



PROJET DE FIN D'ANNÉE

2024 - 2025

Architecture du laboratoire de cybersécurité

Mise en Place d'une Infrastructure Sécurisée pour les Tests de Pénétration et les Opérations de Cyberdéfense

Réalisé par:

Hafedh GUENICHI
Ghofrane LAABIDI
Chokri KHEMIRA
Ibtihel SALHI

Encadré par:

Prof. Sahar BEN YALAA
Département de CYBERSECURITY

Nous dédions ce travail à nos familles et à tous ceux qui nous ont soutenus tout au long de notre parcours académique.

Remerciements

Nous tenons à exprimer notre profonde gratitude envers notre encadrant, Prof. Sahar BEN YALAA, pour son soutien constant, ses conseils précieux et sa disponibilité tout au long de ce projet.

Nous remercions également l'ensemble du corps professoral du département de CYBERSECURITY pour la qualité de l'enseignement dispensé et pour nous avoir fourni les connaissances nécessaires à la réalisation de ce projet.

Nos remerciements s'adressent aussi à l'administration de TEK-UP pour avoir mis à notre disposition les ressources nécessaires pour mener à bien ce travail.

Enfin, nous exprimons notre reconnaissance à nos familles et amis pour leur soutien moral et leurs encouragements durant toute la période de réalisation de ce projet.

Abstract

Ce rapport présente une architecture complète de laboratoire de cybersécurité conçue pour les tests de sécurité avancés, la surveillance et la réponse aux incidents. L'environnement se compose de deux réseaux séparés : un laboratoire de tests d'intrusion avec des systèmes délibérément vulnérables et un centre d'opérations de sécurité (SOC) doté de capacités SIEM et SOAR. L'architecture permet aux professionnels de la sécurité de mener des simulations d'attaques réalistes tout en surveillant et en répondant aux événements de sécurité dans un environnement isolé. Ce document détaille la conception du réseau, les configurations des systèmes et l'intégration entre les composants pour créer une plateforme de formation et de test en cybersécurité pleinement fonctionnelle.

Contents

Remerciements	2
1 Introduction	5
1.1 Présentation du Projet	5
1.2 Objectifs	5
1.3 Composants Clés	6
2 Architecture Réseau	7
2.1 Topologie Réseau	7
2.1.1 Réseau de Test d’Intrusion (192.168.29.0/24)	7
2.1.2 Réseau SOC (192.168.83.0/24)	7
2.2 Flux de Trafic	7
2.3 Configuration du Pare-feu	8
3 Environnement de Test d’Intrusion	10
3.1 Garuda Linux - L’Arsenal du Dragon	10
3.1.1 Caractéristiques Principales	10
3.1.2 Outils de Sécurité Préinstallés	11
3.2 VMs Ubuntu Vulnérables	11
3.2.1 VM 1 : Vulnérabilités de Permissions Utilisateur et SUID	11
3.2.2 VM 2 : Vulnérabilités des Services Réseau	12
3.2.3 VM 3 : Vulnérabilités du Noyau et de Sudo	13
3.3 Intégration de la Surveillance de Sécurité	15
4 Centre d’Opérations de Sécurité (SOC)	16
4.1 Plateforme SIEM Wazuh	16
4.1.1 Architecture Microservices	16
4.1.2 Règles de Détection Personnalisées	17
4.2 Plateforme SOAR	19
4.2.1 The Hive	19
4.2.2 Cortex	20
4.2.3 MISP	21
4.2.4 Déploiement Conteneurisé	22
5 Intégration et Flux de Travail	25
5.1 Flux d’Événements de Sécurité	25
5.2 Intégration Wazuh-SOAR	25
5.3 Flux de Travail de Réponse aux Incidents	27

6 Déploiement et Configuration	28
6.1 Configuration de VMware Workstation	28
6.2 Configuration Réseau	29
6.3 Configuration OPNsense	29
6.4 Déploiement de Wazuh	30
6.5 Déploiement SOAR	32
6.6 Configuration des VMs Vulnérables	32
7 Scénarios d'Utilisation	33
7.1 Exercices de Test d'Intrusion	33
7.2 Surveillance et Détection de Sécurité	33
7.3 Simulation de Réponse aux Incidents	34
7.4 Scénarios de Formation	35
8 Conclusion	36
8.1 Résumé	36
8.2 Améliorations Futures	36
8.3 Réflexions Finales	37
A Scripts d'Installation	38
A.1 Règles du Pare-feu OPNsense	39

List of Figures

1.1	Architecture de haut niveau de l'environnement du laboratoire de cybersécurité	6
2.1	Tableau de bord du pare-feu OPNsense montrant les interfaces réseau et les statistiques de trafic	9
3.1	Environnement de bureau Garuda Linux avec des outils de sécurité	10
3.2	Exemple de vulnérabilités de binaires SUID dans la VM 1	12
3.3	Exemple de vulnérabilités de services réseau dans la VM 2	13
3.4	Exemple de vulnérabilités du noyau et de sudo dans la VM 3	14
3.5	Configuration de l'agent Wazuh sur les VMs vulnérables	15
4.1	Tableau de bord Wazuh montrant les événements de sécurité et les alertes	16
4.2	Architecture microservices de Wazuh	17
4.3	Custom rule for VM1	18
4.4	Alertes Wazuh déclenchées par des tentatives d'exploitation	18
4.5	Tableau de bord de la plateforme SOAR montrant les cas d'incidents	19
4.6	Interface de gestion de cas The Hive	20
4.7	Analyseurs Cortex pour les artefacts de sécurité	21
4.8	Tableau de bord de renseignement sur les menaces MISP	22
4.9	Contenu du docker-compose.yaml	23
4.10	Conteneurs Docker exécutant les composants de la plateforme SOAR	24
5.1	Integration du Wazuh x The Hive	26
6.1	VMware Workstation avec les machines virtuelles du laboratoire	28
6.2	Configuration réseau VMware pour l'environnement du laboratoire	29
6.3	Configuration des interfaces réseau OPNsense	30
6.4	Déploiement de Wazuh à l'aide de Docker Compose	31
6.5	Exécution du script de configuration de VM vulnérable	32
7.1	Exercice de test d'intrusion utilisant Garuda Linux	33
7.2	Tableau de bord de surveillance de sécurité montrant les attaques détectées	34
7.3	Cas de réponse aux incidents dans The Hive	35
8.1	Améliorations futures potentielles de l'architecture du laboratoire	37
A.1	Installation du OPNSense	38
A.2	Regles du OPNSense	39

List of Tables

3.1 Outils de sécurité disponibles dans Garuda Linux	11
--	----

1

Introduction

1.1 Présentation du Projet

Ce projet vise à créer un environnement de laboratoire complet en cybersécurité qui simule des scénarios d'attaque et de défense réels. Le laboratoire est conçu pour fournir un environnement contrôlé permettant aux professionnels de la sécurité de pratiquer des techniques de sécurité offensive tout en surveillant et en répondant simultanément à ces activités à l'aide d'outils de sécurité de niveau entreprise.

L'architecture se compose de deux réseaux isolés connectés par un pare-feu sécurisé :

- Un réseau de test d'intrusion (192.168.29.0/24) contenant des outils d'attaque et des systèmes délibérément vulnérables
- Un réseau d'opérations de sécurité (192.168.83.0/24) hébergeant des plateformes de surveillance et de réponse aux incidents

Cette approche segmentée permet des simulations d'attaques réalistes tout en maintenant un environnement sécurisé pour la surveillance et l'analyse.

1.2 Objectifs

Les principaux objectifs de cette architecture de laboratoire de cybersécurité sont :

- Créer un environnement réaliste pour pratiquer des techniques de sécurité offensive
- Mettre en œuvre une capacité complète de surveillance de la sécurité et de réponse aux incidents
- Permettre l'étude des modèles d'attaque, des méthodes de détection et des procédures de réponse
- Fournir une plateforme pour tester les outils et les configurations de sécurité
- Faciliter l'apprentissage pratique des concepts et technologies de cybersécurité

1.3 Composants Clés

L'architecture du laboratoire intègre plusieurs composants clés :

- **Pare-feu OPNsense** : Fournit une segmentation réseau et un contrôle du trafic entre les réseaux de test d'intrusion et de SOC
- **Garuda Linux** : Une plateforme spécialisée de test d'intrusion avec des outils de sécurité préinstallés
- **VMs Ubuntu Vulnérables** : Trois systèmes délibérément vulnérables conçus pour les tests de sécurité
- **Wazuh SIEM** : Un système de gestion des informations et des événements de sécurité pour la surveillance et les alertes
- **Plateforme SOAR** : Capacités d'orchestration, d'automatisation et de réponse de sécurité via The Hive, Cortex et MISP

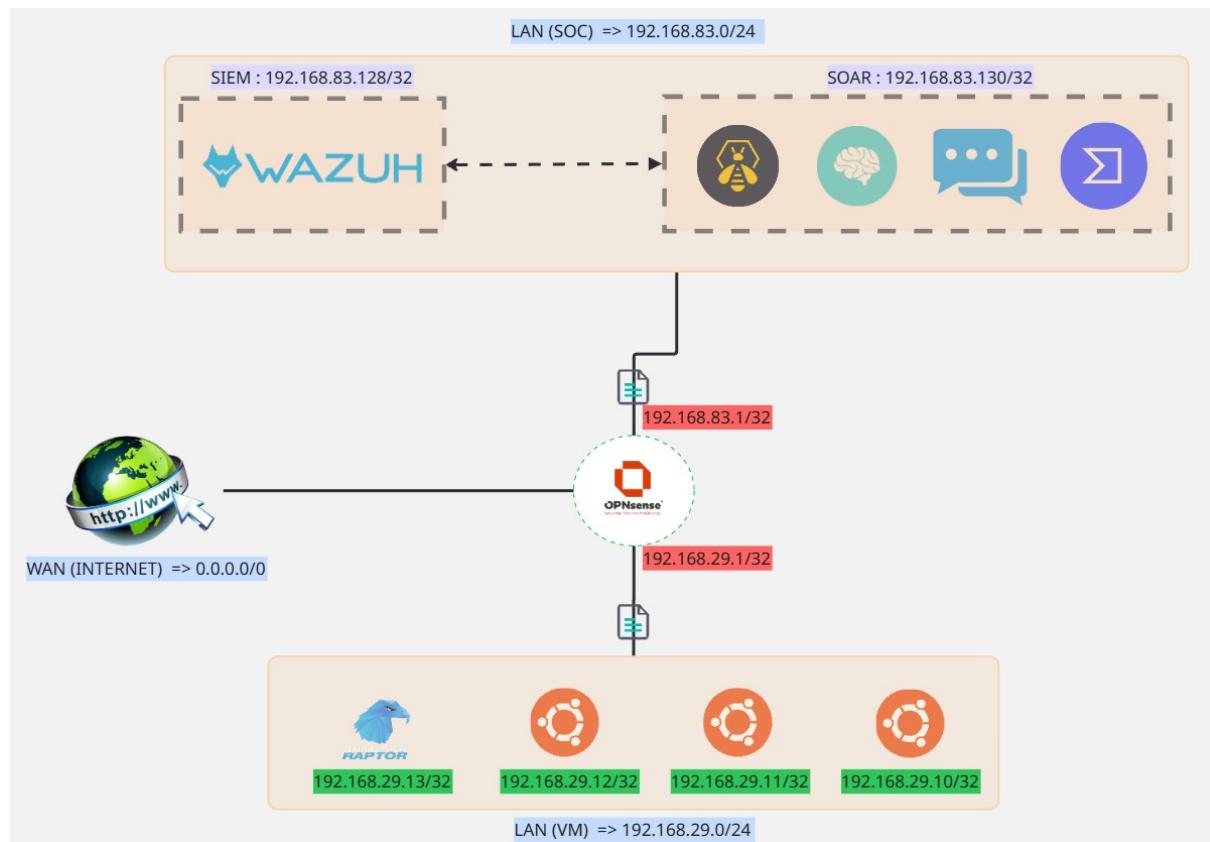


Figure 1.1: Architecture de haut niveau de l'environnement du laboratoire de cybersécurité

