

CYBERSECURITY 3

Lab 02

Détection d'une attaque SSH avec Wazuh

Corrigé pédagogique détaillé - version formateur et apprenant

Finalité du support

Déployer une stack Wazuh reproductible, ingérer des événements SSH simulés, valider le décodage, déclencher une règle de corrélation et documenter l'incident. Le document distingue le sujet d'origine de sa mise en œuvre corrigée pour une stack Wazuh récente.

Public

Apprenants en cybersécurité / analystes SOC débutants

Durée indicative

4 à 6 heures

Prérequis

Linux, Docker, notions de logs et de détection

Livrables

Capture, règle XML, export JSON, rapport d'incident

1. Lecture critique du sujet

Le Lab 02 demande de détecter une attaque SSH par force brute dans Wazuh, à partir de quarante lignes d'échec d'authentification placées dans un fichier auth.log. Il prévoit une architecture Docker, une règle locale et une vérification dans le Dashboard.

Point important

Le principe pédagogique est pertinent, mais le fichier docker-compose.yml fourni dans le sujet ne déploie pas réellement la stack Wazuh complète. Une installation actuelle comprend au minimum un manager, un indexer et un dashboard séparés. La version corrigée ci-dessous s'appuie donc sur la stack single-node officielle, puis ajoute un agent de laboratoire.

Élément	Rôle	Interprétation pédagogique
Image wazuh/wazuh	Image générique indiquée dans le sujet	À remplacer par le dépôt officiel wazuh-docker qui orchestre manager, indexer et dashboard.
Port 443 sur le manager	Le sujet l'associe au manager	Le HTTPS 443 appartient au Dashboard, pas au manager.
sed manager_address	Modification proposée de ossec.conf	La syntaxe Wazuh est XML ; un remplacement textuel de type YAML ne configure pas correctement l'agent.
frequency/timeframe enfants	Balises placées dans le corps de la règle	Dans la syntaxe Wazuh actuelle, frequency et timeframe sont des attributs de <rule>.
if_sid 5710	Filiation directe	Pour une corrélation d'occurrences, utiliser plutôt <if_matched_sid>5710</if_matched_sid>.
API /alerts	Endpoint supposé dans le sujet	La voie la plus fiable pour le lab est l'export depuis Discover/Dashboard ou la lecture de alerts.json ; les API et index patterns varient selon version.

2. Architecture cible corrigée

La stack centrale est composée de trois services. L'agent de laboratoire lit le fichier simulé et transmet chaque événement au manager. Le manager applique les décodeurs et les règles, puis Filebeat envoie les alertes à l'indexer. Le Dashboard interroge ensuite l'indexer.

Élément	Rôle	Interprétation pédagogique
---------	------	----------------------------

Élément	Rôle	Interprétation pédagogique
Wazuh agent	Collecte auth.log	Lit les lignes au format syslog et les transmet au manager.
Wazuh manager	Analyse et corrélation	Décode sshd, applique les règles natives puis la règle locale.
Wazuh indexer	Stockage et recherche	Indexe les alertes et permet les recherches temporelles.
Wazuh dashboard	Visualisation	Affiche les événements et permet l'export des résultats.

2.1 Ressources minimales

Prévoir au moins 4 cœurs, 8 Go de RAM et 50 Go d'espace libre pour une stack single-node de formation. Sur Linux, la valeur `vm.max_map_count` doit être suffisante pour l'indexer.

```
sudo sysctl -w vm.max_map_count=262144
# Vérification
sysctl vm.max_map_count
```

Résultat attendu : la commande affiche `vm.max_map_count = 262144`. Pour rendre le réglage persistant, ajoutez-le dans `/etc/sysctl.conf` puis rechargez avec `sudo sysctl -p`.

