The plan for the deception lab.

**Ubuntu Server installation** (step-by-step from the welcome screen)

1. **Basic server setup** (updates, users, SSH)
2. **Cowrie honeypot installation** (explained fully)
3. **Suricata IDS installation & config** (with rule explanations)
4. **Telegram alerts integration** (why and how it works)
5. **Testing the deception lab** (simulate an attacker)
6. **Final touches** (auto-start services, logs, security tips)
7. **Export a detailed Word document** with all explanations and commands.

## ****Step 1. Install Ubuntu Server (Welcome Screen → Login)****

When you boot into the Ubuntu Server ISO, here’s what you’ll see and what each choice means:

1. **Language Selection**
   * Pick **English** (or your preference).
   * This only affects the installation interface and default system messages.
2. **Keyboard Configuration**
   * Choose your layout (**English (US)** is default).
   * If you’re unsure, keep it default — you can change later.
3. **Network Configuration**
   * Since the honeypot needs to be **reachable on our LAN**, choose the network interface and set it to **DHCP** for now (automatic IP assignment).
   * You can later change to static IP for consistency.
4. **Proxy**
   * Leave blank unless your network requires a proxy to access the internet.
5. **Mirror**
   * Keep default (archive.ubuntu.com) — used for downloading updates.
6. **Storage Configuration**
   * Choose **Use entire disk** (since this VM is dedicated to Cowrie/Suricata).
   * Accept defaults unless you need custom partitions.
7. **User Setup**
   * **Your name:** Goodenough (or anything — just descriptive)
   * **Server name:** honeypotter (important for identification, I chose to use honeypotter)
   * **Username:** Good (this is your primary login account)
   * **Password:** Choose something secure. However anything would do since its a lab.
8. **SSH Setup**
   * Enable **Install OpenSSH server**.
   * This lets you connect remotely from your attacker machine or Kali later.
9. **Featured Server Snaps**
   * Skip for now (we’ll install what we need manually).

Once installation completes and you reboot, you’ll log in with:

text

Username: Good

Password: [your password]

## ****Step 2. Basic Server Setup****

After logging in, first update the system:

sudo apt update && sudo apt upgrade -y

**Why?**

* apt update refreshes the package list (so your system knows the latest software versions available).
* apt upgrade installs all available updates.
* The -y flag auto-confirms prompts.

Check your network interface name:

ip a

You’ll see something like enp0s3. Remember it, Suricata will need it.

## ****Step 3. Installing Cowrie Honeypot****

**Why Cowrie?**  
Cowrie is a medium-interaction honeypot that simulates an SSH and Telnet service, capturing attacker commands and files.

### ****Create Cowrie User****

We don’t want to run Cowrie as root for safety:

sudo adduser --disabled-password --gecos "" cowrie

**Explanation:**

--disabled-password means no password login (service account only).

--gecos "" skips full name and extra details prompts.

### ****Install Dependencies****

sudo apt install git python3 python3-venv python3-pip libssl-dev libffi-dev build-essential -y

**Explanation:**

* git: to clone Cowrie’s repository.
* python3-venv : for creating isolated Python environments.
* python3-pip : Python package installer.
* libssl-dev, libffi-dev, build-essential -- required to compile certain Python packages.

### ****Download Cowrie****

sudo -u cowrie -H bash -c "cd /home/cowrie && git clone https://github.com/cowrie/cowrie.git"

**Explanation:**

* “sudo -u cowrie” runs the command as the cowrie user.
* Clones Cowrie into /home/cowrie/cowrie.

### ****Set Up Python Environment****

bash

sudo -u cowrie -H bash -c "cd /home/cowrie/cowrie && python3 -m venv cowrie-env"

sudo -u cowrie -H bash -c "source /home/cowrie/cowrie/cowrie-env/bin/activate && pip install --upgrade pip && pip install -r requirements.txt"

**Explanation:**

* Creates a virtual environment cowrie-env for Cowrie.
* Installs Cowrie’s Python dependencies from requirements.txt.

### ****Configure Cowrie****

Copy default config:

sudo -u cowrie cp /home/cowrie/cowrie/etc/cowrie.cfg.dist /home/cowrie/cowrie/etc/cowrie.cfg

### ****Start Cowrie****

sudo -u cowrie -H bash -c "cd /home/cowrie/cowrie && bin/cowrie start"

Check status:

bash

sudo -u cowrie -H bash -c "cd /home/cowrie/cowrie && bin/cowrie status"

## ****4.2 Installing Suricata****

Run:

bash

sudo apt install suricata -y

**Explanation:**

* “apt install suricata” gets the latest version from Ubuntu’s package repositories.
* -y confirms installation automatically.

## ****4.3 Checking Network Interfaces****

We need to know which interface Suricata should monitor:

ip a

Look for the interface with your honeypot’s IP address (e.g., 192.168.x.x).  
It’s usually named ens33, eth0, or similar.

**Example output snippet:**

makefile

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER\_UP> ...

inet 192.168.1.50/24 ...

Here, enp0s3 is the interface we’ll use.

## ****4.4 Backing Up Suricata Config****

Before making changes:

sudo cp /etc/suricata/suricata.yaml /etc/suricata/suricata.yaml.bak

**Explanation:**  
If anything breaks, we can restore the original config.

## ****4.5 Editing Suricata Config****

Open the config:

sudo nano /etc/suricata/suricata.yaml

Find:

yaml

af-packet:

- interface: eth0

Replace eth0 with your interface name (e.g., ens33).