2

Architecture Réseau

2.1 Topologie Réseau

L'environnement du laboratoire est construit sur une architecture réseau segmentée avec deux LANs primaires connectés par un pare-feu OPNsense. Cette conception permet un flux de trafic contrôlé entre l'environnement de test d'intrusion et le centre d'opérations de sécurité.

2.1.1 Réseau de Test d'Intrusion (192.168.29.0/24)

Le réseau de test d'intrusion héberge les outils de sécurité offensive et les systèmes cibles vulnérables :

- **Passerelle** : 192.168.29.1 (interface pare-feu OPNsense)
- **Garuda Linux** : 192.168.29.10 (plateforme d'attaque)
- **VM Vulnérable 1** : 192.168.29.101 (vulnérabilités de permissions utilisateur et SUID)
- **VM Vulnérable 2** : 192.168.29.102 (vulnérabilités des services réseau)
- **VM Vulnérable 3** : 192.168.29.103 (vulnérabilités du noyau et sudo)

2.1.2 Réseau SOC (192.168.83.0/24)

Le réseau SOC contient les plateformes de surveillance de sécurité et de réponse aux incidents :

- **Passerelle** : 192.168.83.1 (interface pare-feu OPNsense)
- **Wazuh SIEM** : 192.168.83.128 (plateforme de surveillance de sécurité)
- **Plateforme SOAR** : 192.168.83.130 (automatisation de la réponse aux incidents)

2.2 Flux de Trafic

Le réseau est conçu pour permettre des flux de trafic spécifiques qui permettent la surveillance de la sécurité tout en maintenant une segmentation appropriée :

- Les VMs vulnérables peuvent envoyer des journaux et des événements de sécurité au SIEM Wazuh
- Le SIEM Wazuh peut communiquer avec la plateforme SOAR pour la réponse aux incidents
- Le système Garuda Linux ne peut pas accéder directement au réseau SOC

- Tous les systèmes peuvent accéder à Internet via le pare-feu OPNsense pour les mises à jour et les ressources externes

2.3 Configuration du Pare-feu

Le pare-feu OPNsense sert de point de contrôle central pour le trafic réseau et fournit les fonctions suivantes :

- Segmentation réseau entre les environnements de test d'intrusion et de SOC
- Accès Internet pour tous les systèmes via NAT
- Filtrage du trafic entre les réseaux basé sur les politiques de sécurité
- Interface Web pour la gestion du pare-feu (accessible à 192.168.29.1 et 192.168.83.1)

Règles du Pare-feu OPNsense

Les règles de pare-feu suivantes sont implémentées pour contrôler le trafic entre les réseaux :

- Autoriser les VMs vulnérables (192.168.29.101-103) à communiquer avec le SIEM Wazuh (192.168.83.128)
- Bloquer tout autre trafic du réseau de test d'intrusion vers le réseau SOC
- Autoriser tout le trafic du réseau SOC vers le réseau de test d'intrusion pour la surveillance
- Autoriser tous les systèmes à accéder à Internet pour les mises à jour et les ressources externes

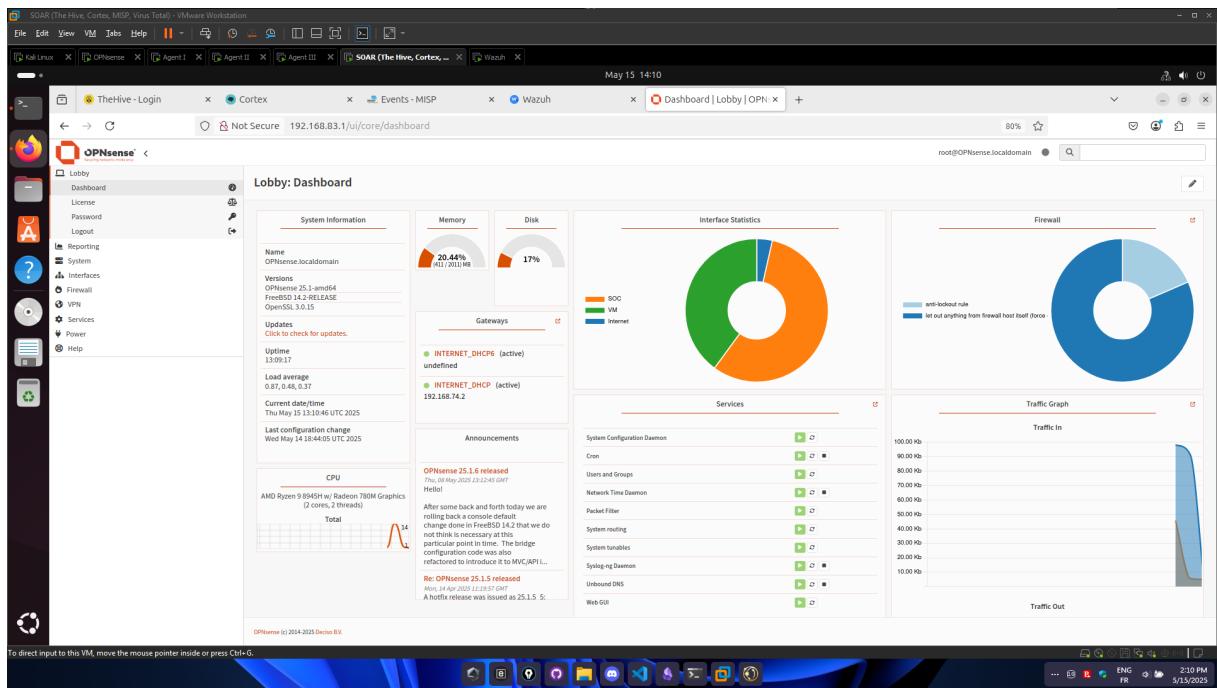


Figure 2.1: Tableau de bord du pare-feu OPNsense montrant les interfaces réseau et les statistiques de trafic

3

Environnement de Test d’Intrusion

3.1 Garuda Linux - L’Arsenal du Dragon

Garuda Linux sert de plateforme d’attaque principale dans l’environnement de test d’intrusion. Cette distribution basée sur Arch est spécifiquement conçue pour les tests d’intrusion et comprend une suite complète d’outils de sécurité.

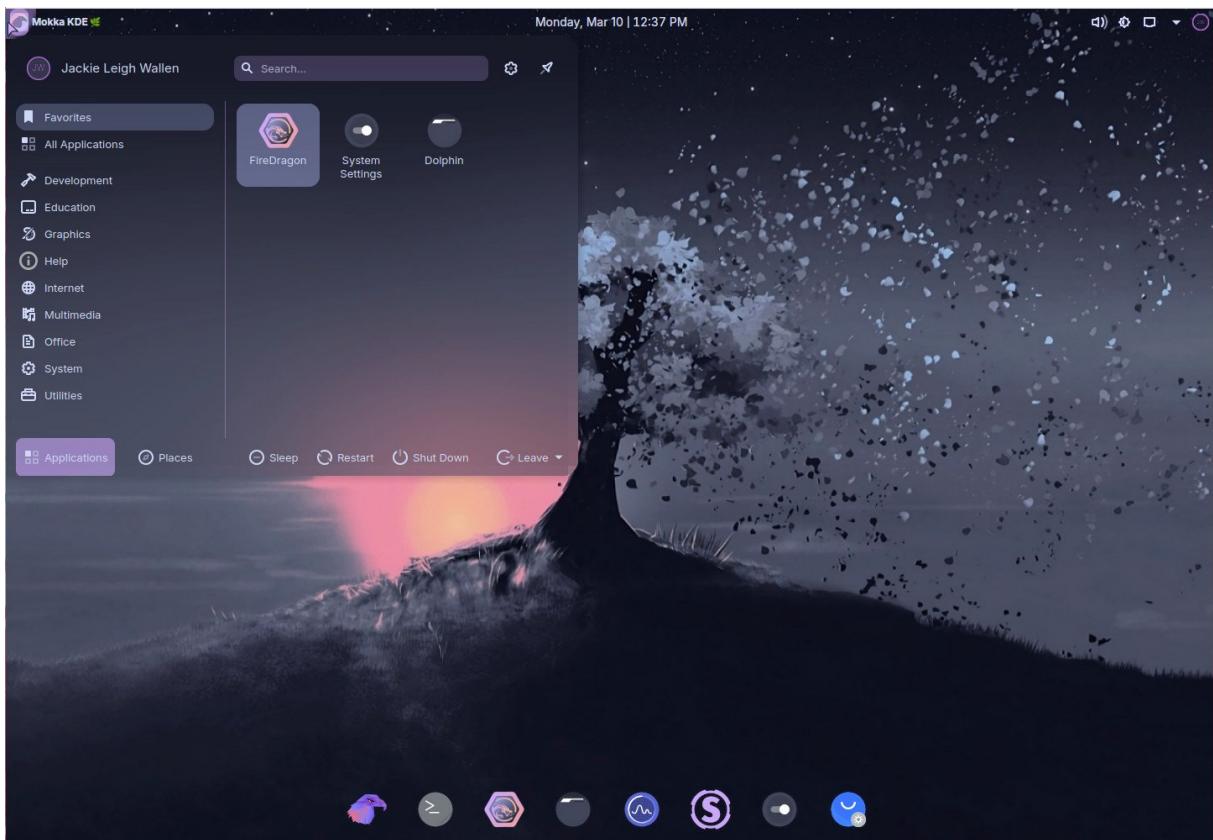


Figure 3.1: Environnement de bureau Garuda Linux avec des outils de sécurité

3.1.1 Caractéristiques Principales

Garuda Linux comprend plusieurs fonctionnalités qui le rendent idéal pour les tests d’intrusion :

- Modèle de publication continue avec les derniers outils de sécurité

- Conception axée sur les performances avec un noyau optimisé
- Suite complète d'outils de test d'intrusion préinstallés
- Interface conviviale avec environnement de bureau personnalisé
- Gestion robuste des paquets via les dépôts Arch et AUR

3.1.2 Outils de Sécurité Préinstallés

Le système Garuda Linux est livré avec une large gamme d'outils de sécurité, notamment :

Catégorie d'Outil	Outils Disponibles
Reconnaissance	Nmap, Masscan, Recon-ng, theHarvester
Analyse de Vulnérabilités	OpenVAS, Nikto, WPScan, SQLmap
Exploitation	Metasploit Framework, Exploit-DB, BeEF
Attaques de Mot de Passe	Hydra, John the Ripper, Hashcat
Tests Sans Fil	Aircrack-ng, Kismet, Wifite
Tests d'Applications Web	Burp Suite, OWASP ZAP, Dirb

Table 3.1: Outils de sécurité disponibles dans Garuda Linux

3.2 VMs Ubuntu Vulnérables

L'environnement de test d'intrusion comprend trois machines virtuelles Ubuntu délibérément vulnérables. Chaque VM est configurée avec des faiblesses de sécurité spécifiques pour fournir des cibles réalistes pour les exercices de test d'intrusion.

