# GOODENOUGH MINI SIEM LAB

THREAT HUNTING LAB.

**Short note before you start:** I wrote commands to be copy‑pasted on your Ubuntu server(my test VM). Where a command requires you to change a value (interface name, IP, webhook URL), I called that out. Read the explanation lines below every command before running it. Also I made sure to attach the the config files in this repo. I would advice against changing or modifying them too much. The suricata and zeek config files are actually for fluent-bit, do not use them on the main suricata .yml and zeek files.

# 1) Goals & architecture

We want a small SOC lab (Ulithak) that:  
• Runs Suricata to IDS/IPS the network traffic and emit alerts as structured JSON (eve.json);  
• Runs Zeek (network-analysis) to provide rich protocol logs;  
• Runs Fluent Bit as the lightweight log shipper to collect Suricata (eve.json), Zeek JSON logs, and other host logs;  
• Sends those logs into OpenSearch (no-login in this lab) and visualizes in OpenSearch Dashboards.

This guide wires those components end-to-end and explains each command.

# 2) Assumptions & prerequisites

* Fresh Ubuntu server (22.04 or 24.04 recommended). Commands assume sudo is available.
* Enough resources: min 2 vCPU, 4 GB RAM (4+ GB recommended for Suricata/Zeek/OpenSearch).
* You have bridged networking on the VM or a network where traffic can be observed.
* You are comfortable running commands as a sudo user (I will call out when sudo is required).
* Know your primary network interface name (we’ll run ip a below to confirm).

# 3) Quick networking checks

1. Update system package lists and upgrade, always first:

sudo apt update && sudo apt -y upgrade

*Explanation*: apt update refreshes the package catalog; apt upgrade -y brings installed packages up-to-date and answers yes automatically.

1. Check your network interfaces and IP addresses (we’ll use this to set Suricata/Zeek capture interface):

ip -br a

*Explanation*: ip -br a prints names and addresses in a concise format; identify the interface that holds your VM’s IP (e.g., enp0s3, eth0, ens33).

Write that interface name down, I’ll reference it YOUR\_IF later.

# 4) Suricata install, configure, test

We’ll install from the Ubuntu repos (simple), back up config, point Suricata to the interface and enable EVE JSON output so Fluent Bit can pick it up.

## 4.1 Install Suricata

sudo apt update  
sudo apt install suricata -y

*Explanation*: Installs Suricata from your APT repos. This is simplest for a lab. If you need the latest Suricata for production, use the OISF repository but repo installs are fine for learning.

**Edit the config file and set which interface to listen on.**

Always remember to make a copy of every config file before touching it, else you’ll end up having to reinstall a lot of things. Common mistake I made a lot.

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

and now we can edit the main filename

sudo nano /etc/suricata/suricata.yaml

Find:

yaml

af-packet:

- interface: eth0

Replace eth0 with your interface name (e.g., enp0s3).

Still in suricata.yaml, check:

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).

Enable and start the daemon:

sudo systemctl enable suricata  
sudo systemctl status suricata --no-pager

*Explanation*: enable --now makes Suricata autostart and starts immediately. status shows its current running state.

## 4.2 Important backup before edits

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

*Explanation*: Always keep a backup of the original config so you can revert.

## 4.3 Detect the interface to monitor (replace in config)

ip a # find interface name that has your IP, e.g., enp0s3

sudo nano /etc/suricata/suricata.yaml

Edit suricata.yaml and set the capture interface under either af-packet (recommended for Linux) or pcap mode. Example snippet to change:

af-packet:  
 - interface: enp0s3 # <--- replace enp0s3 with your interface name  
 threads: 4  
 cluster-id: 99  
 cluster-type: cluster\_flow

*Explanation*: This tells Suricata which NIC to listen on and how many threads to use for the capture engine.

Also confirm the default log directory and EVE JSON output:

default-log-dir: /var/log/suricata  
outputs:  
 - eve-log:  
 enabled: yes  
 filetype: regular  
 filename: eve.json  
 types: ["alert", "http", "dns", "tls", "flow", "fileinfo"]

*Explanation*: EVE JSON (eve.json) is the structured JSON file Suricata writes; Fluent Bit can tail this file.

## 4.4 Update rules (Emerging Threats) and suricata-update

Use suricata-update to fetch and keep rules up to date.

sudo suricata-update

*Explanation*: suricata-update downloads rule sources (EmergingThreats, ETPro if configured) and updates the rule files under /etc/suricata/rules/. After running, suricata-update updates the local rule files used by the engine.