3. Déploiement de la stack Wazuh officielle

3.1 Récupération du dépôt

```
mkdir -p ~/cyberlab/lab02
cd ~/cyberlab/lab02
git clone https://github.com/wazuh/wazuh-docker.git -b v4.14.6
cd wazuh-docker/single-node
```

Élément	Rôle	Interprétation pédagogique
<code>mkdir -p</code>	Crée le répertoire de travail	-p évite une erreur si un parent existe déjà.
<code>git clone</code>	Télécharge le dépôt officiel	-b sélectionne une version déterminée, utile pour la reproductibilité.
<code>single-node</code>	Choisit le scénario pédagogique	Un exemplaire de chaque composant central est suffisant pour ce lab.

3.2 Génération des certificats

```
docker compose -f generate-indexer-certs.yml run --rm generator
```

Cette commande lance temporairement le générateur de certificats. --rm supprime le conteneur une fois le travail terminé. Les certificats restent dans le répertoire monté sur l'hôte.

3.3 Démarrage

```
docker compose up -d
docker compose ps
docker compose logs --tail=50 wazuh.dashboard
```

Le premier démarrage peut prendre plusieurs minutes. Le Dashboard peut afficher temporairement « server is not ready yet » tant que l'indexer n'est pas disponible.

Validation attendue

docker compose ps doit montrer wazuh.manager, wazuh.indexer et wazuh.dashboard dans un état actif. Le Dashboard est accessible sur <https://localhost> avec admin / SecretPassword dans la configuration de démonstration. Changez ce mot de passe dans tout environnement non pédagogique.

4. Préparation des événements SSH simulés

Le sujet d'origine écrit des dates fixes et utilise 12:00:040 pour les dernières itérations, ce qui n'est pas une seconde valide. Le script suivant génère des horodatages valides et concentre les échecs d'une même IP pour garantir la corrélation.

4.1 Script generate_auth_log.sh

```
#!/usr/bin/env bash
set -euo pipefail

LOG_FILE="auth.log"
: > "$LOG_FILE"

for i in $(seq 1 40); do
    sec=$(printf '%02d' $(( (i - 1) % 60 )))
    user=$(printf 'user%02d' "$i")
    port=$((5000 + i))
    printf 'Jul 21 12:00:%s ubuntu sshd[12345]: Failed password for invalid user %s from 192.168.0.10 port %s ssh2\n' \
        "$sec" "$user" "$port" >> "$LOG_FILE"
done

printf 'Jul 21 12:01:00 ubuntu sshd[12345]: Accepted password for validuser from 192.168.0.20 port 5000 ssh2\n' >> "$LOG_FILE"
```

4.2 Explication ligne par ligne

Élément	Rôle	Interprétation pédagogique
#!/usr/bin/env bash	Sélectionne Bash	Plus portable qu'un chemin Bash codé en dur.
set -euo pipefail	Active le mode strict	Arrête le script en cas d'erreur, variable absente ou pipeline défaillant.
: > auth.log	Vide ou crée le fichier	Évite d'empiler des événements lors de plusieurs essais.

Élément	Rôle	Interprétation pédagogique
seq 1 40	Produit 40 itérations	Crée quarante échecs d'authentification.
printf %02d	Force deux chiffres	Produit des secondes 00, 01, 02... valides.
192.168.0.10	IP source constante	Permet d'interpréter les événements comme une rafale provenant d'une même source.
Accepted password	Ajoute un succès	Permet de discuter d'un possible succès après la rafale, sans l'inclure dans la règle d'échec.

```
chmod +x generate_auth_log.sh
./generate_auth_log.sh
wc -l auth.log
head -n 3 auth.log
tail -n 3 auth.log
```

Résultat attendu : 41 lignes, dont 40 échecs et 1 succès.

5. Mise en place d'un agent de laboratoire

Pour une exécution robuste, le moyen le plus simple consiste à lancer un conteneur Ubuntu, y installer l'agent, l'enrôler puis configurer la collecte du fichier. Le nom ou l'adresse du manager doit être résolvable depuis le réseau Docker.