3.2.1 VM 1 : Vulnérabilités de Permissions Utilisateur et SUID

Cette VM se concentre sur les vulnérabilités d'escalade de privilèges locaux liées aux permissions utilisateur et aux binaires SUID.

Vulnérabilités de la VM 1

Les principales vulnérabilités de la VM 1 comprennent :

- Identifiants utilisateur faibles (operator:password123)
- Privilèges sudo excessifs pour des utilisateurs spécifiques
- Binaires SUID personnalisés avec des vulnérabilités d'injection de commandes
- Scripts accessibles en écriture par tous exécutés par root via des tâches cron
- Fichiers sensibles avec des permissions faibles

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main(int argc, char *argv[]) {
6     printf("Exécution de l'outil de diagnostic système en tant que : ");
7     system("id");
8
9     if (argc > 1) {
10         printf("Vérification du fichier : %s\n", argv[1]);
11         char command[256];
12         // Vulnérabilité : pas de validation de l'entrée utilisateur
13         sprintf(command, sizeof(command), "cat %s", argv[1]);
14         system(command);
15     } else {
16         printf("Usage : %s <nom_fichier>\n", argv[0]);
17     }
18
19     return 0;
20 }
```

Figure 3.2: Exemple de vulnérabilités de binaires SUID dans la VM 1

3.2.2 VM 2 : Vulnérabilités des Services Réseau

Cette VM se concentre sur les vulnérabilités des services réseau et les configurations non sécurisées.

Vulnérabilités de la VM 2

Les principales vulnérabilités de la VM 2 comprennent :

- Configuration SSH non sécurisée permettant la connexion root
- Services réseau hérités (Telnet, RSH, FTP)
- Exportations NFS vulnérables avec no_root_squash
- Vulnérabilités d'authentification .rhosts
- Règles de pare-feu faibles permettant tout le trafic

```
1 # Configuration Apache avec CGI activé
2 <Directory /var/www/cgi-bin>
3     Options +ExecCGI
4     AddHandler cgi-script .cgi .pl
5     Require all granted
6 </Directory>
7
8 # Script CGI vulnérable (/var/www/cgi-bin/status.cgi)
9 #!/bin/bash
10 echo "Content-type: text/html"
11 echo ""
12 echo "<html><head><title>System Status</title></head><body>"
13 echo "<h1>System Status</h1>"
14 echo "<pre>"
15 if [ "$QUERY_STRING" != "" ]; then
16     # Vulnérabilité d'injection de commande
17     echo "Checking status for: $QUERY_STRING"
18     eval "ping -c 1 $QUERY_STRING"
19 else
20     echo "No host specified"
21 fi
22 echo "</pre></body></html>"
23
24 -----
25
26 # Exploitation de l'injection de commande
27 curl "http://192.168.29.102/cgi-bin/status.cgi?127.0.0.1;id"
28 curl "http://192.168.29.102/cgi-bin/status.cgi?
127.0.0.1%3Bcat%20/etc/passwd"
```

Figure 3.3: Exemple de vulnérabilités de services réseau dans la VM 2

3.2.3 VM 3 : Vulnérabilités du Noyau et de Sudo

Cette VM se concentre sur les faiblesses de configuration du noyau et les vulnérabilités des règles sudo.

Vulnérabilités de la VM 3

Les principales vulnérabilités de la VM 3 comprennent :

- Fonctionnalités de sécurité du noyau désactivées (ASLR, etc.)
- Privilèges sudo excessifs pour des utilisateurs spécifiques
- Binaires SUID vulnérables avec injection de commandes
- Capacités dangereuses sur des binaires standard
- Services systemd vulnérables avec des scripts accessibles en écriture

```

1 # Désactiver l'ASLR
2 echo 0 > /proc/sys/kernel/randomize_va_space
3
4 # Rendre la modification permanente
5 echo "kernel.randomize_va_space = 0" >> /etc/sysctl.conf
6 sysctl -p
7
8 -----
9
10 // vulnerable.c - Programme vulnérable à l'injection de
11 // shellcode
12 #include <stdio.h>
13 #include <string.h>
14
15 void vulnerable_function(char *input) {
16     char buffer[64];
17     strcpy(buffer, input); // Pas de vérification de la
18     // taille
19 }
20
21 int main(int argc, char *argv[]) {
22     if (argc > 1) {
23         vulnerable_function(argv[1]);
24     }
25     return 0;
26 }
```

Figure 3.4: Exemple de vulnérabilités du noyau et de sudo dans la VM 3

3.3 Intégration de la Surveillance de Sécurité

Les trois VMs vulnérables sont configurées pour envoyer des journaux de sécurité et des événements à la plateforme SIEM Wazuh dans le réseau SOC. Cette intégration permet une surveillance en temps réel des événements de sécurité pendant les exercices de test d'intrusion.

```
1 <ossec_config>
2   <client>
3     <server>
4       <address>192.168.83.128</address>
5       <port>1514</port>
6       <protocol>tcp</protocol>
7     </server>
8     <config-profile>ubuntu, ubuntu20, ubuntu22</config-profile>
9     <notify_time>10</notify_time>
10    <time-reconnect>60</time-reconnect>
11    <auto_restart>yes</auto_restart>
12  </client>
13
14 <syscheck>
15   <disabled>no</disabled>
16   <frequency>43200</frequency>
17   <scan_on_start>yes</scan_on_start>
18   <alert_new_files>yes</alert_new_files>
19   <auto_ignore>no</auto_ignore>
20   <directories check_all="yes" realtime="yes">/etc</directories>
21   <directories check_all="yes" realtime="yes">/usr/bin</directories>
22   <directories check_all="yes" realtime="yes">/usr/sbin</directories>
23   <directories check_all="yes" realtime="yes">/bin</directories>
24   <directories check_all="yes" realtime="yes">/sbin</directories>
25   <directories check_all="yes" realtime="yes">/usr/local/bin</directories>
26   <!-- Configuration spécifique pour VM1 -->
27   <directories check_all="yes" realtime="yes"
      report_changes="yes">/usr/local/bin/sysdiag</directories>
28     <directories check_all="yes" realtime="yes"
      report_changes="yes">/usr/local/bin/system-backup.sh</directories>
29   </syscheck>
30 </ossec_config>
```

Figure 3.5: Configuration de l'agent Wazuh sur les VMs vulnérables

4

Centre d'Opérations de Sécurité (SOC)

4.1 Plateforme SIEM Wazuh

La plateforme SIEM Wazuh sert de système central de surveillance de la sécurité dans le réseau SOC. Elle collecte, analyse et corrèle les événements de sécurité des VMs vulnérables pour détecter et alerter sur les incidents de sécurité.

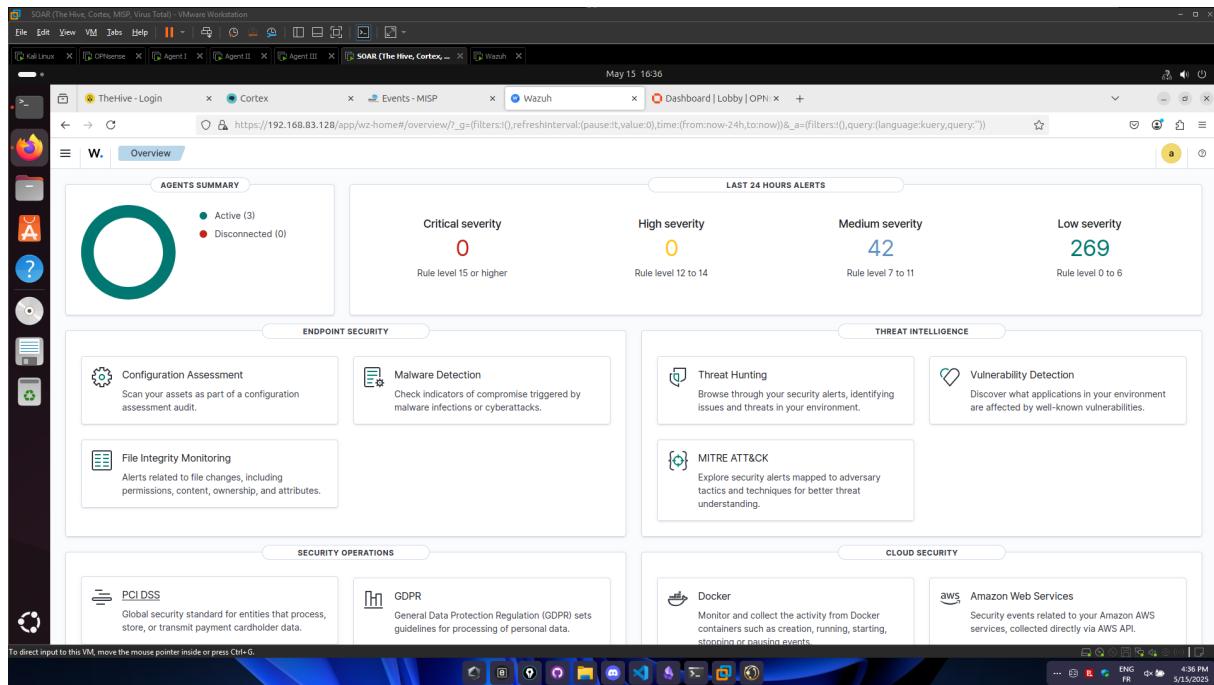


Figure 4.1: Tableau de bord Wazuh montrant les événements de sécurité et les alertes

4.1.1 Architecture Microservices

La plateforme SIEM Wazuh est implémentée comme un ensemble de microservices conteneurisés, notamment :

- **Wazuh Manager** : Composant principal qui reçoit et analyse les événements de sécurité
- **Wazuh Indexer** : Composant basé sur Elasticsearch pour stocker et indexer les données de sécurité
- **Wazuh Dashboard** : Interface Web pour visualiser et interagir avec les données de sécurité

- **Filebeat** : Expéditeur de données léger pour transférer les journaux des agents au gestionnaire

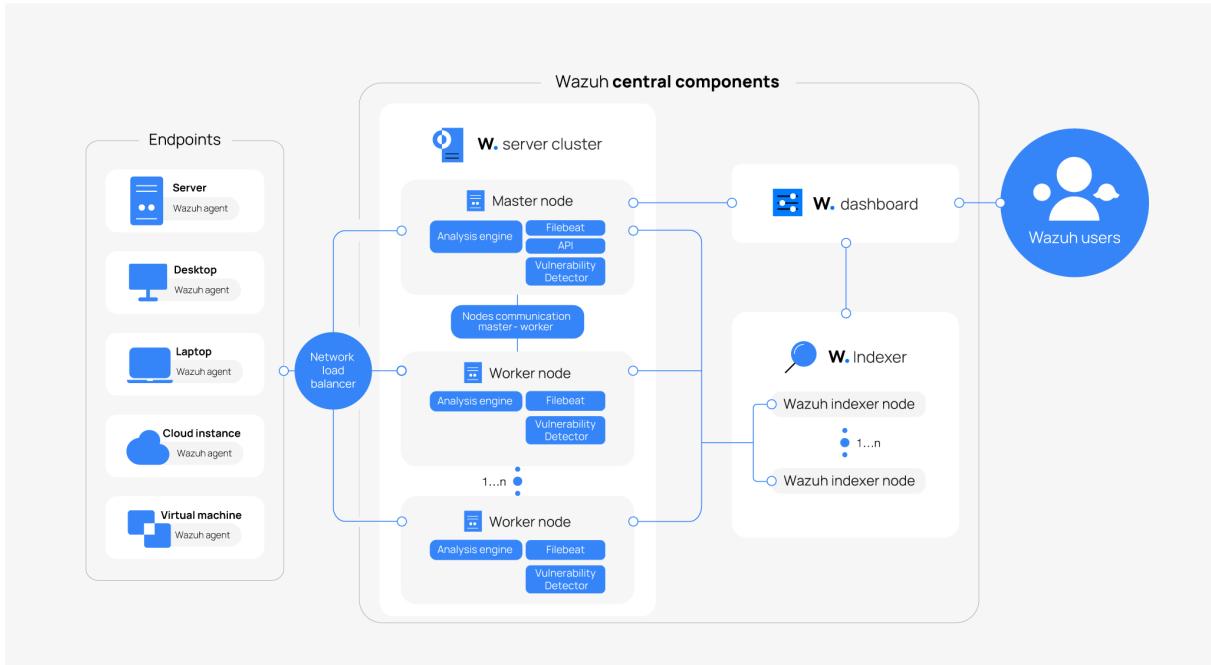


Figure 4.2: Architecture microservices de Wazuh

4.1.2 Règles de Détection Personnalisées

Le SIEM Wazuh est configuré avec des règles de détection personnalisées spécifiquement conçues pour identifier les tentatives d'exploitation contre les VMs vulnérables.

