Cloud Security Simulation – Implementing Network Access Control with UFW Firewall

Objective :

The primary objective of this project is to simulate cloud security concepts on a local Linux system by configuring a firewall using UFW (**Uncomplicated Firewall**). This project is intended to help understand how security groups and network access control work in cloud environments like AWS, Azure, and Google Cloud. The key goal is to implement firewall-based access control, restrict access to essential services, log and monitor firewall activity, prevent brute-force attacks, and ensure security for cloud-like environments.

Tools and Methods Used:

- **Tools:**
  - **UFW (Uncomplicated Firewall):** A simple firewall configuration tool used to control incoming and outgoing network traffic.
  - **Linux-based system:** Any system running Linux (Ubuntu, Kali Linux, Debian) or a virtual machine setup using VirtualBox/VMware.
- **Methods:**
  - **UFW Commands:** Used to configure and manage firewall rules for securing the system.
  - **System Management and Monitoring:** Commands for checking firewall status, allowing necessary services, enabling logging, and monitoring the system's security activity.

Step 1: Install and Enable UFW Firewall

UFW is a **host-based firewall** that allows us to manage **incoming and outgoing network traffic**.

1.1 Check Firewall Status

Before configuring, check if the firewall is active:

sudo ufw status

If inactive, enable it:

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw status
Status: inactive
```

sudo ufw enable

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw enable
Firewall is active and enabled on system startup
```

1.2 Allow Essential Services

Some services need to remain open for system management.

sudo ufw allow OpenSSH

sudo ufw allow 80 # HTTP (Web Server)

sudo ufw allow 443 # HTTPS (Secure Web Traffic)

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw allow OpenSSH
sudo ufw allow 80      # HTTP (Web Server)
sudo ufw allow 443     # HTTPS (Secure Web Traffic)

Rule added
Rule added (v6)
Rule added
Rule added (v6)
Rule added
Rule added (v6)
```

Step 2: Implement Least Privilege Access Control

Cloud security follows the **principle of least privilege**, which means **only necessary ports should be open**.

2.1 Deny All Incoming Traffic by Default

To prevent unauthorized access, **block all incoming traffic** and allow outgoing traffic:

sudo ufw default deny incoming

sudo ufw default allow outgoing

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw default deny incoming
sudo ufw default allow outgoing

Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
```

2.2 Allow SSH Only from Trusted IP

Instead of allowing SSH to **everyone**, restrict access to **only your IP** with our actual public IP):

sudo ufw allow from 152.59.9.196 to any port 22

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw allow from 152.59.9.196 to any port 22

Rule added
```

If we need remote access from a different IP later, update this rule accordingly.

Step 3: Restrict Access to Database and Web Servers

3.1 Allow MySQL/MariaDB Access Only from a Specific IP

If our server is running a database, allow access **only from a trusted source** (e.g., an application server).

sudo ufw allow from 192.168.1.100 to any port 3306

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw allow from 152.59.9.196 to any port 3306

Rule added
```

This prevents **unauthorized database access** from external sources.

3.2 Limit Connections to Prevent Brute-Force Attacks

Limit SSH connections to **only 3 attempts per minute** to prevent brute-force login attacks:

sudo ufw limit OpenSSH

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw limit OpenSSH

Rule updated
Rule updated (v6)
```

Step 4: Enable Logging and Monitor Traffic

4.1 Enable Logging to Track Unauthorized Access

Enable logging to monitor firewall activity:

sudo ufw logging on

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo ufw logging on

Logging enabled
```

View Logs Using journalctl:

Instead of relying on the /var/log/ufw.log file, we can view UFW logs using journalctl (since systemd handles logs on most modern systems):

sudo journalctl -u ufw

```
┌──(root㉿kali)-[/etc/ufw]
└─# sudo journalctl -u ufw

Jan 28 23:01:41 kali systemd[1]: Starting ufw.service - Uncomplicated firewall...
Jan 28 23:01:41 kali ufw-init[10688]: Firewall already started, use 'force-reload'
Jan 28 23:01:41 kali systemd[1]: Finished ufw.service - Uncomplicated firewall.
```

