4. For now, we will login us **Putty**. Use the **kali** username

5. One of the first steps we do is set the root and user password

To change the root password. Run:

sudo passwd

A screen shot of a computer

Description automatically generated

We also need to change the password of the current user. Run:

sudo passwd ec2-user

A screenshot of a computer screen

Description automatically generated

### Setting up Guacamole for remote access – SKIP

Skip this section. I was not able to make Guacamole work

Apache Guacamole is a clientless remote access solution that will allow you to access the Kali Linux instance remotely using a browser. The traditional way of accessing such servers is over SSH, but this will not be able to provide a GUI when accessed from a mobile device.

#### Hardening and installing prerequisites

For starters, we install and set up a firewall and IP blacklisting services to protect against brute-forcing attacks and similar attacks on the internet.

The services we will install are **ufw** and **fail2ban**.

1. Run

sudo apt-get install ufw fail2ban

A screen shot of a computer

Description automatically generated

2. Once the firewall is installed, we need to allow two port that we will be using for remote access.

port 22 for SSH

sudo ufw allow 22

port 55555 for Guacamole

sudo ufw allow 55555

A screenshot of a computer program

Description automatically generated

3. Restart the **ufw** service:

sudo service ufw start

#### Setting Guacamole for SSH and RDP access

4. Next, we need to install the prerequisites for Apache Guacamole. You can do this by executing the following command:

sudo apt-get install build-essential htop libcairo2-dev libjpeg-dev libpng-dev libossp-uuid-dev tomcat10 freerdp2-dev libpango1.0-dev libssh2-1-dev libtelnet-dev libvncserver-dev libpulse-dev libssl-dev libvorbis-dev

A screen shot of a computer

Description automatically generated

5. Post-installation, we need to modify the configuration of Apache Tomcat to listen on port 55555 (as set in our Security Group) rather than the default 8080. To do this, we need to run the following command:

sudo nano /etc/tomcat10/server.xml

Change the connector port from 8080 to 55555

A screen shot of a computer

Description automatically generated

6. Next we set up the RDP service on Kali. To this by installing xrdp:

sudo apt install xrdp

A screenshot of a computer

Description automatically generated

7. Then, allow all users to access the RDP service (the X Session).

Edit this file:

sudo nano /etc/X11/Xwrapper.config

Change the value of **allowed\_users** to anybody

A computer screen with white text and blue text

Description automatically generated

8. Lastly, set the xrdp service to start automatically and enable the services

sudo update-rc.d xrdp enable

sudo systemctl enable xrdp-sesman.service

sudo service xrdp start

sudo service xrdp-sesman start

A black background with text

Description automatically generated

9. Download the source code for Apache Guacamole</a>

*Keep in mind that you need to download the* ***latest guacamole-server.tar.gz****and****guacamole.war****files. At the time writing this documentation, the latest version is****1.5.5****, which we can download using the following command:*

sudo wget https://downloads.apache.org/guacamole/1.5.5/source/guacamole-server-1.5.5.tar.gz

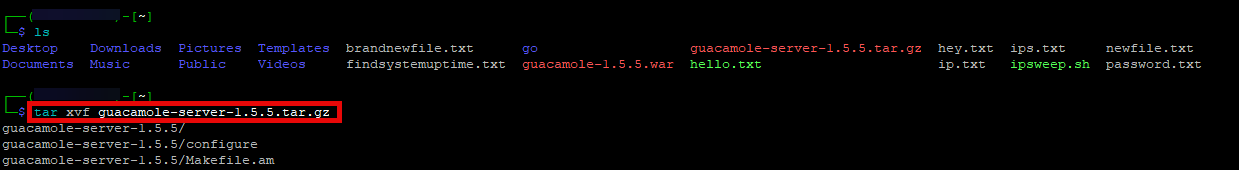
sudo wget https://downloads.apache.org/guacamole/1.5.5/binary/guacamole-1.5.5.war

A screen shot of a computer

Description automatically generated

10. Next, extract the source by executing the following code:

tar xvf guacamole-server-1.5.5.tar.gz



11. Enter the extracted directory so we can start building and installing the package:

cd guacamole-server-1.5.5/

CFLAGS="-Wno-error" ./configure --with-init-dir=/etc/init.d

make -j4

sudo make install

sudo ldconfig

sudo update-rc.d guacd defaults

A computer screen with text and numbers

Description automatically generated

Guacamole has been successfully installed, but further configuration is needed in order to fully set up remote access.