Choix pédagogique

Le corrigé conserve le principe « agent conteneurisé » du sujet. En production, un agent est normalement installé directement sur l'hôte surveillé.

5.1 Création d'un réseau partagé

```
docker network create lab02-net || true
# Connecter le manager au réseau du lab
docker network connect lab02-net single-node-wazuh.manager-1 || true
```

Le nom exact du conteneur manager peut différer. Utilisez `docker compose ps` pour le relever.

5.2 Lancement du conteneur agent

```
cd ~/cyberlab/lab02
docker run -d --name lab02-agent \
--hostname lab02-agent \
--network lab02-net \
-v "$PWD/auth.log:/var/log/lab02-auth.log:ro" \
ubuntu:22.04 sleep infinity
```

5.3 Installation et enrôlement

```
docker exec -it lab02-agent bash
apt-get update
apt-get install -y curl gnupg apt-transport-https ca-certificates
```

```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH \
| gpg --dearmor -o /usr/share/keyrings/wazuh.gpg
echo 'deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main' \
> /etc/apt/sources.list.d/wazuh.list
apt-get update
WAZUH_MANAGER='single-node-wazuh.manager-1' \
WAZUH_AGENT_NAME='lab02-agent' apt-get install -y wazuh-agent
```

La variable WAZUH_MANAGER indique où joindre le manager durant l'installation. WAZUH_AGENT_NAME fixe un nom lisible dans le Dashboard.

5.4 Configuration de la collecte locale

```
python3 - <<'PY'
from pathlib import Path
p = Path('/var/ossec/etc/ossec.conf')
s = p.read_text()
block = '''
<localfile>
<location>/var/log/lab02-auth.log</location>
<log_format>syslog</log_format>
</localfile>
'''
s = s.replace('</ossec_config>', block + '\n</ossec_config>')
p.write_text(s)
PY
/var/ossec/bin/wazuh-control restart
/var/ossec/bin/wazuh-control status
```

Le bloc <localfile> demande à logcollector de suivre le fichier. Le format syslog est adapté aux lignes de type « Jul 21 12:00:00 ubuntu sshd... ».

6. Tester le décodage avant de créer la règle

Une règle ne peut fonctionner que si l'événement est correctement décodé. Le test unitaire évite de chercher une panne dans le Dashboard alors que le problème est en amont.

```
docker exec -it single-node-wazuh.manager-1 /var/ossec/bin/wazuh-logtest
```

Collez ensuite une ligne d'échec, par exemple :

```
Jul 21 12:00:00 ubuntu sshd[12345]: Failed password for invalid user user01 from 192.168.0.10 port 5001 ssh2
```

Résultat attendu : le décodeur sshd est reconnu, srcip vaut 192.168.0.10 et la règle native 5710 peut apparaître pour un utilisateur inexistant.

Diagnostic

Si Phase 2 ne contient pas srcip, vérifiez que la ligne n'a pas été coupée ou reformattée. Si aucune règle SSH ne matche, testez d'abord une ligne brute issue d'un vrai /var/log/auth.log de la distribution utilisée.

7. Création de la règle locale corrigée

La règle d'origine place frequency et timeframe comme éléments enfants et utilise if_sid. La forme suivante corrèle les occurrences d'une règle précédemment déclenchée.

```
<group name="syslog,sshd,authentication_failed,">
<rule id="100001" level="10" frequency="5" timeframe="60">
<if_matched_sid>5710</if_matched_sid>
<same_source_ip />
<description>SSH brute-force suspecté : plusieurs échecs pour des utilisateurs inexistant depuis $(srcip)</description>
```

```
<mitre>
  <id>T1110</id>
</mitre>
<group>authentication_failures,brute_force,</group>
</rule>
</group>
```