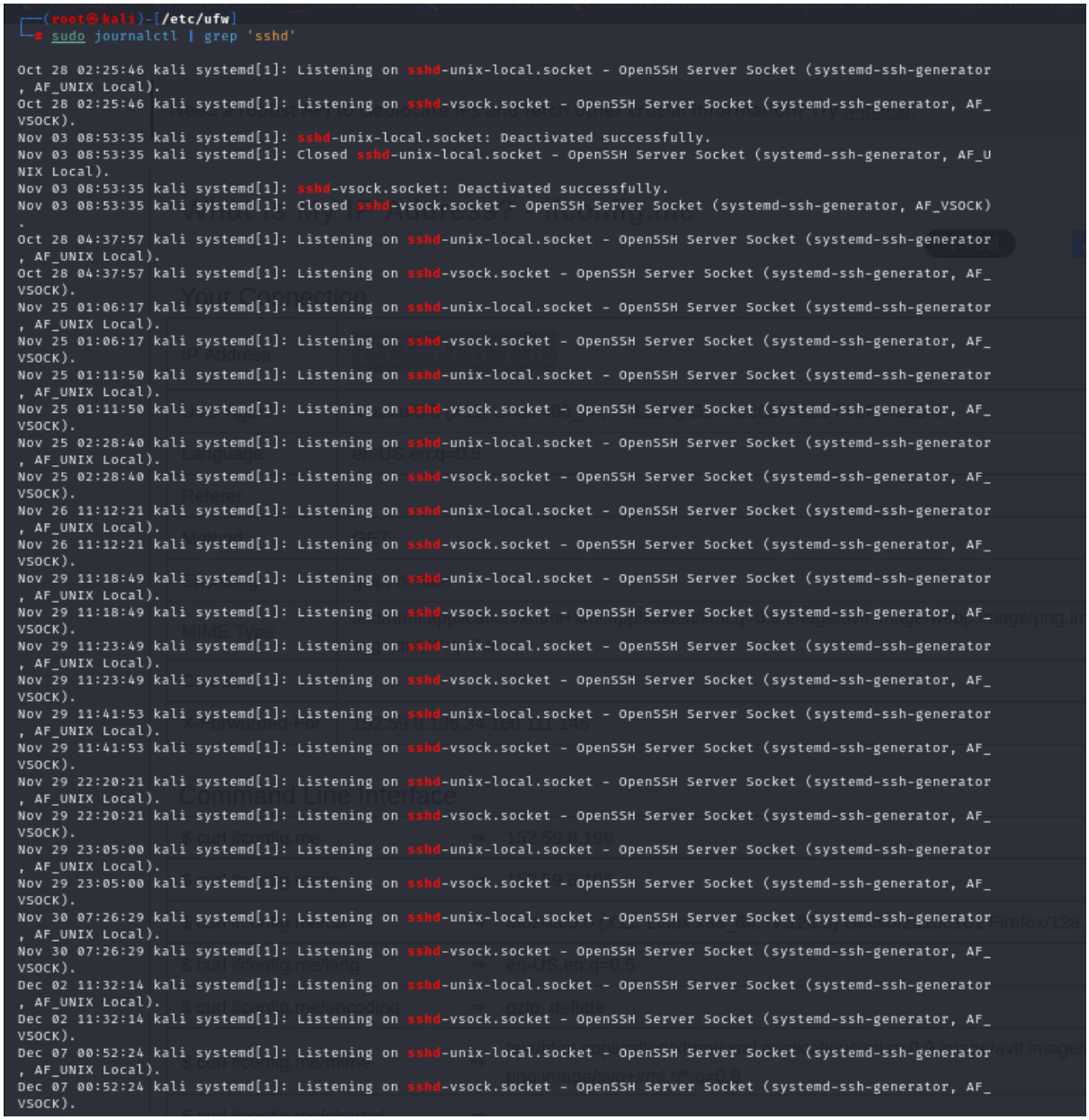
This command will show logs related to UFW.

Check System Logs for Unauthorized Access:

If we want to check for general unauthorized connection attempts, we can also look through the system logs:

sudo journalctl | grep 'sshd'

These steps should give you detailed insights into failed login attempts and unauthorized connections.



4.2 Test Firewall Rules

1.Verify Firewall Status (UFW Rules):

- Run the following command on our Kali machine to check the current status of our firewall and see which ports are open:

  sudo ufw status

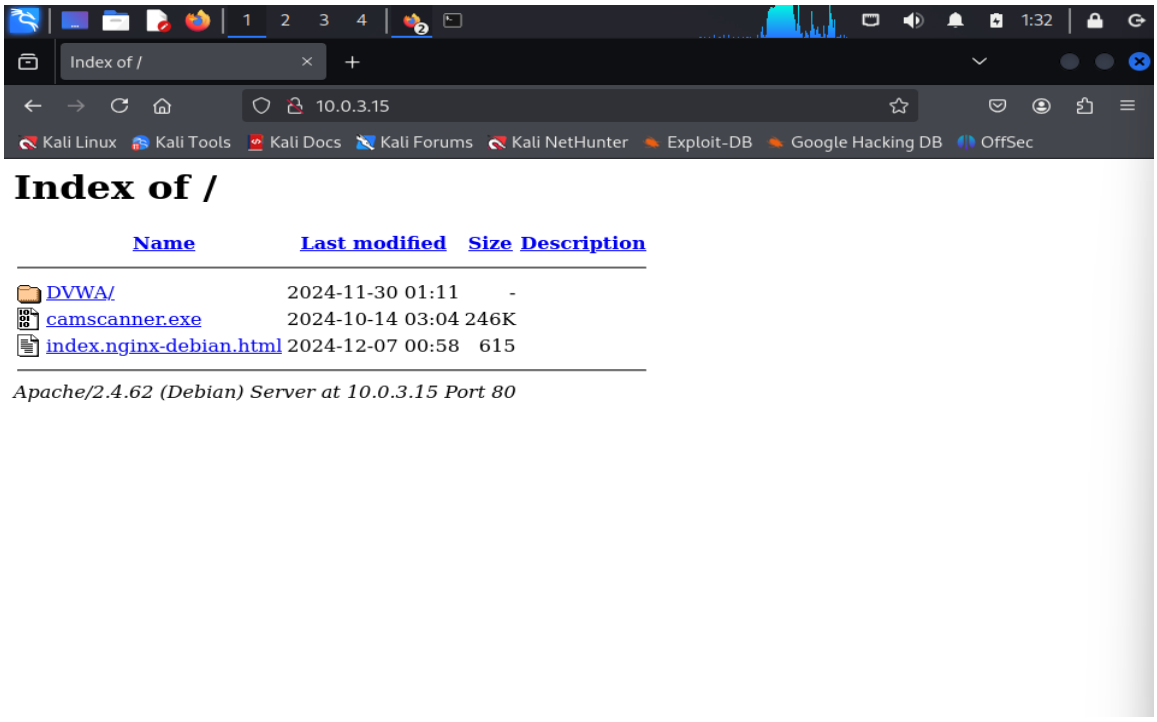
```
┌──(kali㉿kali)-[~]
└─$ sudo ufw status

Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    LIMIT       Anywhere
80                         ALLOW       Anywhere
443                        ALLOW       Anywhere
22                         ALLOW       152.59.9.196
3306                       ALLOW       152.59.9.196
OpenSSH (v6)               LIMIT       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
```

- Make sure the SSH port (22/tcp) is only allowed from our trusted IP (e.g., 152.59.9.196).
- Ports 80/tcp (HTTP) and 443/tcp (HTTPS) should be open for anyone if the web server is supposed to be accessible.

3. Access the Web Server (HTTP/HTTPS):

- **On another system (using our browser or terminal)**, check if the web server (e.g., Apache or Nginx) is accessible over HTTP and HTTPS.
  - Open the browser and enter our Kali machine's IP:

    http://10.0.3.15

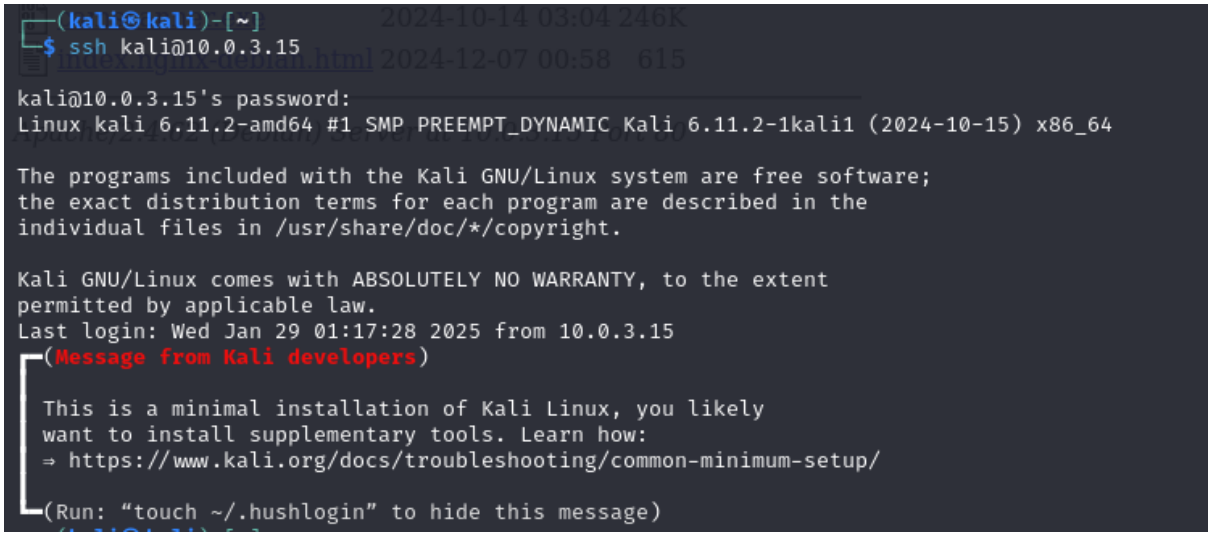


- If we have a web server running on our Kali machine and the firewall is correctly configured, the website should load.