#### Configuring Guacamole

Guacamole's default configuration directory is */etc/guacamole*. We will need to created this directory and its subdirectories

sudo mkdir -p /etc/guacamole/lib/

sudo mkdir -p /etc/guacamole/extensions/

We will also need to perform these actions to avoid write issues with FreeRDP. If you get access denied, switch over to **root**

mkdir -p /usr/sbin/.config/freerdp

chown daemon:daemon /usr/sbin/.config/freerdp

mkdir -p /var/guacamole

chown daemon:daemon /var/guacamole

A screen shot of a computer

Description automatically generated

A computer screen shot of white text

Description automatically generated

Restart both the guacd and tomcat8 services to get Apache Guacamole up and running!

systemctl daemon-reload

sudo service guacd restart

sudo service tomcat10 restart

A screenshot of a computer program

Description automatically generated

Last configuration step that is required—copying the authentication information into the Guacamole client directory.

This is done by executing the following code:

mkdir /usr/share/tomcat10/.guacamole

ln -s /etc/guacamole/guacamole.properties /usr/share/tomcat10/.guacamole

A screen shot of a computer

Description automatically generated

# Exploitation on the Cloud Using Kali Linux – Screenshots Needed

This section walks us through the process of scanning for vulnerabilities in a vulnerable lab, exploiting these vulnerabilities using Metasploit, gaining reverse shells, and various other exploitation techniques. This serves to help pentesters practice on a cloud environment that simulates real-life networks.

We will focus on the process of automated vulnerability scans using the free version of a commercial tool and then exploiting the found vulnerabilities using Metasploit.

## NMAP

Nmap is a great tool that comes pre-installed in Kali Linux. It’s a popular tool used in pentesting due to its ability to discover services and open ports on multiple hosts at once – commonly known as port scanning.

Run the following script to detect anonymous FTP logins:

nmap -sV -sC -p 21 <Victim IP>

What do these switches mean?

* **-sV:** enumerates the version of the service
* **-sC:** check for anonymous FTPs
* **-p:** indicate the port to scan

## Configuring and running Nessus

We will focus on the process of automated vulnerability scans using the free version of a commercial tool and then exploiting the found vulnerabilities using Metasploit. We will set up Nessus on our PentestBox on EC2. Then we shall use it to run basic and advanced scans on the lab that we set up earlier.

### Install Nessus on Kali

Nessus comes in a .deb package that can be directly installed using dpkg.

1. To install Nessus, the first step is to download the .deb package from the tenable website, on https://www.tenable.com/downloads/nessus:

2. Once downloaded, we need to transfer this to our Kali PentestBox on AWS. We can do this file transfer using **WinSCP** on Windows. On Linux/macOS, the native SCP utility can be used. The setup is available at https://winscp.net/eng/download.php

3. Once **WinSCP** is installed, we need to set up a connection to our Kali PentestBox. First, we need to add a new site:

4. Next, we need to add the public key, downloaded from AWS, for authentication. To do this, we need to click on Advanced and set the path to the key on SSH | Authentication:

5. On **SCP**, let's dragged the .deb package into the root folder that was just accessed. Once done, we can get started with installing the package. This can be achieved using dpkg through an SSH shell:

sudo dbkg -l Nessus

6. Let's now start the **Nessus** service and confirm that is running:

sudo /etc/init.d/nessusd start

sudo service nessusd status

7. Once you verify the service is running. Next, we need to set up SSH tunneling to forward port 8834 from the Kali PentestBox to our localhost over the SSH connection. On a Linux Terminal, the following syntax needs to be used:

ssh -L 8834:127.0.0.1:8834 kali@<IP address>

8. On Windows, if you're using PuTTY, the SSH Tunnels can be configured here, by clicking on the Tunnels option after launching PuTTY. Once done, reconnect to the instance and you can now access Nessus on your local machine on https://127.0.0.1:8834.:

Configuring Nessus

Once Nessus has been installed and the SSH tunnel configured, we can access Nessus on the browser by pointing at https://127.0.0.1:8834. We will need to go through a set of first steps to set up Nessus now.