7.1 Signification des éléments

Élément	Rôle	Interprétation pédagogique
id="100001"	Identifiant local	Les IDs personnalisés doivent éviter les IDs natifs ; 100001 convient au lab.
level="10"	Sévérité élevée	Signale un événement nécessitant une investigation rapide.
frequency="5"	Nombre d'occurrences	La règle se déclenche lorsque le moteur atteint le seuil dans la fenêtre.
timeframe="60"	Fenêtre temporelle	Les occurrences doivent se produire dans les 60 secondes.
if_matched_sid	Corrélation	Compte les événements ayant précédemment déclenché la règle 5710.
same_source_ip	Contrainte de contexte	Évite d'agréger artificiellement plusieurs IP sources.
MITRE T1110	Technique associée	Correspond aux tentatives de brute force.
description	Message analyste	\$(srcip) enrichit l'alerte avec l'adresse décodée.

7.2 Installation de la règle

```
docker exec -it single-node-wazuh.manager-1 bash
cat >> /var/ossec/etc/rules/local_rules.xml <<'EOF'

<group name="syslog,sshd,authentication_failed,">
  <rule id="100001" level="10" frequency="5" timeframe="60">
    <if_matched_sid>5710</if_matched_sid>
    <same_source_ip />
    <description>SSH brute-force suspecté : plusieurs échecs pour des utilisateurs inexistants depuis $(srcip)</description>
    <mitre><id>T1110</id></mitre>
    <group>authentication_failures,brute_force,</group>
  </rule>
</group>
EOF

/var/ossec/bin/wazuh-analysisd -t
/var/ossec/bin/wazuh-control restart
```

La commande `wazuh-analysisd -t` valide la syntaxe avant le redémarrage. En cas d'erreur XML, corrigez le fichier avant de relancer le manager.

8. Rejouer les événements

Comme le fichier a été monté avant le démarrage de l'agent, logcollector peut considérer sa position comme déjà lue selon les essais. Pour une démonstration reproductible, ajoutez de nouvelles lignes avec des horodatages courants ou recréez l'agent.

```
for i in $(seq 1 10); do
  now=$(date '+%b %e %H:%M:%S')
  echo "$now ubuntu sshd[23456]: Failed password for invalid user demo$i from 192.168.0.10 port $((6000+i)) ssh2" >> auth-live.log
done
```

Pour utiliser cette variante, montez `auth-live.log` en lecture-écriture ou copiez-le dans le conteneur puis pointez `<location>` vers ce fichier. Le point clé est que les nouvelles lignes doivent être écrites après le démarrage de logcollector.

9. Vérification dans Wazuh Dashboard

Ouvrez `https://localhost`, puis utilisez la vue des événements de sécurité ou Discover selon la version. Filtrez sur `rule.id:100001`. Ajoutez aussi `agent.name:lab02-agent` pour réduire le bruit.

Élément	Rôle	Interprétation pédagogique
<code>rule.id</code>	100001	Confirme que la règle personnalisée a déclenché.
<code>rule.level</code>	10	Indique la criticité attribuée.
<code>data.srcip</code> ou <code>srcip</code>	192.168.0.10	Montre la source de la rafale.
<code>agent.name</code>	lab02-agent	Identifie l'équipement qui a collecté le journal.
<code>full_log</code>	Ligne sshd brute	Permet de conserver le contexte original.
<code>timestamp</code>	Date d'ingestion	À distinguer de l'horodatage présent dans le message syslog.

9.1 Export de la preuve

Méthode recommandée pour ce lab : dans Discover, appliquez le filtre `rule.id:100001`, ouvrez l'événement, puis utilisez l'option d'export CSV/JSON disponible dans votre version. Une preuve brute peut également être extraite depuis le manager :

```
docker exec single-node-wazuh.manager-1 sh -c "grep 'id':100001 /var/ossec/logs/alerts/alerts.json | tail -n 20" > alert-100001.json
```

Le fichier obtenu contient les alertes JSON produites par le manager. Cette méthode ne dépend pas d'un endpoint API susceptible de différer selon les versions.

