

République Tunisienne  
Ministère de l'Enseignement Supérieur  
et de la Recherche Scientifique  
École Supérieur Privée d'ingénierie et de  
technologie  
**TEK-UP**

## RAPPORT DE PROJET DE FIN D'ANNÉE

Présenté en vue de l'obtention de la  
DIPLÔME NATIONAL D'INGÉNIEUR EN SCIENCES APPLIQUÉES ET  
TECHNOLOGIQUES

Sécurité des Systèmes Informatiques et Réseaux

---

# Conception et Mise en place d'une Solution NOC Open Source

---

*Par* BEDHIEFI ROUA  
TERRES DOUAA

Réalisé au sein de



Encadrant académique : Monsieur HDIJI Tarek

Année Universitaire : 2025-2026

# Table des matières

<b>1 Cadre du projet</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Présentation du projet . . . . .	1
1.2.1 Etude de l'existant . . . . .	1
1.2.2 Critique de l'existant . . . . .	2
1.2.3 Solution proposée . . . . .	2
1.2.4 Architecture générale de la solution proposée . . . . .	3
1.3 Gestion de projet . . . . .	3
1.3.1 Comparaison entre la méthodologie classique et la méthodologie agile	3
1.3.2 Méthodologie Agile/Scrum . . . . .	4
1.4 Conclusion . . . . .	6
<b>2 Etude préalable</b>	<b>7</b>
2.1 Introduction . . . . .	7
2.2 Les concepts théoriques : . . . . .	7
2.2.1 La Supervision . . . . .	7
2.2.2 NOC (Network Operations Center) . . . . .	9
2.2.3 SOC (Security Operations Center) . . . . .	10
2.2.4 Comparaison entre NOC et SOC . . . . .	10
2.2.5 Elements clés . . . . .	11
2.3 Etude comparative . . . . .	12
2.3.1 Démarche de l'étude comparative . . . . .	12
2.3.2 Outils de supervision Open source . . . . .	13
2.3.3 Outils de Collecte et d'Analyse des Logs (SIEM) . . . . .	14
2.3.4 Outils de Détection d'Intrusion (IDS/IPS) . . . . .	16
2.3.5 Outils de Threat Intelligence . . . . .	17
2.3.6 Outils de Visualisation et Tableaux de Bord . . . . .	18
2.3.7 Synthèse et choix . . . . .	19
2.3.8 Architecture finale . . . . .	19
2.4 Conclusion . . . . .	20
<b>3 Sprint 1 . Mise en place de la supervision</b>	<b>21</b>
3.1 Introduction . . . . .	21
3.2 Sprint backlog . . . . .	21
3.3 Environnement de travail . . . . .	22
3.3.1 Environnement matériel : . . . . .	22
3.3.2 Environnement logiciel : . . . . .	22
3.4 Mise en place de Zabbix . . . . .	22

3.4.1	Installation zabbix Server . . . . .	22
3.4.2	Ajout des agents . . . . .	23
3.4.3	Supervision des hôtes dans Zabbix . . . . .	26
3.5	Intégration Zabbix et Kibana . . . . .	38
3.6	Conclusion . . . . .	39
<b>4</b>	<b>Sprint 2 : Mise en place du SOC</b>	<b>40</b>
4.1	Introduction . . . . .	40
4.2	Sprint backlog . . . . .	40
4.3	Environnement de travail . . . . .	41
4.3.1	Environnement logiciel : . . . . .	41
4.4	Mise en place de ELK . . . . .	41
4.4.1	Mise en place de Elasticsearch . . . . .	41
4.4.2	Mise en place de logstash . . . . .	42
4.4.3	Mise en place de kibana . . . . .	43
4.5	Déploiement des agents de collecte . . . . .	44
4.5.1	mise en place du filebeat dans la hote ubuntu server . . . . .	44
4.5.2	mise en place du WinLogbeat ds la hot Windows server . . . . .	46
4.6	Mise en place du Suricata . . . . .	47
4.6.1	Activation de la sortie JSON (eve.json) . . . . .	47
4.6.2	Définition de l'interface réseau à surveiller . . . . .	48
4.6.3	Mise à jour des règles de détection . . . . .	48
4.7	Intégration Suricata et ELK . . . . .	49
4.7.1	mise en place du filebeat : . . . . .	49
4.8	Collecte, normalisation et corrélation des logs . . . . .	50
4.8.1	Collecte des différentes sources de logs . . . . .	51
4.8.2	Parsing et traitement . . . . .	52
4.8.3	Normalisation des données dans Elasticsearch . . . . .	54
4.8.4	Visualisations dans Kibana . . . . .	54
4.9	Test et résultat . . . . .	54
4.10	Conclusion . . . . .	58
<b>5</b>	<b>Sprint 3 : Mise en place du threat intelligence</b>	<b>59</b>
5.1	Introduction . . . . .	59
5.2	Sprint backlog . . . . .	59
5.3	Environnement de travail . . . . .	60
5.3.1	Environnement matériel : . . . . .	60
5.3.2	Environnement logiciel : . . . . .	60
5.4	Mise en place du Open CTI . . . . .	60
5.4.1	Configuration du OpenCTI . . . . .	61
5.4.2	Mise en place du MITRE ATTACK . . . . .	63
5.5	Conclusion . . . . .	64
<b>Conclusion Générale</b>		<b>65</b>