```

1  <!-- Règles de Détection VM1 -->
2  <group name="vm1,vulnerability,">
3    <!-- Surveiller l'exécution de binaires SUID -->
4    <rule id="100102" level="13">
5      <if_group>syscheck</if_group>
6      <match>/usr/local/bin/sysdiag|/usr/local/bin/syshelper</match>
7      <description>VM1: Binaire SUID vulnérable accédé</description>
8    </rule>
9
10   <!-- Surveiller les modifications des scripts de tâches cron -->
11  <rule id="100103" level="14">
12    <if_group>syscheck</if_group>
13    <match>/usr/local/bin/system-backup.sh</match>
14    <description>VM1: Script cron vulnérable modifié</description>
15  </rule>
16 </group>

```

Figure 4.3: Custom rule for VM1

18 hits					
May 17, 2025 @ 19:02:01.430 - May 18, 2025 @ 19:02:01.430					
	timestamp	agent.name	rule.description	rule.level	rule.id
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: brute force trying to get access to the system. Authentication failed.	10	5763	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	sshd: authentication failed.	5	5760	
May 18, 2025 @ 18:59:27.6...	Agent-1	PAM: User login failed.	5	5503	
May 18, 2025 @ 18:51:45.7...	Agent-1	Wazuh agent disconnected.	3	504	
May 18, 2025 @ 18:35:34.7...	Agent-1	Host-based anomaly detection event (rootcheck).	7	510	
May 18, 2025 @ 18:35:34.7...	Agent-1	Host-based anomaly detection event (rootcheck).	7	510	

Figure 4.4: Alertes Wazuh déclenchées par des tentatives d'exploitation

4.2 Plateforme SOAR

La plateforme d'Orchestration, d'Automatisation et de Réponse de Sécurité (SOAR) fournit des capacités de réponse aux incidents grâce à un ensemble intégré d'outils. Cette plateforme automatise les opérations de sécurité et permet une réponse coordonnée aux incidents de sécurité.

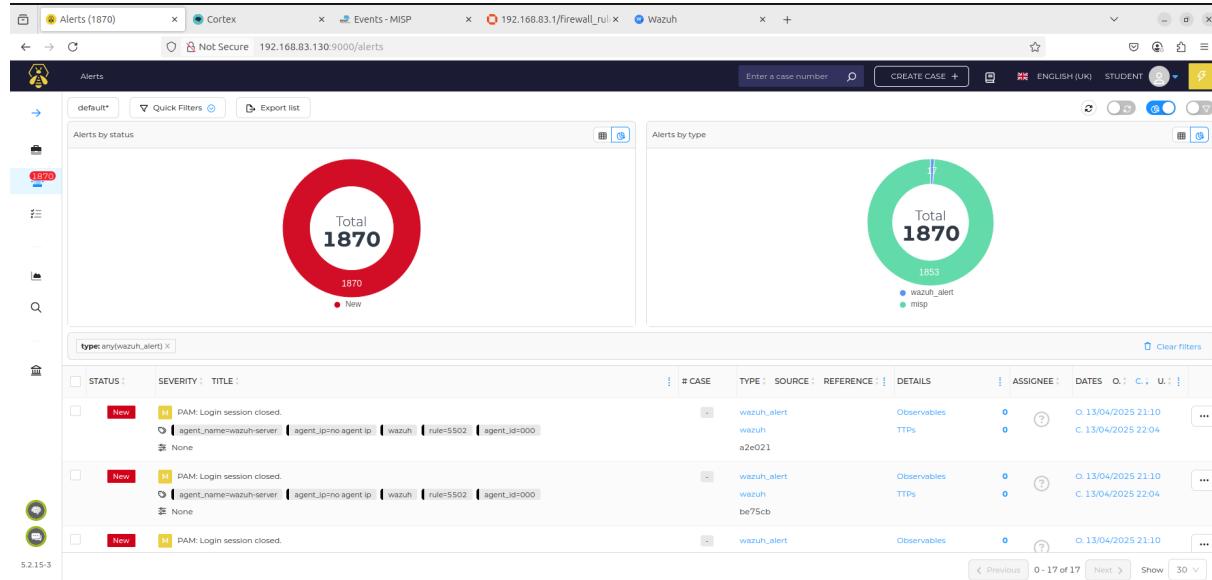


Figure 4.5: Tableau de bord de la plateforme SOAR montrant les cas d'incidents

4.2.1 The Hive

The Hive sert de système central de gestion des cas pour les incidents de sécurité. Il fournit une plateforme pour suivre, documenter et collaborer sur les enquêtes de sécurité.

Fonctionnalités de The Hive

Les principales fonctionnalités de The Hive comprennent :

- Gestion des cas pour les incidents de sécurité
- Attribution et suivi des tâches
- Intégration avec les plateformes de renseignement sur les menaces
- Modèles personnalisables pour différents types d'incidents
- Outils de collaboration pour les équipes de sécurité

CHAPTER 4. CENTRE D'OPÉRATIONS DE SÉCURITÉ (SOC2). PLATEFORME SOAR

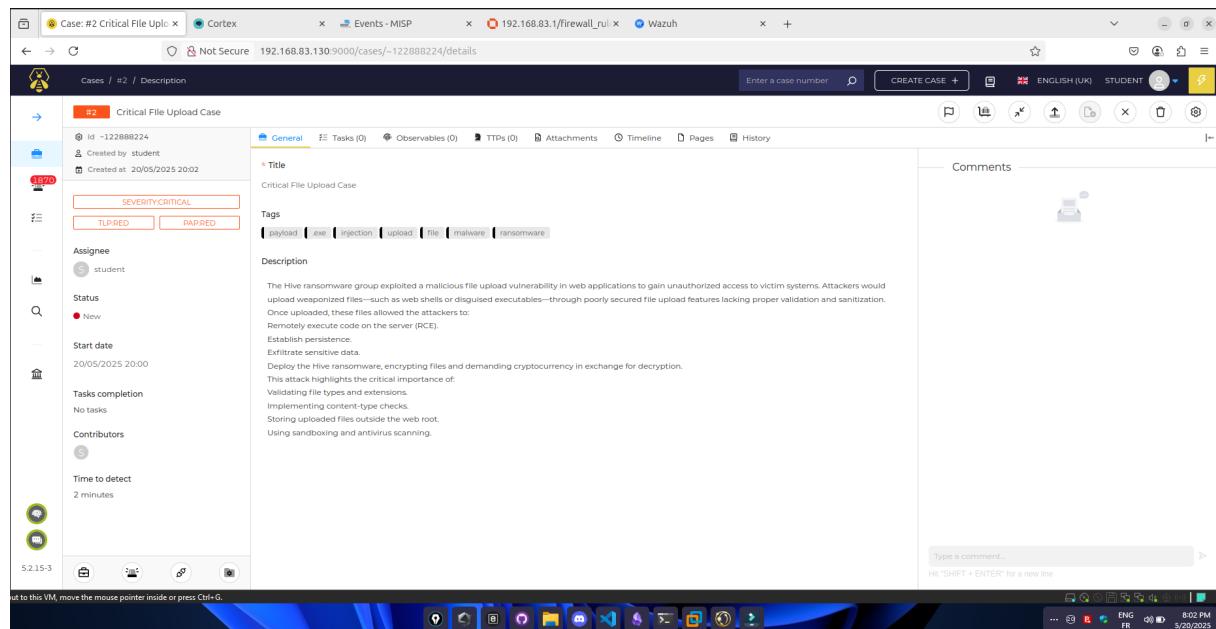


Figure 4.6: Interface de gestion de cas The Hive

4.2.2 Cortex

Cortex fournit des capacités d'analyse automatisée pour les artefacts de sécurité. Il peut analyser des fichiers, des URL, des adresses IP et d'autres indicateurs de compromission pour fournir un contexte supplémentaire aux enquêtes de sécurité.

Analyseurs Cortex

Cortex comprend des analyseurs pour divers types d'artefacts de sécurité :

- Analyseurs de fichiers (VirusTotal, YARA, etc.)
- Analyseurs d'URL (URLScan, PhishTank, etc.)
- Analyseurs d'IP (AbuseIPDB, GreyNoise, etc.)
- Analyseurs de domaine (DNS passif, WHOIS, etc.)
- Analyseurs de hachage (MISP, VirusTotal, etc.)

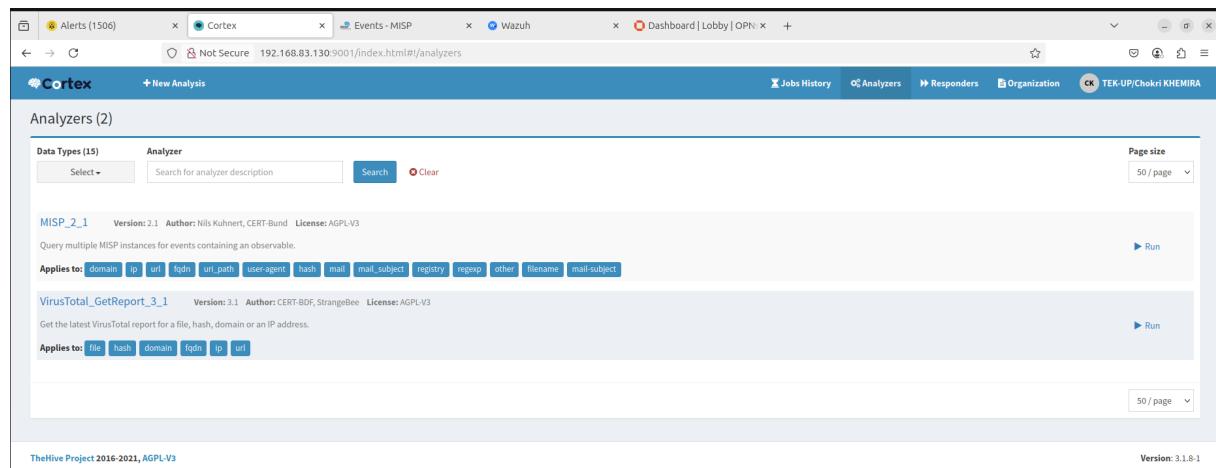


Figure 4.7: Analyseurs Cortex pour les artefacts de sécurité

4.2.3 MISP

La Plateforme de Partage d'Informations sur les Malwares (MISP) fournit des capacités de renseignement sur les menaces. Elle permet la collecte, le stockage et le partage d'indicateurs de compromission et de renseignements sur les menaces.

Fonctionnalités de MISP

Les principales fonctionnalités de MISP comprennent :

- Stockage et corrélation des indicateurs de compromission
- Corrélation automatique des attributs
- Options de partage flexibles pour le renseignement sur les menaces
- Capacités d'exportation dans divers formats
- Intégration avec d'autres outils de sécurité

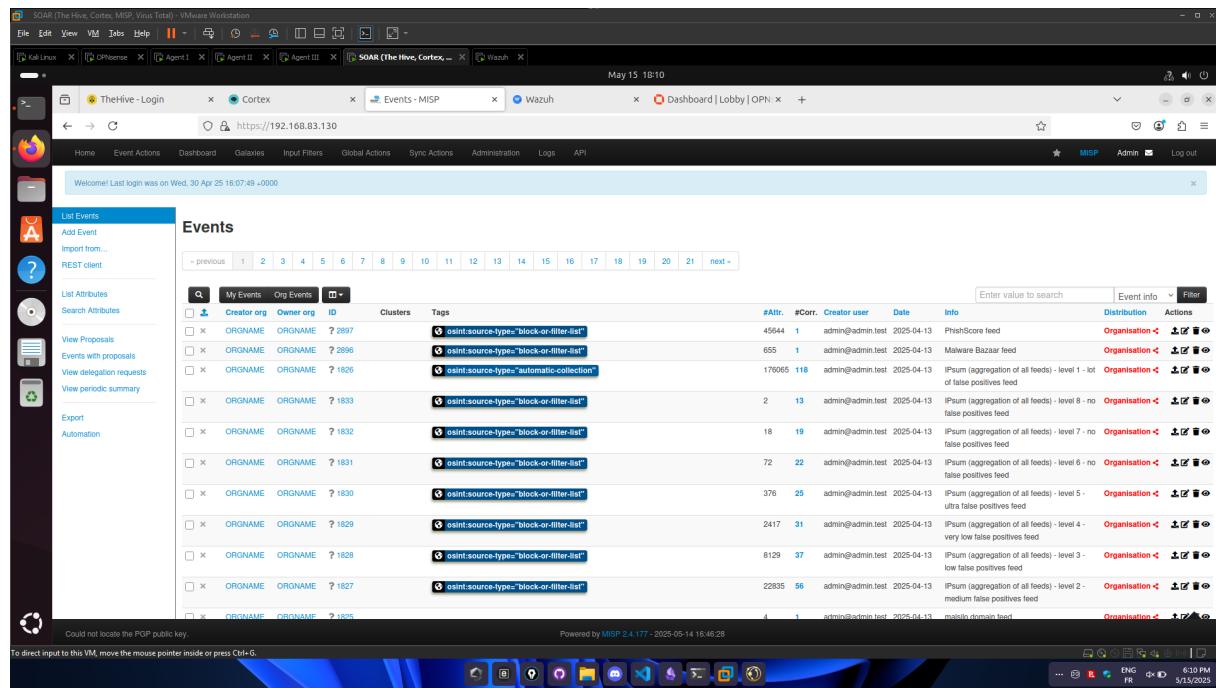


Figure 4.8: Tableau de bord de renseignement sur les menaces MISP

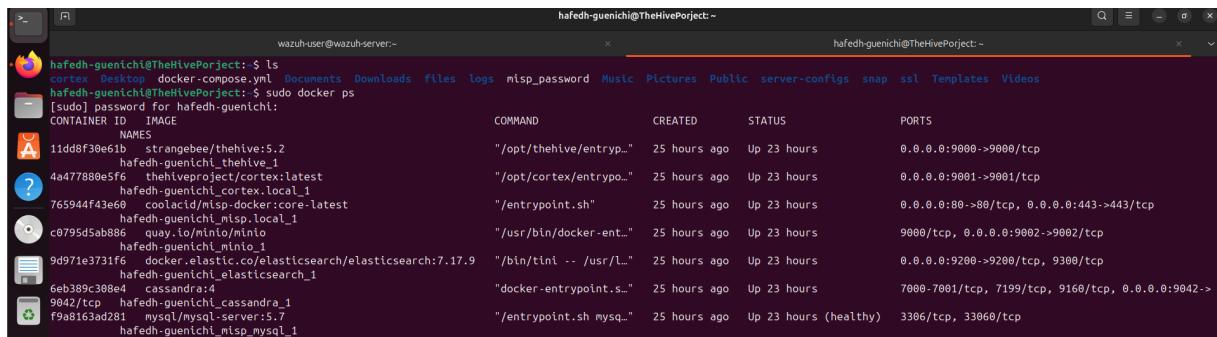
4.2.4 Déploiement Conteneurisé