10. Corrigé des résultats attendus

Élément	Rôle	Interprétation pédagogique
Stack centrale	3 conteneurs actifs	Manager, indexer et dashboard sont opérationnels.
Agent	Visible comme Active	Le manager reçoit les événements du conteneur lab02-agent.
Règle native	5710 observée	Chaque échec pour utilisateur inexistant est reconnu individuellement.
Règle locale	100001, niveau 10	La corrélation signale une rafale depuis 192.168.0.10.
Succès SSH	Événement séparé	La ligne Accepted password ne doit pas être comptée par la règle 100001.
MITRE	T1110	L'alerte est classée dans la technique Brute Force.

10.1 Interprétation analyste

La présence de plusieurs échecs depuis une même IP en moins d'une minute est compatible avec une tentative automatisée. Elle ne prouve pas à elle seule une compromission. Le succès final du fichier simulé provient d'une autre IP dans le corrigé ; il ne doit donc pas être présenté comme la réussite de l'attaquant. Pour démontrer un succès après brute force, utilisez volontairement la même IP et corrélerez les deux types d'événements dans le rapport.

11. Modèle de rapport d'incident corrigé

11.1 Résumé exécutif

Le 21 juillet, le système de supervision Wazuh a détecté une rafale de tentatives d'authentification SSH visant des comptes inexistants. Quarante échecs ont été générés depuis l'adresse 192.168.0.10. La règle personnalisée 100001, de niveau 10, a signalé le dépassement du seuil de cinq occurrences en soixante secondes. Aucun élément du scénario corrigé ne permet d'affirmer qu'un accès a été obtenu par cette adresse.

11.2 Timeline

Heure	Source	Événement	Interprétation
12:00:00	192.168.0.10	Premier échec user01	Début de la rafale
12:00:04	192.168.0.10	Seuil de corrélation atteint	Déclenchement attendu de

Heure	Source	Événement	Interprétation
			100001
12:00:39	192.168.0.10	Quarantième échec	Poursuite de l'activité automatisée
12:01:00	192.168.0.20	Authentification acceptée validuser	Succès distinct ; aucune attribution à la source de la rafale

11.3 Indicateurs

Élément	Rôle	Interprétation pédagogique
192.168.0.10	Source des échecs	IOC de laboratoire ; adresse privée non routable sur Internet.
lab02-agent	Équipement collecteur	Permet de retrouver les événements dans Wazuh.
rule.id 100001	Détection locale	Filtre principal pour retrouver l'alerte corrélée.
T1110	Technique MITRE	Brute Force.

11.4 Recommandations

- Limiter les tentatives SSH avec Fail2ban, sshguard ou un contrôle équivalent.
- Désactiver l'authentification par mot de passe lorsque l'usage de clés est possible.
- Interdire la connexion directe de root et appliquer le principe du moindre privilège.
- Restreindre l'exposition SSH par pare-feu, VPN, bastion ou listes d'autorisation.
- Corréler les échecs avec les succès, les créations de comptes, sudo et les connexions inhabituelles.
- Conserver une synchronisation NTP fiable pour rendre les timelines exploitables.

12. Dépannage

Élément	Rôle	Interprétation pédagogique
Dashboard indisponible	Indexer encore en démarrage	Attendre, consulter docker compose logs et vérifier vm.max_map_count.
Agent absent	Adresse manager ou réseau	Tester la résolution DNS, le port 1515 et les

Élément	Rôle	Interprétation pédagogique
	erroné	logs /var/ossec/logs/ossec.log.
Aucun événement	localfile mal placé ou fichier inchangé	Vérifier ossec.conf, redémarrer l'agent et ajouter une nouvelle ligne après démarrage.
5710 non déclenchée	Format non reconnu	Tester la ligne avec wazuh-logtest et conserver le format syslog attendu.
100001 non déclenchée	Règle mal écrite ou IP variables	Valider avec wazuh-analysisd -t, utiliser if_matched_sid et une IP constante.
Erreur XML	Balise non fermée	Restaurer local_rules.xml et tester avant redémarrage.
Alerte répétée	Seuil atteint plusieurs fois	Ajouter ignore="60" si l'objectif est une seule alerte par minute.