# Table des figures

1.1	L'architecture générale du projet . . . . .	3
1.2	Vue globale du processus Scrum . . . . .	6
2.3	Architecture finale après les choix des outils . . . . .	20
3.4	Logo VMware . . . . .	22
3.5	Dashboard zabbix . . . . .	23
3.6	fichier de configuration Zabbix-agent . . . . .	24
3.7	le processus d'ajout de l'hôte Ubuntu server dans Zabbix . . . . .	25
3.8	le processus d'ajout de l'hôte windows server dans Zabbix . . . . .	25
3.9	l'état des hôtes sur zabbix . . . . .	26
3.10	Ajout d'un élément pour surveiller l'utilisation CPU . . . . .	28
3.11	Création d'un déclencheur (Trigger) pour surveiller l'utilisation CPU . . . . .	29
3.12	Ajout d'un élément pour l'utilisation de la mémoire RAM . . . . .	30
3.13	Création du déclencheur pour l'utilisation de la mémoire RAM . . . . .	31
3.14	. . . . .	31
3.15	Création d'un élément pour surveiller le processus tomcat . . . . .	32
3.16	Création d'un élément pour surveiller le processus tomcat . . . . .	32
3.17	Résultat de l'alerte tomcat is not running on server ubuntu . . . . .	33
3.18	Création d'un Item pour vérifier la disponibilité du port SSH . . . . .	34
3.19	Création d'un Déclencheur pour alerter en cas d'indisponibilité du service SSH . . . . .	34
3.20	Création d'un élément pour surveiller le processus SSH . . . . .	35
3.21	Création d'un déclencheur pour surveiller le processus SSH . . . . .	36
3.22	. . . . .	36
3.23	Collecte des métriques de l'hôte . . . . .	37
3.24	Résultat du métrique . . . . .	37
3.25	Evaluation du service DNS . . . . .	37
3.26	Résultat . . . . .	38
3.27	Evaluation de la consommation de la CPU . . . . .	38
3.28	le résultat de l'alerte alert RAM . . . . .	38
3.29	génération du token . . . . .	39
3.30	pipeline lssssssogstash dans /etc/logstash/conf.d/zabbix.conf . . . . .	39
4.31	Logo VMware . . . . .	41
4.32	. . . . .	42
4.33	Activation du sécurité dans elasticseach . . . . .	42
4.34	configuration du logstash . . . . .	43
4.35	activation d'authentification dans kibana . . . . .	44
4.36	Tableau de bord de kibana . . . . .	44
4.37	Vérification d'installation du filebeat . . . . .	45
4.38	configuration des modules dans filbeat . . . . .	45

4.39 activation des modules dans filbeat . . . . .	45
4.40 validation du config . . . . .	45
4.41 configuration d'envoie des logs . . . . .	46
4.42 Démarrage du filebeat . . . . .	46
4.43 Installation du winlogbeat . . . . .	46
4.44 Choix des logs windows collectés . . . . .	47
4.45 configuration du winlogbeat . . . . .	47
4.46 activation de la sortie json . . . . .	48
4.47 définition d'interface . . . . .	48
4.48 Rédemarrage du suricata . . . . .	49
4.49 configuration des modules . . . . .	49
4.50 configuration de la sortie vers elasticsearch . . . . .	50
4.51 configuration de la sortie vers kibana . . . . .	50
4.52 extrait des logs du serveur ubuntu . . . . .	51
4.53 extrait des logs du serveur windows . . . . .	51
4.54 extrait des logs de l'ids . . . . .	52
4.55 logs du serveur ubuntu parsés . . . . .	53
4.56 logs du réseau parsés . . . . .	53
4.57 logs du serveur windows parsés . . . . .	54
4.58 attaque brute-force simulé . . . . .	55
4.59 alerte généré par suricata . . . . .	55
4.60 création du règle . . . . .	56
4.61 alerte affichée dans kibana . . . . .	56
4.62 attaque brute-force dans le serveur ubuntu . . . . .	57
4.63 alerte généré par suricata . . . . .	57
4.64 création du règle de détection dans kibana . . . . .	58
4.65 alerte affichée . . . . .	58
5.66 Logo VMware . . . . .	60
5.67 Configuration du fichier docker-compose . . . . .	61
5.68 Configuration des variables . . . . .	62
5.69 Lancement du service OpenCTI . . . . .	62
5.70 Tableau de bord du OpenCTI . . . . .	63
5.71 Configuration des paramètres de MITR ATTACK . . . . .	63