La plateforme SOAR est déployée à l'aide de conteneurs Docker orchestrés avec Docker Compose. Cette approche offre modularité, évolutivité et facilité de gestion.

```
● ● ●

 1 version: '3'
 2 services:
 3   thehive:
 4     image: thehiveproject/thehive4:latest
 5     depends_on:
 6       - cassandra
 7     ports:
 8       - "9000:9000"
 9     environment:
10       - CQL_HOSTNAMES=cassandra
11     volumes:
12       - thehive_data:/opt/thp/thehive/data
13     restart: unless-stopped
14
15   cortex:
16     image: thehiveproject/cortex:latest
17     depends_on:
18       - elasticsearch
19     ports:
20       - "9001:9001"
21     volumes:
22       - cortex_data:/opt/cortex/data
23     restart: unless-stopped
24
25   misp:
26     image: coolacid/misp-docker:core-latest
27     depends_on:
28       - misp_db
29     ports:
30       - "80:80"
31       - "443:443"
32     volumes:
33       - misp_data:/var/www/MISP
34     restart: unless-stopped
```

Figure 4.9: Contenu du docker-compose.yaml



The screenshot shows a terminal window with two tabs. The left tab is titled "hafedh-guenichi@TheHiveProject:~" and the right tab is also titled "hafedh-guenichi@TheHiveProject:~". The terminal displays the output of the command "sudo docker ps". The output lists several Docker containers with their names, images, commands, creation times, statuses, and ports.

CONTAINER ID	NAMES	IMAGE	COMMAND	CREATED	STATUS	PORTS
11dd8f30ee1b	strangebee/thehive:5.2	hafedh-guenichi/thehive_1	"/opt/thehive/entryp..."	25 hours ago	Up 23 hours	0.0.0.0:9000->9000/tcp
44477880e5f6	thehiveproject/cortex:latest	hafedh-guenichi_cortex_local_1	"/opt/cortex/entryp..."	25 hours ago	Up 23 hours	0.0.0.0:9001->9001/tcp
765944f43e68	coolacid/misp-docker:core-latest	hafedh-guenichi_misp_local_1	"/entrypoint.sh"	25 hours ago	Up 23 hours	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
c0795d5ab886	quay.io/minio/minio	hafedh-guenichi_minio_1	"/usr/bin/docker-ent..."	25 hours ago	Up 23 hours	9008/tcp, 0.0.0.0:9002->9002/tcp
9d971e3731f6	docker.elastic.co/elasticsearch/elasticsearch:7.17.9	hafedh-guenichi_elasticsearch_1	"/bin/tini -- /usr/L..."	25 hours ago	Up 23 hours	0.0.0.0:9200->9200/tcp, 9300/tcp
6eb389c308e4	cassandra:4	hafedh-guenichi_cassandra_1	"docker-entrypoint.s..."	25 hours ago	Up 23 hours	7080-7081/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp
f9a8163ad281	mysql/mysql-server:5.7	hafedh-guenichi_misp_mysql_1	"/entrypoint.sh mysq..."	25 hours ago	Up 23 hours (healthy)	3306/tcp, 33060/tcp

Figure 4.10: Conteneurs Docker exécutant les composants de la plateforme SOAR

5

Intégration et Flux de Travail

5.1 Flux d'Événements de Sécurité

Le laboratoire de cybersécurité intégré permet un flux complet d'événements de sécurité de l'attaque à la détection et à la réponse :

1. Un attaquant utilise Garuda Linux pour exploiter des vulnérabilités dans les VMs cibles
2. Les VMs vulnérables génèrent des événements de sécurité qui sont collectés par les agents Wazuh
3. Les agents Wazuh transmettent les événements au Wazuh Manager dans le réseau SOC
4. Le Wazuh Manager analyse les événements et génère des alertes basées sur les règles de détection
5. Les alertes sont affichées dans le tableau de bord Wazuh et peuvent être transmises à The Hive
6. The Hive crée des cas pour les incidents de sécurité qui nécessitent une enquête
7. Les analystes utilisent Cortex pour analyser les artefacts liés à l'incident de sécurité
8. MISP fournit des renseignements sur les menaces pour enrichir l'enquête
9. Les actions de réponse peuvent être automatisées ou exécutées manuellement en fonction des résultats

5.2 Intégration Wazuh-SOAR

L'intégration entre le SIEM Wazuh et la plateforme SOAR permet une réponse automatisée aux incidents :

- Les alertes Wazuh sont transmises à The Hive à l'aide d'un script d'intégration personnalisé
- The Hive crée des cas basés sur la gravité et le type d'alerte
- Les analyseurs Cortex sont automatiquement déclenchés pour les artefacts pertinents
- Les actions de réponse peuvent être automatisées en fonction de playbooks prédéfinis