4. Test SSH from Authorized IP (152.59.9.196):

- **From your trusted IP (152.59.9.196)**, try to SSH into the Kali machine:

  ssh kali@10.0.3.15



This connection should be allowed, as our IP is trusted. If everything is set up correctly, we will be able to access the machine.

By performing these steps, we can confirm that our firewall rules are working as intended: blocking unauthorized IPs from accessing SSH, allowing necessary ports for HTTP/HTTPS, and ensuring that your system's security is enforced

---

Step 5: Advanced Hardening with Custom Rules

5.1 Block Specific Malicious IP Addresses

If we detect an attack from a certain IP, block it permanently:

sudo ufw deny from 203.0.113.45

```
┌──(kali㉿kali)-[~]
└─$ sudo ufw deny from 203.0.113.45

[sudo] password for kali:
Rule added
```

5.2 Allow Connections Only at Specific Time Ranges

To restrict access **during non-working hours**, use:

sudo ufw allow from 192.168.1.50 to any port 22 proto tcp comment 'Allow SSH only during office hours'

```
┌──(kali㉿kali)-[~]
└─$ sudo ufw allow from 192.168.1.50 to any port 22 proto tcp comment 'Allow SSH only during office hours'

Rule added
```

we can later modify this rule based on actual requirements.

Challenges Faced and How They Were Overcome:

1. **Issue with SSH Access Timeout:**

During initial configuration, I faced issues with SSH access timing out, possibly due to firewall misconfiguration or incorrect IP rules. This was resolved by ensuring the firewall rules allowed SSH from trusted IP addresses and by checking if SSH was running properly.

2. **Log File Access:**

Initially, I encountered issues where the UFW log file was not found. This was resolved by ensuring that logging was enabled in UFW and by checking the correct log directory (/var/log/ufw.log).

3. **Firewall Configuration Conflicts:**

While configuring multiple services, I encountered conflicts in the firewall rules, especially with database access. To overcome this, I carefully reviewed the rules and ensured that specific ports were only allowed from trusted IPs.

Results and Outcomes:

1. **Successful Firewall Configuration:**

The firewall was successfully configured to allow only necessary traffic, including SSH from a specific trusted IP, HTTP, and HTTPS access for web servers, and MySQL database access from trusted IP addresses.

2. **Enhanced Security with Least Privilege:**

By denying all incoming traffic by default and allowing only specific services, I implemented the principle of least privilege effectively.

3. **Logging and Monitoring:**

Firewall activity was successfully logged, and I was able to track unauthorized access attempts via the log files.

4. **Brute-Force Attack Prevention:**

The SSH connection limit was set, effectively mitigating the risk of brute-force attacks.

5. **Advanced Hardening:**

Malicious IPs were blocked, and time-based access restrictions were applied to ensure enhanced security during non-working hours.

Future Scope :

1. **Integration with Cloud Platforms:** Expanding UFW configurations to integrate with cloud security groups in AWS, Azure, and Google Cloud.
2. **Automation & Orchestration:** Implementing Ansible or Terraform to automate firewall rule deployment and ensure consistency across multiple servers.
3. **Enhanced Threat Intelligence:** Integrating threat intelligence feeds to automatically block known malicious IP addresses.
4. **AI-Powered Anomaly Detection:** Using machine learning to analyze firewall logs and detect suspicious traffic patterns for proactive security measures.
5. **IPv6 Security Considerations:** Extending firewall rules to support IPv6 networks and ensure compatibility with modern cloud architectures.

Security Purposes in Real-Life Data :

1. **Protecting Web Servers:** Preventing unauthorized access and DDoS attacks on business-critical web applications.
2. **Securing Remote Access:** Allowing SSH access only from trusted locations to mitigate the risk of brute-force attacks.
3. **Database Security:** Ensuring only approved application servers can connect to production databases, reducing attack vectors.
4. **Regulatory Compliance:** Enforcing firewall policies to meet industry security standards such as PCI-DSS, GDPR, and HIPAA.
5. **Incident Response & Forensics:** Using firewall logs to trace malicious activities and enhance forensic investigations.

Conclusion:

This project successfully simulated cloud security concepts on a local Linux machine using UFW to implement network access control, restrict access to essential services, and ensure the system was secure from unauthorized access. The principles of cloud security groups were applied using firewall rules, and advanced hardening techniques were implemented for better protection.