# Liste des tableaux

1.1	Comparaison entre méthodologie classique et méthodologie agile scrum . . . . .	4
2.2	Tableau comparatif NOC vs SOC . . . . .	11
2.3	Tableau comparatif des outils de supervision Open Source . . . . .	14
2.4	Tableau comparatif des outils de collecte et d'analyse des logs Open Sourc . . . . .	15
2.5	Tableau comparatif des outils de Détection d'Intrusion Open Source . . . . .	16
2.6	Tableau comparatif des outils de Threat Intelligence Open Source . . . . .	17
2.7	Tableau comparatif des outils de Visualisation et Tableaux de Bord Open Source . . . . .	18
2.8	Tableau récapitulatif des outils choisis . . . . .	19
3.9	sprint backlog 1 . . . . .	21
3.10	caractéristiques matérielles et logicielles . . . . .	22
4.11	sprint backlog 2 . . . . .	40
4.12	caractéristiques matérielles et logicielles . . . . .	41
5.13	sprint backlog 3 . . . . .	59
5.14	caractéristiques matérielles et logicielles . . . . .	60

# **chapitre 1**

## **Cadre du projet**

### **1.1 Introduction**

Dans ce chapitre, nous inscrivons le projet dans son contexte global. Nous commençons par présenter le cadre général, en introduisant la problématique, la solution proposée ainsi que l'étude de l'existant. Par la suite, nous détaillons la méthode de travail adoptée pour la réalisation du projet.

### **1.2 Présentation du projet**

Ce projet s'inscrit dans une démarche d'amélioration de la supervision et de la sécurité des infrastructures informatiques virtualisées. Il vise à concevoir et à mettre en place une solution open source intégrée permettant à la fois la surveillance en temps réel des ressources systèmes et réseaux, ainsi que la détection et l'analyse des incidents de sécurité.

#### **1.2.1 Etude de l'existant**

Les infrastructures informatiques modernes se composent de serveurs, de réseaux et d'applications interconnectés, mais elles souffrent souvent d'un manque de supervision et de sécurité unifiées. Les outils utilisés sont généralement dispersés, difficiles à gérer et ne permettent pas une visibilité complète sur l'état global du système. De plus, l'absence d'une centralisation des logs et d'une corrélation des événements rend la détection d'incidents complexe et lente.

Dans un tel environnement, la gestion des incidents repose principalement sur des interventions manuelles, ce qui augmente le temps de réaction et le risque de défaillance. Les alertes, lorsqu'elles existent, ne couvrent qu'une partie des équipements ou services critiques, laissant certaines anomalies ou tentatives d'attaques passer inaperçues.

Cette situation met en évidence la nécessité d'une solution intégrée capable de superviser à la fois les performances de l'infrastructure et les aspects liés à la sécurité. Une telle solution doit permettre de centraliser la supervision, d'automatiser la détection des anomalies, de corrélérer les événements techniques et sécuritaires, et d'améliorer la réactivité des équipes face aux incidents.

C'est dans cette optique que s'inscrit notre projet, visant à concevoir et à mettre en place

une solution NOC/SOC open source permettant une surveillance complète, une détection proactive des menaces et une centralisation efficace des informations au sein d'une infrastructure virtuelle.