13. Questions de validation et réponses

1. Pourquoi le sujet d'origine ne peut-il pas afficher un Dashboard avec seulement wazuh/wazuh ?

Parce qu'une stack Wazuh moderne sépare le manager, l'indexer et le dashboard. Le manager seul n'héberge pas l'interface web complète.

2. Pourquoi utiliser if_matched_sid au lieu de if_sid ?

if_sid établit une dépendance hiérarchique dans l'analyse du même événement. if_matched_sid sert à corréler plusieurs événements ayant précédemment déclenché une règle.

3. Pourquoi same_source_ip est-il utile ?

Il garantit que le seuil correspond à une même source, plutôt qu'à cinq utilisateurs légitimes distincts ayant chacun échoué une fois.

4. La règle prouve-t-elle une compromission ?

Non. Elle signale une activité compatible avec du brute force. Une compromission exige des preuves complémentaires, notamment un succès corrélé et des actions post-authentification.

5. Pourquoi tester avec wazuh-logtest ?

Pour vérifier séparément le décodage et la logique de règle avant de dépendre de la collecte, du transport, de l'indexation et du Dashboard.

6. Quel est le risque du mot de passe SecretPassword ?

Il s'agit d'un secret de démonstration connu. Il doit être remplacé avant toute exposition ou usage réel.

14. Grille d'évaluation

Critère	Points	Attendus
Préparation hôte et certificats	10	Preuve fournie et explication correcte
Stack Wazuh opérationnelle	15	Preuve fournie et explication correcte
Agent enrôlé et actif	15	Preuve fournie et explication correcte
Fichier de logs valide et reproductible	10	Preuve fournie et explication correcte
Test wazuh-logtest documenté	10	Preuve fournie et explication correcte
Règle 100001 correcte et validée	15	Preuve fournie et explication correcte
Alerte retrouvée et exportée	10	Preuve fournie et explication correcte
Rapport d'incident et timeline	10	Preuve fournie et explication correcte
Analyse critique et recommandations	5	Preuve fournie et explication correcte

Total : 100 points.

15. Checklist apprenant

- ☐ La stack centrale démarre sans erreur bloquante.
- ☐ Le Dashboard est accessible en HTTPS.
- ☐ L'agent lab02-agent apparaît actif.
- ☐ Le fichier contient 40 échecs et 1 succès.
- ☐ Une ligne est reconnue par wazuh-logtest.
- ☐ local_rules.xml est un XML valide.
- ☐ La règle 100001 utilise if_matched_sid, frequency et timeframe.
- ☐ L'alerte 100001 est visible et exportée.
- ☐ Le rapport distingue détection, suspicion et preuve de compromission.
- ☐ Les captures masquent tout secret sensible.

16. Références techniques

Élément	Rôle	Interprétation pédagogique
Sujet fourni	Lab 02 - Détection d'une attaque SSH avec Wazuh, version	Consultée pour la version corrigée du lab.

Élément	Rôle	Interprétation pédagogique
	2026063001.	
Déploiement Docker Wazuh	https://documentation.wazuh.com/current/deployment-options/docker/wazuh-container.html	Consultée pour la version corrigée du lab.
Utilisation des conteneurs	https://documentation.wazuh.com/current/deployment-options/docker/container-usage.html	Consultée pour la version corrigée du lab.
Collecte de fichiers journaux	https://documentation.wazuh.com/current/user-manual/capabilities/log-data-collection/monitoring-log-files.html	Consultée pour la version corrigée du lab.
Règles personnalisées	https://documentation.wazuh.com/current/user-manual/ruleset/rules/custom.html	Consultée pour la version corrigée du lab.
Syntaxe des règles	https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/rules.html	Consultée pour la version corrigée du lab.