```

 1  #!/usr/bin/env python3
 2  import json
 3  import requests
 4  import sys
 5  import os
 6  from datetime import datetime
 7
 8  # Configuration de l'API The Hive
 9  thehive_url = 'http://192.168.83.130:9000'
10 thehive_api_key = os.environ.get('THEHIVE_API_KEY')
11 thehive_headers = {
12     'Authorization': f'Bearer {thehive_api_key}',
13     'Content-Type': 'application/json'
14 }
15
16 # Traiter l'alerte Wazuh
17 def process_alert(alert_json):
18     alert = json.loads(alert_json)
19
20     # Créer un cas The Hive
21     case = {
22         "title": f'Alerte Wazuh: {alert["rule"]['description']}',
23         "description": json.dumps(alert, indent=2),
24         "severity": map_severity(alert['rule']['level']),
25         "tags": ["wazuh", alert['rule']['groups']],
26         "tlp": 2,
27         "pap": 2,
28         "customFields": {
29             "sourceIp": {"string": alert.get('srcip', 'N/A')},
30             "destIp": {"string": alert.get('dstip', 'N/A')},
31             "ruleId": {"string": str(alert['rule']['id'])},
32             "agentName": {"string": alert['agent']['name']}
33         }
34     }
35
36     # Envoyer à The Hive
37     response = requests.post(
38         f'{thehive_url}/api/case',
39         headers=thehive_headers,
40         data=json.dumps(case)
41     )
42
43     if response.status_code == 201:
44         print(f"Cas créé avec succès: {response.json()['id']}")
45     else:
46         print(f"Erreur lors de la création du cas: {response.text}")
47
48 # Mapper la gravité Wazuh à la gravité The Hive
49 def map_severity(level):
50     if level >= 12:
51         return 3 # Elevé
52     elif level >= 8:
53         return 2 # Moyen
54     else:
55         return 1 # Faible
56
57 if __name__ == '__main__':
58     # Lire l'alerte depuis stdin
59     alert_json = sys.stdin.read()
60     process_alert(alert_json)

```

Figure 5.1: Integration du Wazuh x The Hive

5.3 Flux de Travail de Réponse aux Incidents

L'environnement intégré prend en charge un flux de travail complet de réponse aux incidents :

1. **Détection** : Wazuh détecte les événements de sécurité basés sur des règles personnalisées
2. **Triage** : Les alertes sont priorisées en fonction de la gravité et de l'impact
3. **Investigation** : Les analystes enquêtent sur les incidents à l'aide de The Hive et Cortex
4. **Confinement** : Des actions de réponse sont exécutées pour contenir la menace
5. **Éradication** : La cause première de l'incident est traitée
6. **Récupération** : Les systèmes sont restaurés à un fonctionnement normal
7. **Leçons Apprises** : Les incidents sont documentés pour référence future

6

Déploiement et Configuration

6.1 Configuration de VMware Workstation

L'ensemble du laboratoire de cybersécurité est déployé sur VMware Workstation 17.6 Pro. Cette plateforme de virtualisation fournit l'isolation et la flexibilité nécessaires pour l'environnement du laboratoire.

Configuration de VMware Workstation

Paramètres clés de VMware Workstation :

- VMware Workstation 17.6 Pro
- Réseaux host-only pour les réseaux de laboratoire isolés
- Réseau NAT pour l'accès Internet
- Capacités de snapshot pour la récupération du système
- Allocation de ressources basée sur les exigences du système

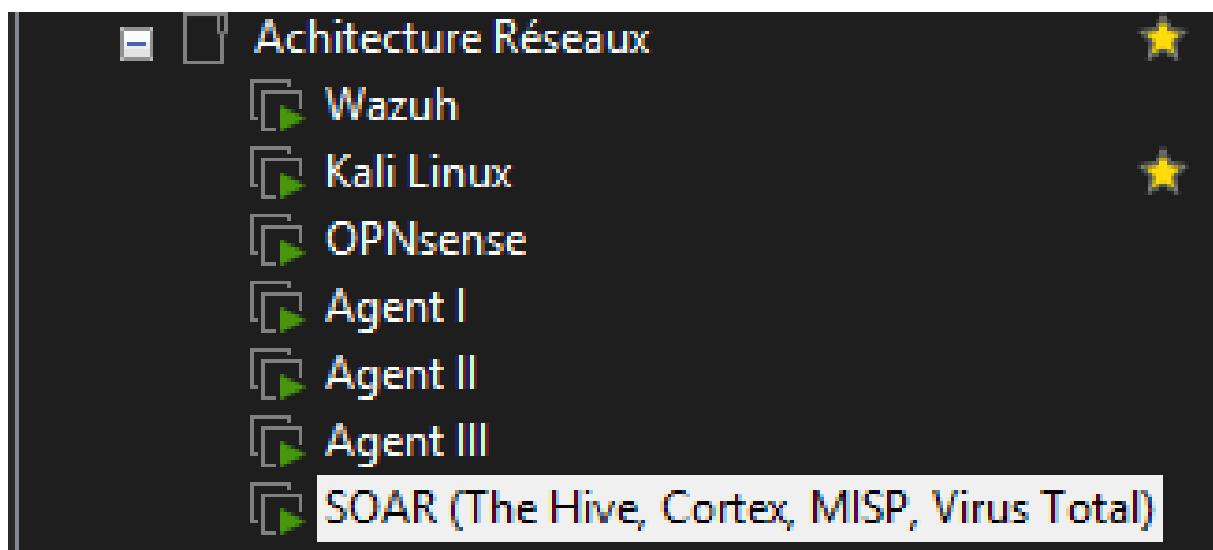
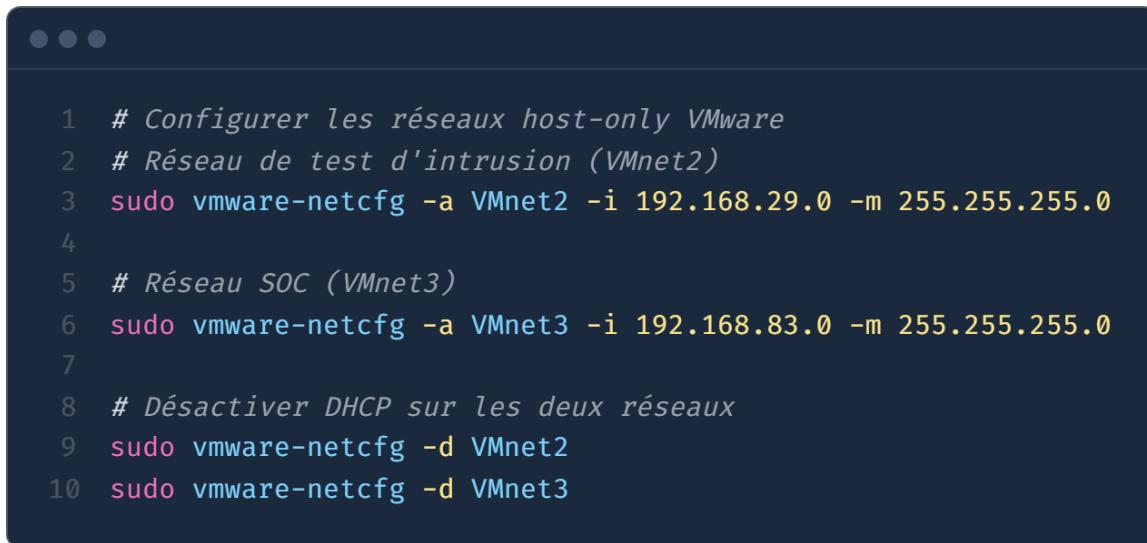


Figure 6.1: VMware Workstation avec les machines virtuelles du laboratoire

6.2 Configuration Réseau

L'environnement du laboratoire nécessite des configurations réseau spécifiques dans VMware Workstation :

- Deux réseaux host-only pour les environnements de test d'intrusion et de SOC
- Réseau NAT pour l'accès Internet
- Attributions d'IP statiques pour tous les systèmes