### 1.2.2 Critique de l'existant

Les environnements informatiques actuels, bien que fonctionnels, présentent plusieurs limites importantes en matière de supervision et de sécurité. L'absence d'une plateforme centralisée pour la surveillance et la détection des incidents conduit à une gestion fragmentée des systèmes et des réseaux. Les outils utilisés sont souvent disparates, nécessitant des configurations indépendantes et ne permettant pas une corrélation efficace des informations. Cette absence d'unification rend la supervision globale complexe et peu efficace. En cas de panne, d'erreur système ou d'activité suspecte, les interventions doivent être réalisées manuellement, ce qui augmente considérablement les délais de réaction et les risques d'erreurs humaines. De plus, le manque de visibilité en temps réel sur l'état des serveurs, des applications et du réseau empêche toute anticipation des anomalies ou des menaces potentielles. Sur le plan de la sécurité, l'absence de centralisation des journaux système et d'un mécanisme de détection automatisée limite la capacité à identifier les attaques réseau ou les comportements anormaux. Les alertes, lorsqu'elles existent, ne sont pas corrélées avec les autres événements de l'infrastructure, ce qui réduit la pertinence de l'analyse et la rapidité de réponse. Ainsi, il devient nécessaire d'adopter une approche plus intégrée et intelligente, combinant supervision (NOC) et sécurité opérationnelle (SOC). Une telle solution permettrait de centraliser les données, d'automatiser les alertes, d'améliorer la visibilité sur l'ensemble du système et d'optimiser la détection des incidents techniques et sécuritaires.

### 1.2.3 Solution proposée

La solution envisagée dans le cadre de ce projet vise à concevoir et à mettre en place une plateforme open source intégrée combinant supervision opérationnelle (NOC) et sécurité opérationnelle (SOC). Cette approche permettra d'unifier la surveillance de l'infrastructure, la détection des anomalies et la corrélation des événements de sécurité au sein d'un environnement unique. La partie NOC assurera la supervision centralisée de l'infrastructure virtuelle, incluant les serveurs, les services applicatifs et les ressources réseau. Elle permettra de surveiller en temps réel les performances du système, de générer des alertes automatiques en cas de dysfonctionnement et de fournir une vue globale de l'état de l'environnement. La partie SOC, quant à elle, sera chargée de la collecte, de l'analyse et de la corrélation des logs afin de détecter les activités suspectes et les incidents de sécurité. Elle intégrera également un système de détection d'intrusions (IDS/IPS) pour identifier les attaques réseau et un moteur de renseignement sur les menaces (Threat Intelligence) afin d'enrichir les alertes par des indicateurs de compromission connus. L'ensemble de ces composants sera interconnecté au sein d'une même architecture afin d'assurer une supervision complète et proactive. Cette intégration favorisera une meilleure visibilité sur l'état global du système, une réduction du temps de réaction face aux incidents, et une optimisation des performances et de la sécurité de l'infrastructure virtuelle.

### 1.2.4 Architecture générale de la solution proposée

Nous avons présenté l'architecture de notre solution, qui vise à mettre en place une infrastructure intégrée combinant un NOC et un SOC open source, afin d'assurer à la fois la supervision opérationnelle et la sécurité globale de l'environnement informatique. Cette architecture permet de centraliser la surveillance des serveurs, des applications et du réseau, tout en collectant et corrélant les événements de sécurité issus de différentes sources. Elle offre une vision unifiée de l'état de l'infrastructure, facilite la détection proactive des anomalies et des menaces, et améliore la réactivité face aux incidents. Grâce à cette intégration, la supervision technique et la sécurité opérationnelle sont automatisées et consolidées au sein d'une même plateforme, garantissant ainsi une meilleure disponibilité, performance et protection de l'ensemble du système.