If you want to download the Emerging Threats open rules manually (explicit example):

cd /tmp  
curl -LO https://rules.emergingthreats.net/open/suricata-5.0/emerging.rules.tar.gz  
sudo tar -xvzf emerging.rules.tar.gz -C /etc/suricata/rules --strip-components=0

*Explanation*: manual rule install alternative. Prefer suricata-update normally. (*Please note that I didn’t actually use this manual approach, however, while researching, I stumbled upon it and and just saved it for just in case. I didn’t need it at all, but I decided to include it in here anyway.*)

## 4.5 Test the Suricata configuration

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

*Explanation*: -T is test mode (check config). If you see Configuration provided was successfully loaded, proceed.

## 4.6 Start/Restart Suricata after config changes

sudo systemctl restart suricata  
sudo systemctl status suricata --no-pager

## 4.7 Where to look for alerts

Open the EVE JSON file (Suricata writes here by default):

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

*Explanation*: tail -f streams new JSON lines as they arrive. You can use jq to parse fields:

sudo apt install jq -y  
sudo tail -n 200 /var/log/suricata/eve.json | jq 'select(.alert) | {time: .timestamp, sig: .alert.signature, src: .src\_ip, dst: .dest\_ip}'

*Explanation*: This extracts time, signature, and IPs for recent alerts.

# 5) Zeek install, configure, produce JSON logs

Zeek does deep network analysis and gives protocol-level logs (conn, http, dns, files, etc.). We will install from source (good for lab) and configure JSON output so Fluent Bit can parse Zeek logs easily.

The sample you gave is a correct starting point. I expanded and clarified the steps and added a few dependency checks.

## 5.1 Install build dependencies

sudo apt update  
sudo apt install -y cmake make gcc g++ flex bison libpcap-dev libssl-dev python3 python3-pip libgeoip-dev libmaxminddb-dev libjemalloc-dev

*Explanation*: Zeek needs C/C++ build tools, packet capture libs, crypto libraries, and optional libs for GEOIP and memory optimizations. If a package doesn’t exist on your distro, install the equivalent.

## 5.2 Download and build Zeek (under /opt)

cd /opt  
sudo git clone --recursive https://github.com/zeek/zeek.git  
cd zeek  
sudo ./configure  
sudo make -j$(nproc)  
sudo make install

*Explanation*: Cloning with --recursive ensures submodules are pulled. ./configure prepares the build; make -j$(nproc) compiles using all CPUs; make install places Zeek binaries under /usr/local/zeek (or /opt/zeek depending on build\_local settings).

Note that, the make process really takes long, I recommend adding the j$(nproc) it would allow your system to use all available CPUs for the process. It involves C and C files are heavy, especially if you’re using a virtual machine like me.

## 5.3 Add Zeek to PATH

echo 'export PATH=/usr/local/zeek/bin:$PATH' >> ~/.bashrc  
source ~/.bashrc

*Explanation*: Makes zeek, zeekctl, and zeek-cut available in your shell.

## 5.4 Basic Zeek control and deploy

sudo /usr/local/zeek/bin/zeekctl check  
sudo /usr/local/zeek/bin/zeekctl deploy

*Explanation*: zeekctl check verifies configuration; zeekctl deploy starts the Zeek node(s).

To run Zeek directly on an interface (quick test):

sudo /usr/local/zeek/bin/zeek -i YOUR\_IF

*Explanation*: Runs Zeek in the foreground capturing live traffic on YOUR\_IF.

## 

## 5.5 Where Zeek writes logs

Default: /usr/local/zeek/logs/current (conn.log, http.log, dns.log, files.log). If JSON is enabled, files will be .json. Use tail -f:

sudo tail -f /usr/local/zeek/logs/current/conn.log

*Explanation*: Watch Zeek logs in real time.

# 6) Fluent Bit install script and configuration

We will use the fluent-bit install script (curl installer), then configure Fluent Bit to collect Suricata eve.json and Zeek JSON logs and send them to OpenSearch.

## 6.1 Install Fluent Bit using the official install script (one-liner)

**Preferred (convenience single line, will add repository and keys automatically):**

curl -s https://raw.githubusercontent.com/fluent/fluent-bit/master/install.sh | sudo bash  
# then install package (depending on distro packaging name):  
sudo apt-get update

sudo apt-get install -y fluent-bit

*Explanation*: The script sets up the Fluent Bit package repository and GPG keys for you. Some distros/packages use package name td-agent-bit (Treasure Data variant) or fluent-bit depending on the OS. Use whichever is available for your distro. If you installed via td-agent-bit, the main config file is /etc/td-agent-bit/td-agent-bit.conf. If fluent-bit, config is /etc/fluent-bit/fluent-bit.conf.

Note: Always review any install script before piping into bash. For a lab you may trust the official repo URL above.

## 6.2 Fluent Bit service controls

sudo systemctl enable fluent-bit  
sudo systemctl status fluent-bit -l --no-pager

*Explanation*: Start and enable the shipper daemon.

## 6.3 Fluent Bit config: inputs, parser, outputs

Below are example snippets. I recommend creating a dedicated conf: /etc/td-agent-bit/conf.d/ultihak.conf and include it from the main file. But many distros use a single main file — I include full blocks so you can copy/paste.

## **Understanding Fluent Bit Config Structure**

Fluent Bit has **one main config file** and **many small per-service configs**:

* **Main config file** → /etc/fluent-bit/fluent-bit.conf  
  🔹 Controls **global settings** like log level, parsers, and where to load extra configs.  
  🔹 **DO NOT put Zeek or Suricata outputs here**.  
  🔹 Keep it clean and minimal.
* **Extra configs folder** → /etc/fluent-bit/conf.d/  
  🔹 Each file here handles **one data source** (e.g., Zeek, Suricata).  
  🔹 This is where we define **inputs** and **outputs**.  
  🔹 Much easier to manage.

## **What You Should NOT Touch**

Inside /etc/fluent-bit/fluent-bit.conf:

* **DON’T** add [OUTPUT] here.
* **DON’T** define Zeek or Suricata inputs here.
* **DON’T** edit any default parsers unless we need custom ones (we won’t for now).

Just leave it **minimal** like this:

[SERVICE]

flush 5

daemon Off

log\_level info

parsers\_file parsers.conf

plugins\_file plugins.conf

http\_server On

http\_listen 0.0.0.0

http\_port 2020

@include /etc/fluent-bit/conf.d/\*.conf

### 

Everything related to **Zeek** and **Suricata** goes into the /etc/fluent-bit/conf.d/ folder.  
We’ll create two files:

### **Zeek Config →** /etc/fluent-bit/conf.d/zeek.conf

[INPUT]

Name tail

Path /var/log/zeek/\*.log

Parser json

Tag zeek

[OUTPUT]

Name opensearch

Match zeek

Host 127.0.0.1

Port 9200

Index zeek-logs

Suppress\_Type\_Name On

Logstash\_Format Off

### **Suricata Config →** /etc/fluent-bit/conf.d/suricata.conf

[INPUT]

Name tail

Path /var/log/suricata/eve.json

Parser json

Tag suricata

[OUTPUT]

Name opensearch

Match suricata

Host 127.0.0.1

Port 9200

Index suricata-logs

Suppress\_Type\_Name On

Logstash\_Format Off

## 6.4 Fluent Bit troubleshooting

* Check logs:  
  sudo journalctl -u fluent-bit -f

*Explanation*: watch Fluent Bit runtime logs for parser errors or HTTP failures to OpenSearch.

* If parser errors occur, inspect the first few lines of the file and try jq locally to verify valid JSON.

head -n 5 /var/log/suricata/eve.json | jq .

# 7) OpenSearch & Dashboards install, disable security for lab, configure

We will follow a tarball install. I expanded with best practices and the specific extra steps needed to *remove Dashboards’ security plugin* if you want no login screen.

## 7.1 Download & extract OpenSearch (example uses 2.15.0 as you provided)

cd /usr/local  
sudo wget https://artifacts.opensearch.org/releases/bundle/opensearch/2.15.0/opensearch-2.15.0-linux-x64.tar.gz  
sudo tar -xvzf opensearch-2.15.0-linux-x64.tar.gz  
sudo mv opensearch-2.15.0 opensearch

*Explanation*: Downloads and unpacks the OpenSearch tarball into /usr/local/opensearch.

## 7.2 Create opensearch user and assign permissions

sudo useradd -r -m -s /usr/sbin/nologin opensearch  
sudo chown -R opensearch:opensearch /usr/local/opensearch

*Explanation*: Creates a service user and assigns directory ownership — safer than running as root.

## 7.3 Configure OpenSearch (disable security plugin for local lab)

Edit /usr/local/opensearch/config/opensearch.yml and ensure the essential cluster & networking settings are present. Example final file fragments:

cluster.name: mini-siem  
node.name: ultihak  
network.host: 0.0.0.0 # Accept traffic from the host network — lock this down on real systems  
http.port: 9200  
discovery.type: single-node  
plugins.security.disabled: true # DISABLE SECURITY PLUGIN (lab only)  
  
# JVM options are in /usr/local/opensearch/config/jvm.options

*Explanation & caution*: plugins.security.disabled: true tells OpenSearch to not enforce its Security plugin. This is *only* for labs and local usage. Disabling security exposes the cluster protect it through firewalls or private networks. In some OpenSearch versions you may also need to remove demo config or remove the Dashboards plugin, see note below.

## 7.4 Start OpenSearch as the opensearch user

sudo -u opensearch /usr/local/opensearch/bin/opensearch &  
# or to run foreground to debug:  
# sudo -u opensearch /usr/local/opensearch/bin/opensearch

*Explanation*: Running as the opensearch user avoids root-run processes and keeps file permissions sane. Start in background with & for lab testing; later we convert to systemd unit.

Test connectivity:

curl -s http://localhost:9200 | jq .

*Explanation*: Should return cluster name and version info JSON. If it fails, check logs in /usr/local/opensearch/logs and journal.

## 7.5 Install OpenSearch Dashboards (tarball)

cd /usr/local  
sudo wget https://artifacts.opensearch.org/releases/bundle/opensearch-dashboards/2.15.0/opensearch-dashboards-2.15.0-linux-x64.tar.gz  
sudo tar -xvzf opensearch-dashboards-2.15.0-linux-x64.tar.gz  
sudo mv opensearch-dashboards-2.15.0 opensearch-dashboards  
sudo chown -R opensearch-dashboards:opensearch-dashboards /usr/local/opensearch-dashboards || true

*Explanation*: Downloads and extracts Dashboards. Ownership depends on your OS packaging; (tarball installs often run under your user for dev use).

Edit /usr/local/opensearch-dashboards/config/opensearch\_dashboards.yml and set:

server.host: "0.0.0.0"  
opensearch.hosts: ["http://localhost:9200"]  
# remove or comment out any security config lines if you disabled security in OpenSearch

### Important: Remove Dashboards security plugin when OpenSearch security is disabled

If OpenSearch security plugin is disabled or removed, the Dashboards side security plugin must also be removed; otherwise you’ll get login screens or errors. Remove the Dashboards security plugin (tarball install):

cd /usr/local/opensearch-dashboards/bin  
sudo ./opensearch-dashboards-plugin remove securityDashboards

*Explanation*: This removes the Dashboards security plugin so it will operate against an unprotected OpenSearch cluster.

Start Dashboards (foreground for first-run troubleshooting):

cd /usr/local/opensearch-dashboards  
sudo -u opensearch-dashboards bin/opensearch-dashboards  
# or run as your user for testing: ./bin/opensearch-dashboards

Open your browser and visit http://<server-ip>:5601 (or http://localhost:5601 on the server) and verify Dashboards loads without a login.

**If Dashboards still shows a login** you likely need to:  
• Confirm plugins.security.disabled: true is present in OpenSearch and OpenSearch was restarted.  
• Confirm you removed the securityDashboards plugin from the Dashboards install.

# 8) Integration: Fluent Bit → OpenSearch index patterns, templates, Dashboards

With Fluent Bit sending logs to OpenSearch, create index patterns and basic dashboards for Suricata & Zeek.

## 8.2 Create index patterns (OpenSearch Dashboards UI)

1. Open Dashboards → Dashboards Management → Index Patterns.
2. Create pattern suricata-\*, choose @timestamp as the time field (or whatever field you used).
3. Repeat for zeek-\*.

## 8.3 Build a few quick visualizations

* Alerts over time (date histogram on @timestamp, split by alert.signature).
* Top source IPs (terms aggregation on src\_ip).
* Top destination ports or URLs (for Zeek/http logs).

You can save these visualizations and put them into a Dashboard for monitoring.

# 9) Testing: generate traffic to trigger Suricata & Zeek

Use common scanners and probes from another host on your network to generate alerts and entries.

* Basic port scan (should create Suricata scan alerts):

nmap -sS -Pn -T4 <honeypot-ip>

* Application HTTP probe (to create HTTP logs):

curl -v http://<honeypot-ip>/nonexistent

* Try a more obvious signature trigger (Be safe in your lab):

curl -A "Nmap Scripting Engine" http://<honeypot-ip>

Then watch the logs:

sudo tail -f /var/log/suricata/eve.json | jq 'select(.alert) | {time: .timestamp, sig: .alert.signature, src: .src\_ip}'  
sudo tail -f /usr/local/zeek/logs/current/conn.log  
sudo journalctl -u td-agent-bit -f

Open Dashboards and check the suricata-\* index visualizations; you should see spikes and signature names.

# 10) Troubleshooting: mistakes I made earlier and how to avoid them

I reviewed previous run-throughs and corrected these common mistakes in this guide:

* **Filebeat vs Fluent Bit confusion:** Don’t try to convert JSON to NDJSON manually for Suricata; Suricata already produces newline-delimited JSON (EVE JSON). Fluent Bit tail + Parser json handles it directly. Avoid creating NDJSON files manually.
* **OpenSearch security disabled but Dashboards still prompting for login** - you must remove the Dashboards securityDashboards plugin or use the correct env var for Docker images. This guide shows the opensearch-dashboards-plugin remove securityDashboards step.
* **Fluent Bit parser errors:** often caused by multi-line JSON or broken lines. Always verify the first few lines via head and jq before shipping.
* **Wrong interface set for Suricata/Zeek:** check with ip -br a and place the exact interface name in suricata.yaml and zeek node.cfg. Mismatched interface names silently cause “no traffic seen” problems.
* **Permissions:** if Fluent Bit cannot read eve.json, check file ownership: sudo chown td-agent-bit:td-agent-bit /var/log/suricata/eve.json or use sudo setfacl to grant the service user read access.
* **Index mapping problems in Dashboards:** create index templates ahead of time so IPs are mapped to ip and dates parsed as date (see examples above).

# 11) Persistence & systemd (services & watchers)

**OpenSearch & Dashboards as systemd services** — you can create simple systemd units so services start at boot. Example unit for OpenSearch.

# /etc/systemd/system/opensearch.service  
[Unit]  
Description=OpenSearch  
After=network.target  
  
[Service]  
Type=simple  
User=opensearch  
ExecStart=/usr/local/opensearch/bin/opensearch  
Restart=on-failure  
LimitNOFILE=65536  
  
[Install]  
WantedBy=multi-user.target

For Dashboards and for any custom watcher scripts (e.g., suricata\_watch.sh that sends alerts to Telegram/Discord), create proper systemd units rather than launching with nohup for reliability.

# 12) Security & safety notes (read this)

* Disabling plugins.security is only for **isolated lab environments** on private networks. It is insecure in any production or internet-exposed environment.
* If you disable security, restrict access to OpenSearch/Dashboards with host firewall rules (ufw) or network ACLs. Example to allow only localhost and your admin IP:

sudo apt install ufw -y  
sudo ufw default deny incoming  
sudo ufw allow from 192.168.1.0/24 to any port 9200 proto tcp # adjust to your admin network  
sudo ufw allow 5601/tcp # if you need Dashboards  
sudo ufw enable

* Keep rule updates and signature sources up to date (Suricata) and remove any demo credentials left by packages.

## Useful troubleshooting one-liners

* Check Suricata config:

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

* Check Zeek status & logs:

sudo /usr/local/zeek/bin/zeekctl status  
sudo tail -n 200 /usr/local/zeek/logs/current/conn.log

* Check Fluent Bit journal:

sudo journalctl -u td-agent-bit -f

# Final notes

* I built this guide to be **copy/paste friendly** and to avoid the mistakes from my earlier runs, and trust me, I made a lot of mistakes. I had to nuke everything and start again over and over and over again, more than 10 times. I spent 3 weeks on this lab. I started with elastic search and Kibana, but they took a toll on my ram, and I switched to opensearch and filebeat. Filebeat is hardcoded to work with Elasticsearch. I tried many methods but failed to configure filebeat to work correctly with opensearch. And then I finally switched to Fluent bit. Only God knows how happy I was after I finally got everything working together. There was a lot of frustration for me, and I wanted to even quit, but after some persistence I got it to work. So I made this guide in such a way that you won’t have to suffer like I did.