```

1 # Configurer les réseaux host-only VMware
2 # Réseau de test d'intrusion (VMnet2)
3 sudo vmware-netcfg -a VMnet2 -i 192.168.29.0 -m 255.255.255.0
4
5 # Réseau SOC (VMnet3)
6 sudo vmware-netcfg -a VMnet3 -i 192.168.83.0 -m 255.255.255.0
7
8 # Désactiver DHCP sur les deux réseaux
9 sudo vmware-netcfg -d VMnet2
10 sudo vmware-netcfg -d VMnet3

```

Figure 6.2: Configuration réseau VMware pour l'environnement du laboratoire

6.3 Configuration OPNsense

Le pare-feu OPNsense nécessite une configuration spécifique pour segmenter correctement les réseaux et contrôler le flux de trafic :

- Trois interfaces réseau : WAN, LAN1 (test d'intrusion) et LAN2 (SOC)
- Règles de pare-feu pour contrôler le trafic entre les réseaux
- Configuration NAT pour l'accès Internet

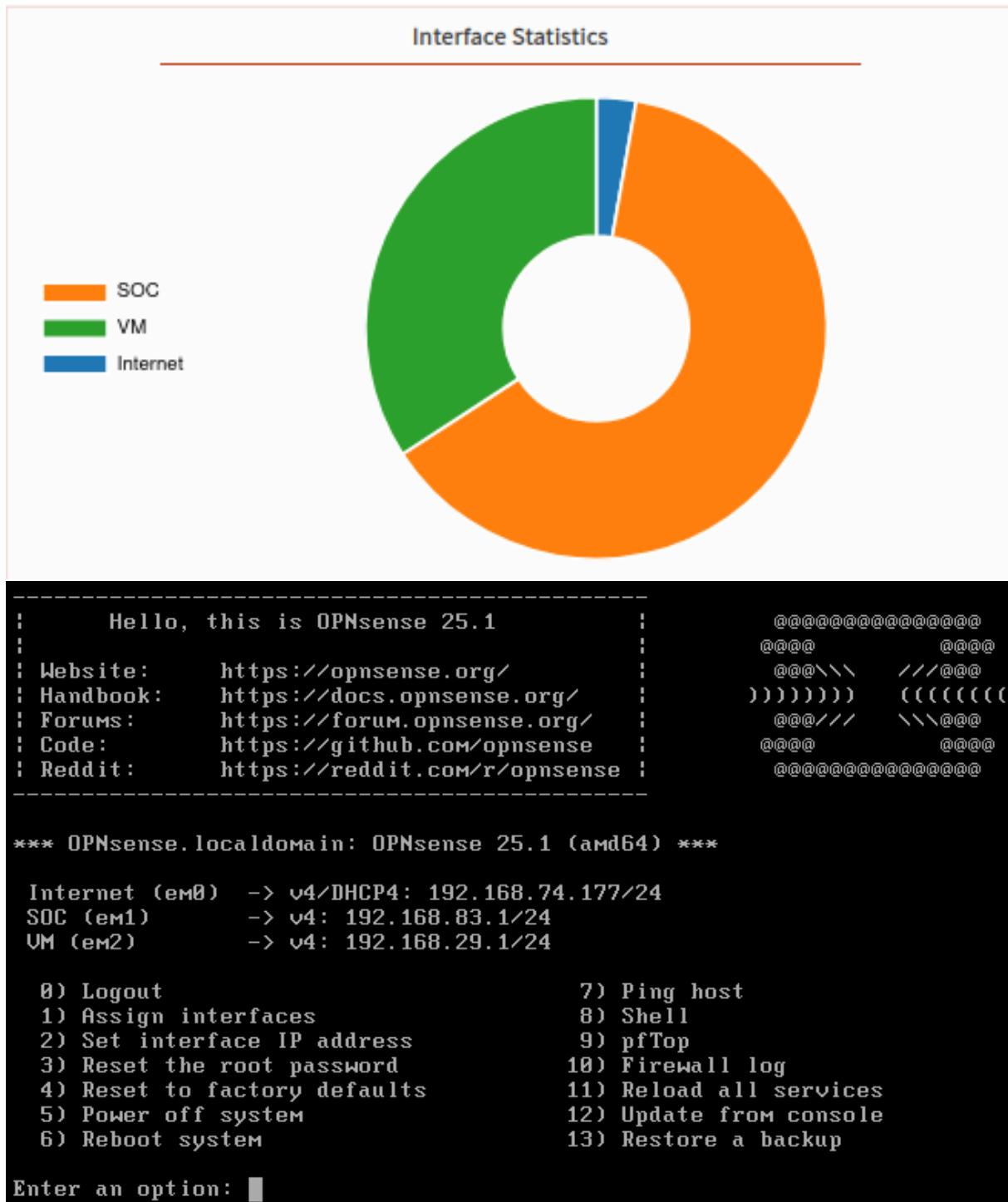


Figure 6.3: Configuration des interfaces réseau OPNsense

6.4 Déploiement de Wazuh

La plateforme SIEM Wazuh est déployée comme un ensemble de microservices conteneurisés :

```
1 # Installer Docker et Docker Compose
2 apt-get update
3 apt-get install -y docker.io docker-compose
4
5 # Cloner le dépôt Docker Wazuh
6 git clone https://github.com/wazuh/wazuh-docker.git
7 cd wazuh-docker/single-node
8
9 # Déployer la pile Wazuh
10 docker-compose up -d
```

Figure 6.4: Déploiement de Wazuh à l'aide de Docker Compose

6.5 Déploiement SOAR

La plateforme SOAR est déployée à l'aide d'une configuration Docker Compose personnalisée.

6.6 Configuration des VMs Vulnérables

Les VMs Ubuntu vulnérables sont configurées avec des faiblesses de sécurité spécifiques :

```

1 #!/bin/bash
2
3 # VM1 : Vulnérabilités de Permissions Utilisateur et SUID
4 # Créer des utilisateurs vulnérables
5 useradd -m -s /bin/bash operator
6 echo "operator:password123" | chpasswd
7
8 # Ajouter operator à sudoers avec NOPASSWD pour des commandes spécifiques
9 echo "operator ALL=(ALL) NOPASSWD: /usr/bin/find, /usr/bin/python3, /bin/cp, /bin/cat" >
    /etc/sudoers.d/operator
10 chmod 440 /etc/sudoers.d/operator
11
12 # Créer un binaire SUID vulnérable
13 cat > /tmp/vulnsuid.c << 'EOF'
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <unistd.h>
17
18 int main(int argc, char *argv[]) {
19     printf("Exécution de l'outil de diagnostic système en tant que : ");
20     system("id");
21
22     if (argc > 1) {
23         printf("Vérification du fichier : %s\n", argv[1]);
24         char command[256];
25         snprintf(command, sizeof(command), "cat %s", argv[1]);
26         system(command);
27     } else {
28         printf("Usage : %s <nom_fichier>\n", argv[0]);
29     }
30
31     return 0;
32 }
33 EOF
34
35 gcc /tmp/vulnsuid.c -o /usr/local/bin/sysdiag
36 chmod u+s /usr/local/bin/sysdiag
37
38 # Vulnérabilités supplémentaires...

```

Figure 6.5: Exécution du script de configuration de VM vulnérable

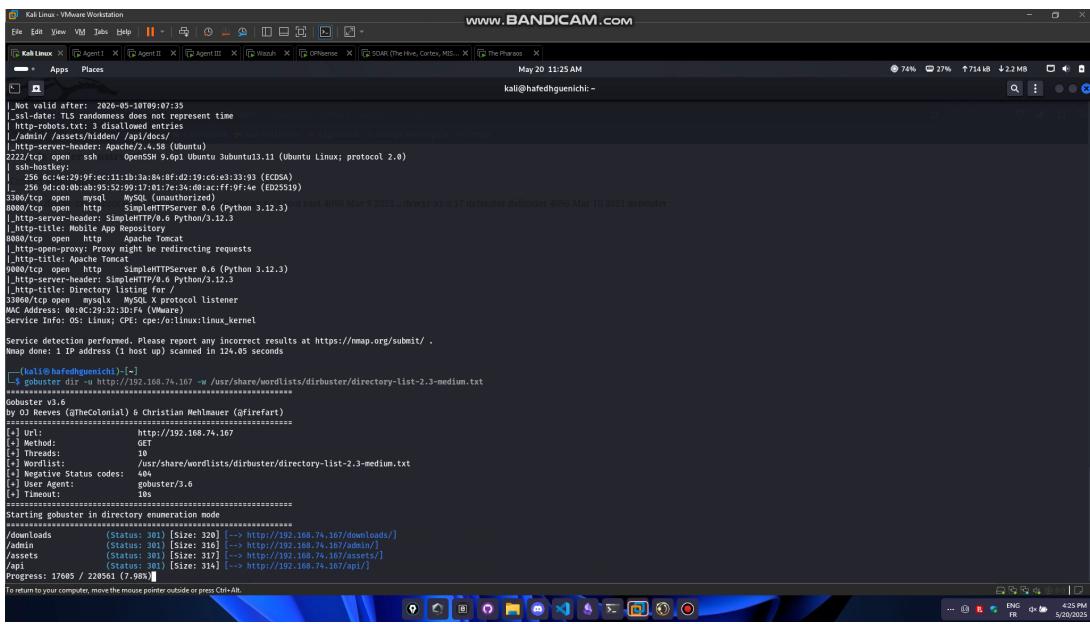
7

Scénarios d'Utilisation

7.1 Exercices de Test d'Intrusion

L'environnement du laboratoire permet divers exercices de test d'intrusion :

- Reconnaissance des VMs vulnérables
- Analyse de vulnérabilités et exploitation
- Escalade de priviléges par diverses techniques
- Activités post-exploitation
- Mouvement latéral entre les systèmes



The screenshot shows a Kali Linux terminal window with several tabs open. The current tab displays a network scan output from Nmap. The output includes information about hosts at 192.168.74.107, such as port 2222/tcp listening on OpenSSH 9.0.1p1 Ubuntu 1ubuntu13.11 (Ubuntu Linux; protocol 2.0). It also lists other ports like 8000/tcp (SimpleHTTPServer 0.6 Python/3.12.3) and 8080/tcp (Apache Tomcat). Below the Nmap output, a gobuster command is running to find directory paths on the target host. The command is: gobuster dir -u http://192.168.74.107 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt. The gobuster output shows various directory paths found on the target server.

Figure 7.1: Exercice de test d'intrusion utilisant Garuda Linux

7.2 Surveillance et Détection de Sécurité

L'environnement SOC permet des exercices de surveillance et de détection de sécurité :

- Surveillance en temps réel des événements de sécurité

- Triage et investigation des alertes
- Corrélation d’événements à travers plusieurs systèmes
- Chasse aux menaces à l’aide des données de sécurité collectées
- Développement et test de règles de détection personnalisées

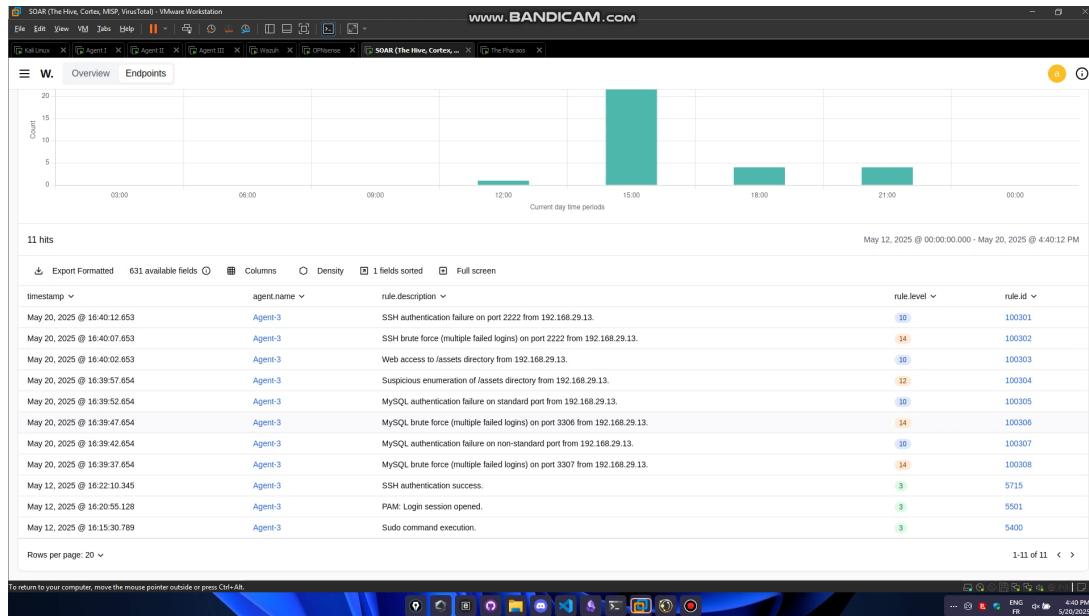


Figure 7.2: Tableau de bord de surveillance de sécurité montrant les attaques détectées

7.3 Simulation de Réponse aux Incidents

L’environnement intégré permet des simulations complètes de réponse aux incidents :

- Détection des incidents de sécurité
- Gestion et documentation des cas
- Analyse et enrichissement des artefacts
- Exécution d’actions de réponse
- Analyse et rapport post-incident

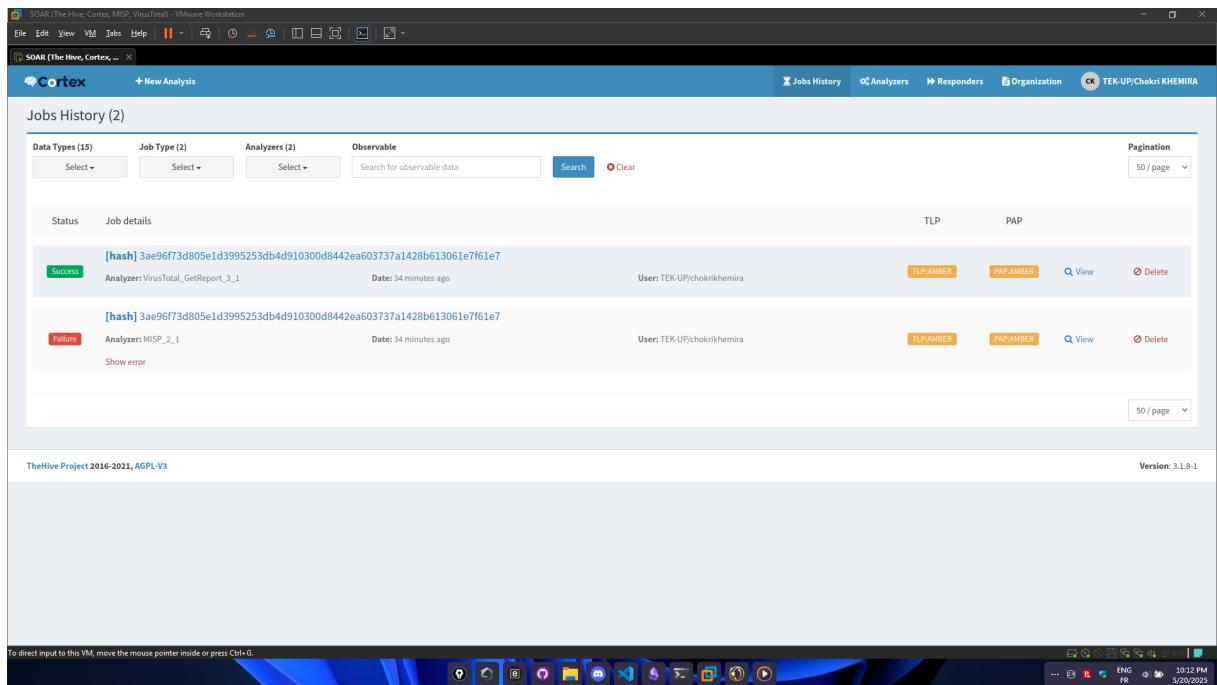


Figure 7.3: Cas de réponse aux incidents dans The Hive

7.4 Scénarios de Formation

L'environnement du laboratoire peut être utilisé pour divers scénarios de formation :

- Formation en sécurité offensive
- Formation en sécurité défensive
- Exercices d'équipe bleue
- Exercices d'équipe rouge
- Exercices d'équipe pourpre

8

Conclusion

8.1 Résumé

Cette architecture de laboratoire de cybersécurité fournit un environnement complet pour les tests de sécurité, la surveillance et la réponse aux incidents. La conception de réseau segmenté avec des systèmes délibérément vulnérables et des outils de sécurité de niveau entreprise permet des exercices et des scénarios de formation en sécurité réalistes.

Les composants clés de l'architecture comprennent :

- Pare-feu OPNsense pour la segmentation réseau
- Garuda Linux pour les tests d'intrusion
- VMs Ubuntu vulnérables pour les tests de sécurité
- SIEM Wazuh pour la surveillance de sécurité
- Plateforme SOAR pour la réponse aux incidents

Cet environnement intégré permet un flux de travail de sécurité complet de l'attaque à la détection et à la réponse, fournissant une expérience pratique précieuse pour les professionnels de la sécurité.

8.2 Améliorations Futures

Plusieurs améliorations pourraient être apportées à l'environnement du laboratoire à l'avenir :

- Systèmes vulnérables supplémentaires avec différents types de vulnérabilités
- Intégration avec des services de sécurité basés sur le cloud
- Implémentation de technologies de déception (pots de miel, etc.)
- Capacités avancées d'analyse du trafic réseau
- Détection d'anomalies basée sur l'apprentissage automatique

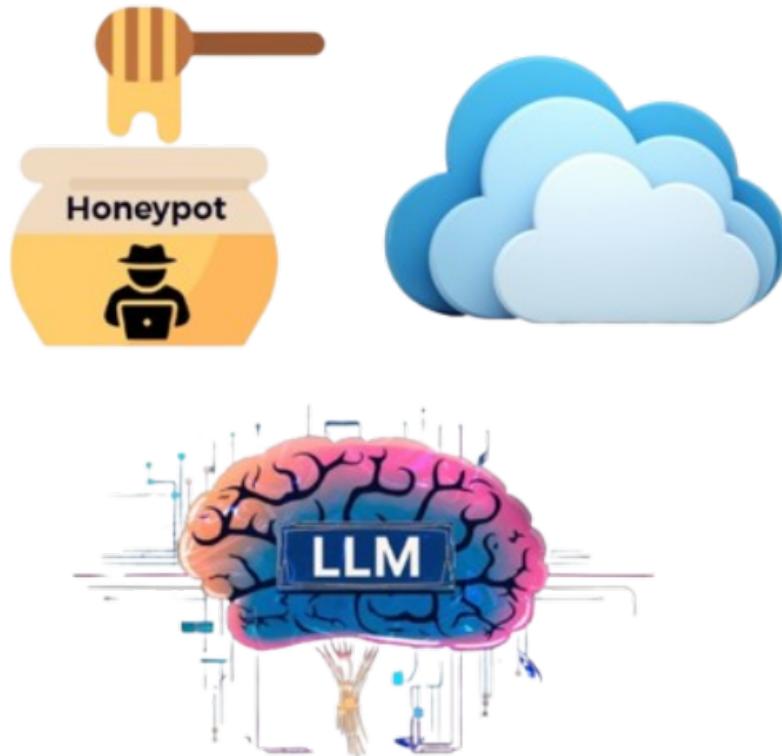


Figure 8.1: Améliorations futures potentielles de l'architecture du laboratoire

8.3 Réflexions Finales

Cette architecture de laboratoire de cybersécurité démontre l'importance d'une approche intégrée des tests de sécurité et de la surveillance. En combinant des outils de sécurité offensifs et défensifs dans un environnement contrôlé, les professionnels de la sécurité peuvent acquérir une expérience et des connaissances précieuses sur le cycle de vie complet de la sécurité.

L'environnement du laboratoire fournit une plateforme pour l'apprentissage continu et l'amélioration des compétences en cybersécurité, permettant aux professionnels de la sécurité de rester en avance sur les menaces et les vulnérabilités en évolution.

A

Scripts d'Installation

```
...  
1 # Script d'installation d'OPNsense  
2 # Télécharger l'ISO OPNsense  
3 wget https://mirror.ams1.nl.leaseweb.net/opnsense/releases/22.7/OPNsense-22.7-dvd-  
amd64.iso.bz2  
4 bunzip2 OPNsense-22.7-dvd-amd64.iso.bz2  
5  
6 # Créer une VM et installer OPNsense  
7 # Configurer les interfaces réseau  
8 # WAN: NAT  
9 # LAN1: VMnet2 (192.168.29.0/24)  
10 # LAN2: VMnet3 (192.168.83.0/24)
```

Figure A.1: Installation du OPNSense

A.1 Règles du Pare-feu OPNsense

```
 1 <filter>
 2   <rule>
 3     <type>pass</type>
 4     <interface>vm</interface>
 5     <ipprotocol>inet</ipprotocol>
 6     <source>
 7       <network>192.168.29.10/32</network>
 8       <network>192.168.29.11/32</network>
 9       <network>192.168.29.12/32</network>
10     </source>
11     <destination>
12       <network>192.168.83.128/32</network>
13     </destination>
14     <protocol>tcp</protocol>
15     <description>Autoriser les VMs Vulnérables vers le SIEM Wazuh</description>
16   </rule>
17
18   <rule>
19     <type>block</type>
20     <interface>lan</interface>
21     <ipprotocol>inet</ipprotocol>
22     <source>
23       <network>192.168.29.13/32</network>
24     </source>
25     <destination>
26       <network>192.168.83.128/32</network>
27     </destination>
28     <description>Bloquer Garuda vers le SOC</description>
29   </rule>
30 </filter>
```

Figure A.2: Regles du OPNSense

Bibliography

- [1] Wazuh Documentation, *Manuel Utilisateur Wazuh*, <https://documentation.wazuh.com/current/index.html>
- [2] The Hive Project, *Guide Utilisateur The Hive*, <https://docs.thehive-project.org/>
- [3] Documentation Cortex, *Guide Utilisateur Cortex*, <https://github.com/TheHive-Project/CortexDocs>
- [4] Projet MISP, *Guide Utilisateur MISP*, <https://www.misp-project.org/documentation/>
- [5] Documentation OPNsense, *Manuel Utilisateur OPNsense*, <https://docs.opnsense.org/>
- [6] Garuda Linux, *Documentation Garuda Linux*, <https://wiki.garudalinux.org/>
- [7] Documentation Docker, *Guide Utilisateur Docker Compose*, <https://docs.docker.com/compose/>
- [8] Documentation VMware, *Guide Utilisateur VMware Workstation Pro*, <https://docs.vmware.com/en/VMware-Workstation-Pro/>
- [9] Documentation Ubuntu, *Guide du Serveur Ubuntu*, <https://ubuntu.com/server/docs>