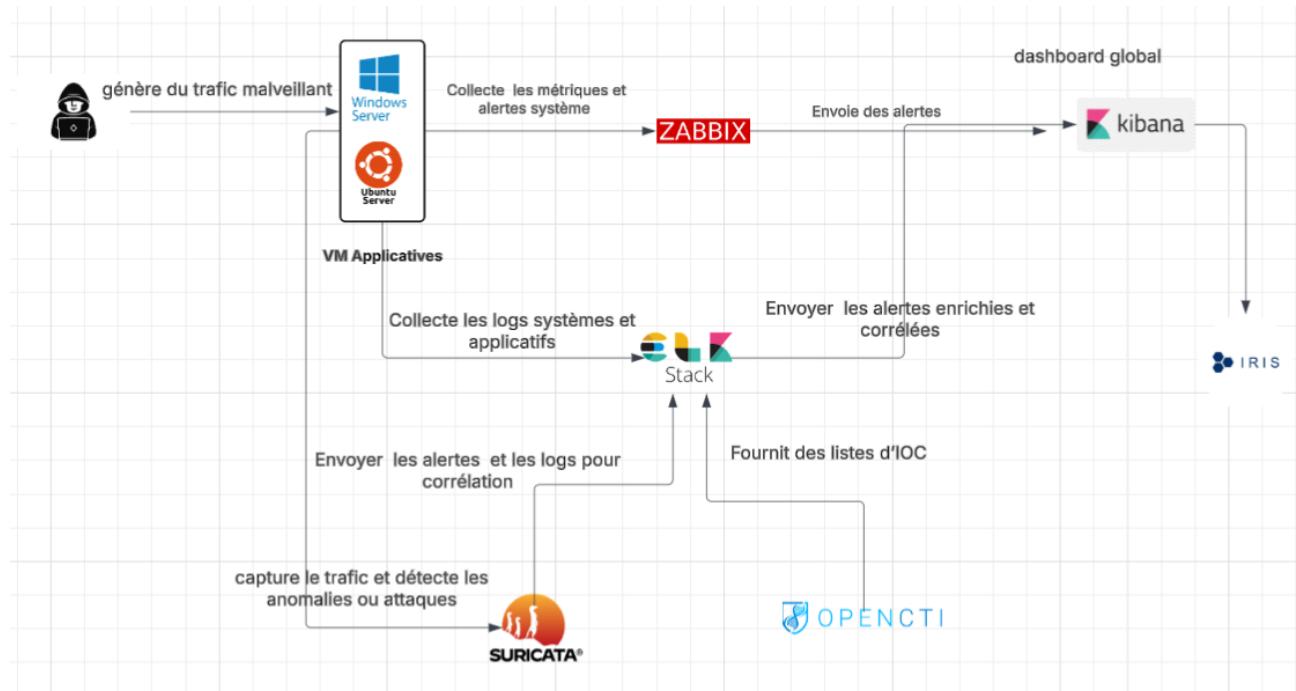


FIGURE 1.1 – L'architecture générale du projet

## 1.3 Gestion de projet

Avant de débuter ce projet, il est primordial de sélectionner une méthodologie de travail et de définir un planning. Cela permet d'organiser les différentes phases du projet, d'identifier les tâches et les ressources nécessaires, de planifier les délais et les livrables, et de garantir la qualité et la cohérence du travail effectué tout en minimisant les risques et les incertitudes .

### 1.3.1 Comparaison entre la méthodologie classique et la méthodologie agile

Les méthodologies classiques et agiles sont deux approches de gestion de projet distinctes. les méthodologies classiques de gestion de projet, comme le modèle en cascade,

reposent sur une planification rigide et linéaire où chaque phase (analyse, conception, développement, test, déploiement) est réalisée successivement. Cette approche nécessite que toutes les exigences soient définies dès le départ, ce qui limite la flexibilité et la capacité à s'adapter aux changements en cours de projet. À l'inverse, Scrum, en tant que méthodologie agile, adopte une approche itérative et incrémentale, favorisant l'adaptation continue, la collaboration étroite entre les parties prenantes et la livraison rapide de fonctionnalités. Elle est particulièrement bien adaptée aux projets complexes ou évolutifs, comme ceux liés à la sécurité et à l'infrastructure IT, où les besoins peuvent changer en fonction des risques ou des technologies.

Le tableau 1.1 présente une comparaison entre la méthodologie classique en cascade et la méthodologie agile Scrum, en mettant en évidence leurs principales différences en termes de gestion de projet.

Critère	cascade	Scrum
Approche	Linéaire et séquentielle	Itérative et incrémentale
Planification	Fixe dès le départ	Révisée à chaque sprint
Flexibilité	Faible	Élevée
Livraison	En une seule fois à la fin du projet	Continue à la fin de chaque sprint
Gestion des changements	Difficile à intégrer	Facile à adapter en cours de projet
Implication du client	Faible, surtout en début et fin	Forte et continue
Documentation	Très détaillée dès le départ	Juste suffisante (juste-en-temps)
Adaptabilité	Moins adaptée aux environnements changeants	Très adaptée aux projets dynamiques
Contrôle qualité	En fin de cycle	À chaque sprint (intégration continue)

TABLE 1.1 – Comparaison entre méthodologie classique et méthodologie agile scrum

Après avoir comparé les deux méthodes, nous avons décidé d'adopter la méthodologie agile pour notre projet et plus précisément la méthodologie scrum .

### 1.3.2 Méthodologie Agile/Scrum

Scrum est un cadre de processus flexible et léger qui met l'accent sur le travail d'équipe, la responsabilité et les processus itératifs pour le développement de produits. De plus en plus d'organisations dans le monde adoptent Scrum, notamment dans le cadre de projets de développement de logiciels agiles. En appliquant les principes de Scrum dans un programme de cybersécurité, les administrateurs de la sécurité peuvent développer, mettre en œuvre et gérer efficacement les processus et les contrôles.