1. First step is to create an account

2. Once you enter credentials and proceed to the next step. We will activate a home license. We can grab one at https://www.tenable.com/products/nessus-home by filling in the following form:

3. Once you've received the activation code by email, enter it into the web interface and trigger the initialization process. Now Nessus goes through the process of downloading data that is needed for the scanning of network assets:

This process usually takes a few minutes, so there's enough time to go grab a cup of coffee while this is happening.

**### Perform the first Nessus Scan**

1. Once on the new scan tab, we need to start a Basic Network Scan:

2. After clicking on Basic Network Scan, we need to give a scan name and enter the IPs of the two other hosts that we set up in the lab:

3. Next up, we configure the DISCOVERY and ASSESSMENT options. For discovery, let's request a scan of all services:

This has the advantage of enumerating all services running on a host and discovers hosts if no traditional services are running on them.

4. Let's configure Nessus to scan web applications as well:

5. Finally, we Launch the scan:

Scanning is a time-consuming process, so this would take around 15 to 20 minutes to complete on average, if not more.

## Exploiting a vulnerable Linux VM

Our first target is the Ubuntu instance that we set up in our lab. We will go through the scan results for this host and try to gain unauthorized access to the host.

### Nessus scan on Ubuntu Server

Let's start with the Nessus scan results for our Ubuntu server host:

Unsurprisingly, we just find a bunch of information vulnerabilities, since there are just two services installed—FTP and SSH. The FTP server has a backdoor baked into it; however, it has not come out as a critical vulnerability. If you look at the last result in the Linux scan, it does detect that vsftpd 2.3.4 is installed, which comes with a backdoor.

To summarize the other results on this page, the Nessus SYN scanner simply lists a number of services enabled on the host:

There is more useful information on this page that can be manually inspected. We shall now focus on exploitation of the **\*\*vsftpd\*\*** service that we installed on the Ubuntu server.

### Exploitation on Ubuntu

To exploit the **\*\*vsftpd\*\*** service, we shall use **\*\*Metasploit\*\***, which comes with Kali Linux built in. This can be loaded up by simply entering msfconsole into the Terminal:

msfconsole

We can simply search for the name of the service to see if there are any associated exploits. Run the following:

search vsftpd

This will turn up a list of the exploits with that specific keyword.

We can use this exploit by running the following:

use exploit/unix/ftp/vsftpd\_234\_backdoor

This changes the prompt to that of the exploit. Now all that needs to be done is to run the following:

set RHOST <ip address of Ubuntu server>

Finally, just run *exploit*, and **vsftpd** exploit would be executed to provide an interactive reverse shell with root privileges:

Using this reverse shell, you have full freedom to run whatever commands are supported on the OS. This is a good place to play around with auxiliary and post-exploitation modules on Metasploit.

## Exploiting a vulnerable Windows VM – IN PROGRESS

Nessus scan for Windows throws up a number of issues thanks to the end-of-life OS being used, as well as the outdated server.

There are several issues dealing with outdated OpenSSL and PHP installations, as well as a couple of findings pointing out that Windows Server 2016 is an EOL OS

### Exploitation on Windows

The vulnerable web application has an SQL injection vulnerability.

*SQL injection allows an attacker to inject arbitrary SQL queries and execute them on the backend DBMS.*

This vulnerability is present on the following URL:

http://<ip>/books1.php?title=&author=t

An **SQL injection** on a web application that is potentially running with admin privileges means that there is a possibility of a complete takeover of the web application.

For this purpose, we use sqlmap. To attack the URL with **sqlmap**, the syntax is as follows:

sqlmap --url="http://<IP>/books1.php?title=&author=t"

A **sqlmap** confirms that the injection vulnerability is present

Next, use **sqlmap** to gain shell access on the remote server.

Then it follows it up by uploading a web shell that executes commands and returns the output of the command, all with a single command. To trigger this, execute the following:

sqlmap --url="http://<IP>/books1.php?title=&author=t" --os-shell --tmp-path=C:\\xampp\\htdocs

* The **--os-shell** asks sqlmap to spawn a shell using the method described previously and
* The **--tmp-path** value specifies where to upload the PHP files for the purpose of spawning a shell.

Once the command is executed, user input would be prompted twice.

1. First instance is to select the technology, which is PHP in this case

2. Second instance is to trigger full path disclosures, which can be enabled.

If everything goes well, we should be presented with an interactive shell: