Cloud & IoT Security

Assignment 1: Configuring Cloud Firewall to Allow Only HTTP and SSH Access

Objective

The aim of this assignment is to create a virtual machine (VM) in a cloud environment (AWS, Azure, or Google Cloud) and configure its firewall to allow **only HTTP (port 80)** and **SSH (port 22)** access while blocking all other inbound network traffic. Verification is conducted through testing allowed and denied port accessibility.

Platform Used

For this assignment, **Amazon Web Services (AWS)** EC2 was used due to its simplicity in managing security groups and network configurations.
(You can perform equivalent actions in **Azure NSG** or **Google Cloud VPC firewall**.)

Step 1: Launch a Virtual Machine Instance

1. Log in to the **AWS Management Console**.
2. Go to **Services → EC2 → Instances → Launch Instance**.
3. Choose an Amazon Machine Image (AMI) such as **Ubuntu 22.04 LTS**.
4. Select an instance type (e.g., t2.micro, free tier eligible).
5. Configure network settings:
    - Select the default VPC.
    - Choose **Create new security group**.
6. Enter instance name and key pair for SSH authentication.

Step 2: Configure the Firewall (Security Group) Rules

In AWS, **Security Groups** act as a virtual firewall to control inbound and outbound network traffic.

Create or edit the security group associated with the VM as follows:

Inbound Rules

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Your IP (or CIDR)	Allow SSH access
HTTP	TCP	80	0.0.0.0/0	Allow web access

Remove any other pre-existing inbound rules such as HTTPS (443) or ICMP.

Outbound Rules

Keep the default outbound rule (allow all traffic) to enable updates and communications initiated by the VM.

Save and attach this security group to the instance.

Step 3: Connect to the Instance via SSH

From a terminal on your local system, connect to the VM using its public IP address:

```
ssh -i yourkey.pem ubuntu@<public-ip-address>
```

If successful, this confirms SSH (port 22) access is working correctly.

Step 4: Enable and Test HTTP Access

1. Install Apache web server on the VM:

```
sudo apt update
```

```
sudo apt install apache2 -y
```

2. Once installed, confirm the server is running:

```
systemctl status apache2
```

3. Open a browser and navigate to:

text

http://<instance-public-ip>

You should see the default Apache web page, confirming HTTP (port 80) access is open.

Step 5: Verify Other Ports Are Blocked

Use nmap or telnet to scan the instance for open ports:

Example command from your local system:

```
nmap -Pn <public-ip-address>
```

Expected output should show only:

- Port 22/tcp open (SSH)
- Port 80/tcp open (HTTP)

All other ports should appear as **filtered or closed**, confirming that your firewall settings are enforced.

Step 6: Verification and Evidence

Test Description	**Expected Result**	**Status**
SSH connection (port 22)	Connection successful	Passed
HTTP access (port 80)	Apache page loads successfully	Passed
Other ports (e.g., 21, 25, 8080)	Connection refused or filtered	Passed

Step 7: Azure and Google Cloud Alternatives

- **Azure:** Configure inbound rules in the **Network Security Group (NSG)** for the VM. Allow ports 22 and 80 only; block all others.
- **Google Cloud:** Create **VPC firewall rules** to allow only TCP ports 22 and 80 to the VM network tag.

Google Cloud firewall verification commands:

```
gcloud compute firewall-rules list
```

```
nmap <vm-external-ip>
```

Conclusion

The assignment successfully demonstrated the setup of a secure virtual machine instance with restricted network access. The security group/firewall was effectively configured to allow only **SSH (22)** and **HTTP (80)** traffic while preventing unauthorized inbound connections across all other ports. This principle enhances cloud instance security by reducing the exposed attack surface.

Assignment 2:

Objective:

Connect two IoT devices (e.g., Raspberry Pi or ESP32) to a Wi-Fi network. Use Wireshark to capture network traffic and identify any unsecured communications between the devices.

I choose Android Phone and Laptop.

Network Traffic Capture and Analysis Between Android Phone and Laptop Using Wireshark

Equipment and Tools Used

- Android smartphone connected to Wi-Fi
- Laptop connected to the same Wi-Fi network
- Wireshark installed on the laptop

Step 1: Setup HTTP Server on Laptop

1. Open a terminal on the laptop.
2. Start a simple HTTP server using Python 2 on port 80:

```
sudo python2 -m http Server 80
```

3. Confirm that the server is running and listening on port 80.

Step 2: Connect Devices and Generate Traffic

1. Both devices connected to the same Wi-Fi network (note their IP addresses).
2. From the Android device, open a browser and navigate to the laptop's IP: http://<laptop-ip>. This generates HTTP traffic captured by Wireshark.
3. Additionally, from the laptop, ping the Android device IP:

```
Ping 192.168.0.113
```

This generates ICMP packets seen in Wireshark.

Step 3: Capture Traffic in Wireshark

1. Open Wireshark and start capturing on the laptop's network interface connected to the Wi-Fi.
2. Apply a capture filter to limit traffic to the two devices:

```
text
host <laptop-ip> or host <android-ip>
```

3. Perform the browsing and ping steps.
4. Stop capture after enough packets are collected.

Step 4: Analyze Captured Traffic

1. Apply a display filter to focus on interactions between devices:

```
text
ip.addr == <laptop-ip> && ip.addr == <android-ip>
```

2. Filter HTTP traffic:

```
text
http
```

3. Filter ICMP (ping) traffic:

```
text
icmp
```

4. Analyze HTTP packets to see that the server response is unencrypted (HTTP on port 80).
5. Examine ping request and reply packets under ICMP protocol.
6. Note absence of encryption on HTTP traffic, highlighting security risk if sensitive data is sent.

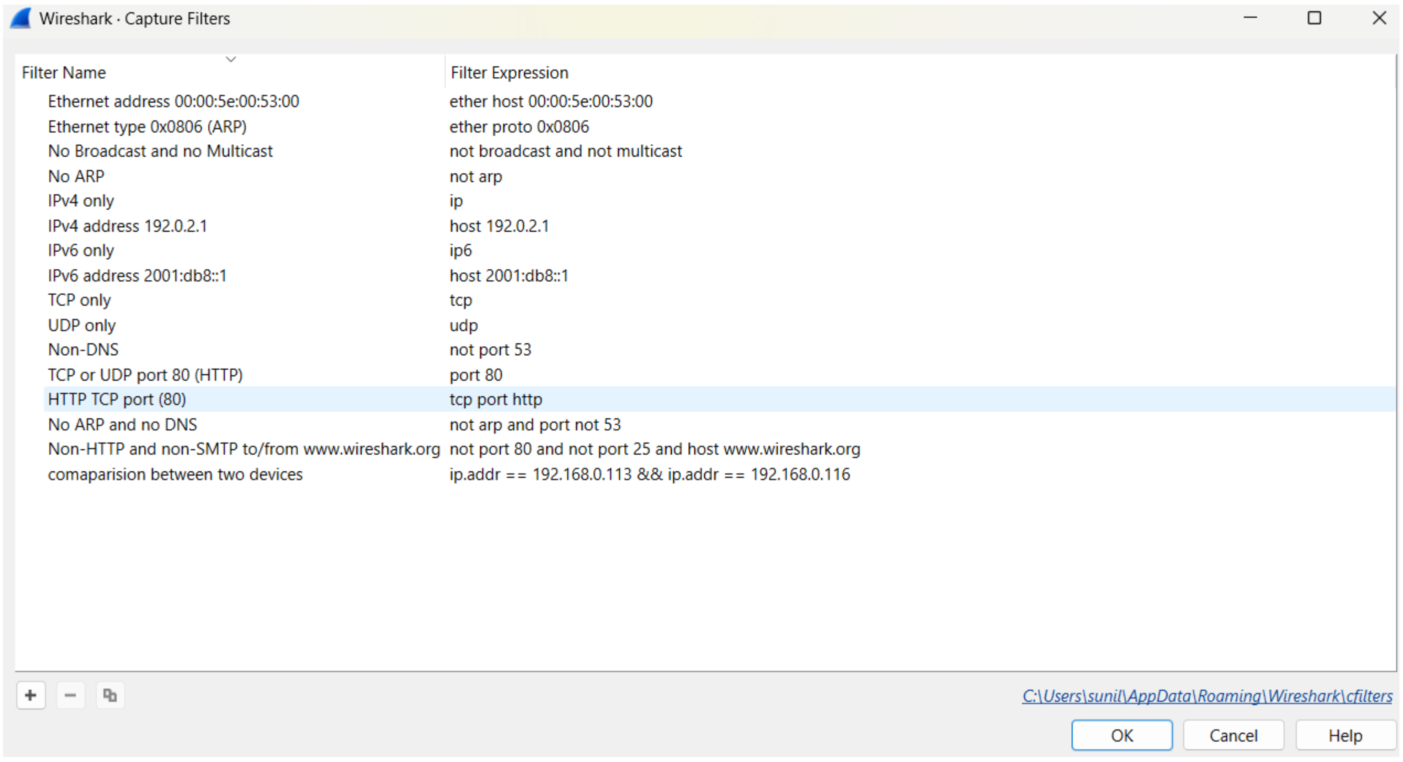
Findings

- HTTP communication between the Android phone and laptop is unencrypted, visible in plaintext to any network sniffer.
- ICMP ping packets are seen as expected, confirming connectivity.
- Potential security risk: Any login or sensitive data sent over HTTP can be intercepted.

Conclusion

This practical exercise illustrates the exposure of unencrypted HTTP traffic and the nature of ICMP packets during device communication over Wi-Fi. Securing IoT device interactions with encrypted protocols like HTTPS or VPN tunnels is advisable to protect privacy.

traffic between two devices.pcapng



Wireshark · Capture Filters

Filter Name	Filter Expression
Ethernet address 00:00:5e:00:53:00	ether host 00:00:5e:00:53:00
Ethernet type 0x0806 (ARP)	ether proto 0x0806
No Broadcast and no Multicast	not broadcast and not multicast
No ARP	not arp
IPv4 only	ip
IPv4 address 192.0.2.1	host 192.0.2.1
IPv6 only	ip6
IPv6 address 2001:db8::1	host 2001:db8::1
TCP only	tcp
UDP only	udp
Non-DNS	not port 53
TCP or UDP port 80 (HTTP)	port 80
HTTP TCP port (80)	tcp port http
No ARP and no DNS	not arp and port not 53
Non-HTTP and non-SMTP to/from www.wireshark.org	not port 80 and not port 25 and host www.wireshark.org
comaparision between two devices	ip.addr == 192.168.0.113 && ip.addr == 192.168.0.116

C:\Users\sunil\AppData\Roaming\Wireshark\cfilters