Il existe quatre principes de Scrum particulièrement pertinents pour les projets de cybersécurité :

- **Transparence** : Les processus doivent être visibles et compréhensibles pour toutes

les personnes impliquées et les membres de l'équipe doivent avoir une compréhension commune des objectifs du projet.

- **Inspection** : Le travail doit être régulièrement examiné pour identifier rapidement les problèmes qui pourraient avoir un impact sur l'objectif du projet.
- **Adaptation** : Si un aspect du processus empêche les livrables du projet d'être utilisables, le processus doit être adapté rapidement pour garantir la qualité et les objectifs de l'entreprise.
- **Délimitation du temps** : Les événements Scrum ont des durées minimales et maximales définies pour garantir une utilisation efficace du temps et une valeur commerciale maximale dans les délais les plus courts.

#### Les rôles dans Scrum :

**Product Owner** : Il représente les besoins du client. C'est un expert métier chargé de définir les fonctionnalités du produit, de les prioriser et de maintenir à jour le *Product Backlog*. Il collabore étroitement avec l'équipe tout au long du projet.

**Scrum Master** : Il veille à la bonne application de la méthodologie Scrum. Il aide l'équipe à surmonter les obstacles et s'assure que les pratiques agiles sont respectées durant chaque itération.

**Équipe de développement** : Composée de membres auto-organisés, elle est responsable de livrer un incrément fonctionnel du produit à chaque fin de sprint. Il n'y a pas de hiérarchie formelle au sein de l'équipe.

#### Le fonctionnement itératif :

**Sprint** : C'est une période de travail fixe, généralement de 2 à 4 semaines, durant laquelle l'équipe développe une partie concrète et livrable du produit.

#### Les artefacts Scrum :

**Product Backlog** : Il s'agit de la liste complète et priorisée des fonctionnalités à développer (sous forme de User Stories), élaborée en collaboration avec le product owner.

**Sprint Backlog** : Sous-ensemble du product Backlog, il contient les éléments que l'équipe s'engage à livrer à la fin du sprint.

#### Les réunions Scrum :

**Planification du Sprint** : Réunion en début de sprint permettant à l'équipe de choisir les tâches à réaliser en fonction des priorités du product owner.

**Revue de Sprint** : À la fin de chaque sprint, cette réunion permet de présenter les livrables réalisés au Product Owner et d'ajuster le product backlog si besoin.

**Rétrospective de Sprint** : Elle se tient après la revue de sprint. L'équipe y discute des points d'amélioration du processus et identifie des actions concrètes pour optimiser les prochains sprints.

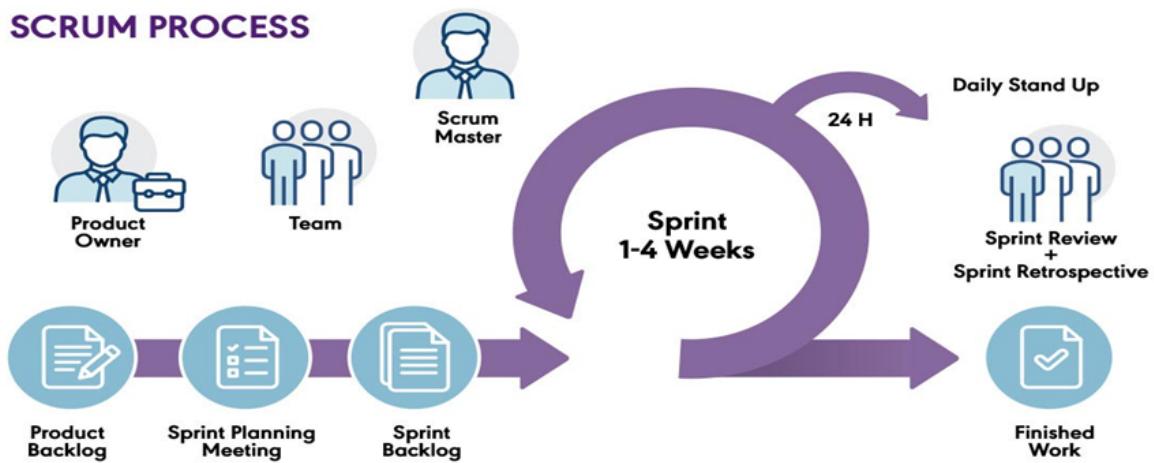


FIGURE 1.2 – Vue globale du processus Scrum

## 1.4 Conclusion

Dans ce chapitre, nous avons défini le cadre général de notre projet ainsi que la méthodologie adoptée pour sa mise en œuvre. Le chapitre suivant sera consacré à l'étude théorique des principaux concepts liés au projet, accompagnée d'une analyse comparative des approches et solutions existantes.