**Set Logging Location**  
Still in suricata.yaml, check:

yaml

default-log-dir: /var/log/suricata/

This is where Suricata will store alerts (eve.json is the key file we’ll later parse for Discord alerts).

## ****4.6 Testing Suricata****

Run:

sudo suricata -T -c /etc/suricata/suricata.yaml -v

**Explanation:**

* -T = test mode (checks config for errors).
* -c specifies the config file.
* -v = verbose output.

If it says **Configuration provided was successfully loaded**, you’re good.

However if you face any issues try updating suricata with the commnad “sudo suricata update”

## ****4.7 Starting Suricata in IDS Mode****

Run:

sudo systemctl enable suricata

sudo systemctl start suricata

sudo systemctl status suricata

**Explanation:**

* enable = start automatically on boot.
* start = run now.
* status = check it’s active.

## ****4.8 Where Alerts Will Appear****

Suricata logs alerts to:

/var/log/suricata/eve.json

You can watch them live:

sudo tail -f /var/log/suricata/eve.json

✅ At this stage:

* Cowrie is running (trap).
* Suricata is running (network watcher).
* Both are on the same VM with your network bridged

# ****Step 5. Adding Discord Alerts to Suricata****

## ****5.1 Why Discord?****

* **Real-time**: You don’t have to keep looking at the server logs.
* **Portable**: You’ll get alerts anywhere, even on your phone.
* **Secure**: Alerts go straight to you, no open dashboard for attackers to find.

## 1️⃣ Get a Discord webhook URL:

• In Discord, right-click the target channel → Edit Channel → Integrations → Webhooks → New Webhook.

• Name it (e.g., Honeypot Alerts) and copy the Webhook URL.

2️⃣ Create the Discord alert script:

sudo nano /usr/local/bin/discord\_alert.sh

#!/bin/bash

WEBHOOK\_URL="YOUR\_DISCORD\_WEBHOOK\_URL"

MESSAGE="$1"

curl -H "Content-Type: application/json" \

-X POST \

-d "{\"content\": \"$MESSAGE\"}" \

$WEBHOOK\_URL

Replace YOUR\_DISCORD\_WEBHOOK\_URL with your actual URL.

3️⃣ Make it executable:

sudo chmod +x /usr/local/bin/discord\_alert.sh

## *****You could also use Telegram if you want, but I used Discord.*****

## ****5.2 Create a Telegram Bot****

1. **Open Telegram** and search for **@BotFather**.
2. Start a chat and type:

/start

/newbot

1. BotFather will ask for:
   * **Name**: e.g., Honeypot Alerts
   * **Username** (must end with “bot”): e.g., HoneypotNotifierBot
2. You’ll get a **Bot Token** (looks like 123456789:ABC...XYZ).  
   **Keep this safe** — it’s like a password for your bot.

## ****5.3 Get Your Chat ID****

* Open Telegram and search for **@userinfobot**.
* Type /start and it will reply with your **Chat ID** (a number like 987654321).

## ****5.4 Make the Telegram Alert Script****

We’ll make a shell script that Suricata can call to send alerts.

sudo nano /usr/local/bin/telegram\_alert.sh

Paste:

#!/bin/bash

BOT\_TOKEN="YOUR\_TELEGRAM\_BOT\_TOKEN"

CHAT\_ID="YOUR\_CHAT\_ID"

MESSAGE="$1"

curl -s -X POST https://api.telegram.org/bot${BOT\_TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d text="$MESSAGE"

Replace:

* “YOUR\_TELEGRAM\_BOT\_TOKEN” → Your bot token from BotFather.
* “YOUR\_CHAT\_ID” → The number from @userinfobot.

Make executable:

sudo chmod +x /usr/local/bin/telegram\_alert.sh

**Test it manually:**

/usr/local/bin/telegram\_alert.sh "Test alert from honeypot"

You should see a message pop up in Telegram.

## ****5.5 Monitor Suricata Alerts and Send to Telegram****

We’ll make a script to “watch” /var/log/suricata/eve.json for any alert entries.

CopyEdit

sudo nano /usr/local/bin/suricata\_watch.sh

Paste:

#!/bin/bash

tail -Fn0 /var/log/suricata/eve.json | \

while read line; do

if echo "$line" | grep -q '"alert"'; then

ALERT\_MSG=$(echo "$line" | jq -r '.alert.signature')

/usr/local/bin/telegram\_alert.sh "Suricata Alert: $ALERT\_MSG"

fi

done

Make executable:

sudo chmod +x /usr/local/bin/suricata\_watch.sh

## ****5.6 Install jq (needed for extracting alert text)****

sudo apt install jq -y

jq lets us pull just the “alert name” from Suricata’s JSON logs.

## ****5.7 Run the Alert Watcher****

nohup /usr/local/bin/suricata\_watch.sh &

* nohup keeps it running even if you close the terminal.
* The & puts it in the background.

## ****5.8 Test the Alerts****

From another machine, try to:

nmap -A <honeypot-ip>

If Suricata rules detect the scan, you should get a Telegram alert like:

Suricata Alert: ET SCAN Nmap Scripting Engine User-Agent Detected

✅ At this point:

* Cowrie catches attacker commands.
* Suricata detects and logs suspicious network behavior.
* You get instant Telegram notifications.