# chapitre 2

## Etude préalable

### 2.1 Introduction

Dans ce chapitre, nous présentons les concepts essentiels liés à la supervision des infrastructures (NOC), à la sécurité opérationnelle (SOC), ainsi qu'aux mécanismes de détection d'intrusion (IDS/IPS), de corrélation d'événements (SIEM) et de Threat Intelligence. Nous exposons ensuite les notions théoriques nécessaires à la compréhension de notre démarche, avant de réaliser une analyse comparative des solutions open source et de conclure par les choix technologiques adoptés pour la mise en œuvre de notre projet.

### 2.2 Les concepts théoriques :

Dans cette section, nous explorerons les concepts théoriques en fournissant :

#### 2.2.1 La Supervision

##### Définition

La supervision, également connue sous le terme de monitoring , est le processus de surveillance et de gestion en temps réel des performances, de la disponibilité, et de la sécurité de l'ensemble des composants d'une infrastructure informatique, incluant les équipements matériels, logiciels, serveurs, bases de données, et applications. Elle vise à détecter et résoudre rapidement les anomalies, à assurer une performance optimale, et à protéger l'infrastructure contre les menaces. L'objectif principal est de s'assurer que tous les éléments de l'infrastructure fonctionnent correctement et efficacement, tout en permettant une réponse rapide aux incidents.

##### Rôle de la supervision

La supervision joue un rôle essentiel pour garantir le bon fonctionnement, la performance optimale et la sécurité des réseaux informatiques. Voici ses principales fonctions :

**La disponibilité :** La supervision permet de s'assurer que tous les éléments du réseau (serveurs, routeurs, commutateurs, etc.) sont opérationnels en permanence. En cas de panne ou d'indisponibilité, les administrateurs réseau sont immédiatement alertés pour intervenir et rétablir le service.

**Les performances :** La supervision analyse les flux de données sur le réseau pour détecter les goulots d'étranglement et les points de congestion, ce qui permet d'optimiser la

répartition des ressources et d'améliorer la performance globale du réseau. En fournissant des données en temps réel, elle permet également aux administrateurs de prendre des mesures préventives pour éviter les ralentissements ou les dégradations de service.

**La sécurité** : La supervision des réseaux est essentielle pour assurer la sécurité en détectant les activités suspectes ou malveillantes, comme les tentatives d'intrusion ou les attaques par déni de service (DDoS), ce qui permet de réagir rapidement avant que ces menaces n'affectent la sécurité du réseau. Elle joue également un rôle clé dans la conformité aux politiques de sécurité et aux réglementations en matière de protection des données, en fournissant des journaux d'événements et des rapports d'audit détaillés.

### Aspects de la supervision

Les aspects de la supervision couvrent différents domaines de l'infrastructure informatique, chacun ayant un rôle spécifique dans la gestion et le maintien de la performance, de la disponibilité, et de la sécurité des systèmes. Voici les principaux aspects de la supervision :

**La supervision réseau** : assure la disponibilité et la performance des connexions en surveillant la bande passante, la latence, et le taux d'erreurs. Elle inclut également la vérification des équipements tels que les routeurs, switches, pare-feux, et autres dispositifs réseau pour garantir leur bon fonctionnement. De plus, la gestion des adresses IP fait partie de cette supervision, en vérifiant la disponibilité des adresses et l'accessibilité des ressources réseau.

**La supervision système** : Contrôle l'utilisation des ressources matérielles telles que le CPU, la mémoire, l'espace disque, et la température des composants, tout en assurant que les services critiques comme les serveurs web et les bases de données sont opérationnels et fonctionnent correctement. Elle inclut également l'analyse des journaux systèmes pour détecter les erreurs, les alertes de sécurité, et les anomalies dans le comportement des machines.

**La supervision des services cloud** : Surveille l'utilisation des ressources cloud, y compris les instances virtuelles, le stockage, et les services en ligne, tout en s'assurant que ces services sont disponibles et répondent aux attentes en matière de performance. Elle inclut également l'optimisation des coûts en surveillant l'utilisation des services pour minimiser les dépenses associées aux ressources cloud.

Ces aspects de la supervision travaillent ensemble pour garantir une vue complète et un contrôle efficace de l'ensemble de l'infrastructure informatique, assurant ainsi que les systèmes sont performants, sécurisés et disponibles pour les utilisateurs.

### Avantages de la supervision

La supervision offre de nombreux avantages essentiels pour la gestion des systèmes informatiques, notamment :

**Détection rapide des anomalies** : permet d'identifier et de réagir rapidement aux problèmes avant qu'ils n'affectent gravement l'infrastructure.

**Amélioration de la performance** : optimise l'utilisation des ressources pour assurer une performance maximale des systèmes et des applications.

**Réduction des temps d'arrêt** : minimise les interruptions de service en permettant une intervention proactive et rapide.

**Sécurité renforcée** : surveille les menaces potentielles et les activités suspectes pour protéger les systèmes contre les attaques.

### 2.2.2 NOC (Network Operations Center)

#### Définition

Un Network Operations Center (NOC), ou centre d'opérations réseau, est une structure technique dédiée à la surveillance, la gestion et la maintenance des infrastructures informatiques et réseau. Son objectif principal est de garantir la disponibilité, la performance et la fiabilité de l'ensemble des ressources informatiques, qu'il s'agisse de serveurs, de services applicatifs ou d'équipements réseau. Le NOC agit comme un centre de contrôle centralisé, où les administrateurs peuvent observer en temps réel l'état de l'infrastructure, détecter les anomalies et intervenir rapidement en cas d'incident.

#### Fonctions principales

Le NOC remplit plusieurs fonctions essentielles au bon fonctionnement d'une infrastructure IT :

**Supervision en temps réel** : surveillance continue des ressources (CPU, mémoire, trafic réseau, disponibilité des services).

**Maintenance proactive** : détection précoce des anomalies et exécution d'actions correctives pour éviter les pannes.

**Alerting et notifications** : émission d'alertes automatiques en cas de défaillance, envoyées par e-mail, SMS ou tableau de bord.

**Reporting et analyse** : génération de rapports périodiques sur la performance, la disponibilité et la qualité de service.

**Gestion des incidents** : suivi du cycle de vie des incidents depuis la détection jusqu'à leur résolution.

#### Fonctionnement du NOC

Le fonctionnement d'un NOC repose sur un ensemble d'agents installés sur les serveurs et équipements du réseau, chargés de collecter les données de supervision (métriques, journaux, états des services, etc.). Ces données sont ensuite transmises à un serveur central de supervision qui les analyse, les stocke et déclenche des alertes en cas d'anomalie.

Les opérateurs du NOC utilisent des interfaces graphiques (tableaux de bord) pour visualiser en temps réel la santé du système, identifier les problèmes, et coordonner les actions correctives.

Le NOC peut également être intégré à d'autres systèmes d'information (comme un SOC ou une CMDB) pour enrichir les analyses et améliorer la réactivité globale.

#### Avantages d'un NOC

L'adoption d'un NOC présente de nombreux avantages :

**Centralisation de la supervision** : toutes les informations critiques sont regroupées dans une seule plateforme.

**Réduction du temps d'intervention** : les incidents sont détectés et signalés en temps réel.

**Amélioration de la disponibilité** : la maintenance proactive limite les interruptions

de service.

**Optimisation des performances** : le suivi constant des ressources permet d'anticiper les surcharges ou dysfonctionnements.

**Amélioration de la coordination** : le NOC facilite la communication entre les équipes techniques et la hiérarchisation des priorités d'intervention.

**Base de support pour la sécurité (SOC)** : en intégrant la supervision technique à la sécurité, le NOC devient un pilier pour la détection rapide des anomalies liées à des attaques.

### 2.2.3 SOC (Security Operations Center)

#### Définition

Un SOC (Security Operations Center), ou Centre des Opérations de Sécurité, est une entité centralisée au sein d'une organisation chargée de surveiller, détecter, analyser et répondre en temps réel aux incidents de sécurité informatique. Il regroupe des outils, des processus et une équipe spécialisée (analystes, ingénieurs sécurité, responsables SOC) qui travaillent ensemble pour protéger les systèmes d'information, les réseaux, les applications et les données contre les cybermenaces.

Le SOC collecte et corrèle en continu les données issues de différentes sources (pare-feu, IDS/IPS, endpoints, serveurs, applications, etc.) grâce à des solutions comme les SIEM (Security Information and Event Management) et les SOAR (Security Orchestration, Automation and Response), dans le but de :

- Déetecter les comportements anormaux ou malveillants.
- Réagir rapidement aux incidents.
- Assurer une surveillance continue 24/7.
- Renforcer la posture de cybersécurité de l'organisation.

### 2.2.4 Comparaison entre NOC et SOC

La gestion moderne des infrastructures informatiques repose sur deux centres opérationnels complémentaires :

le NOC (Network Operations Center), dédié à la supervision et à la performance des systèmes, et le SOC (Security Operations Center), axé sur la détection et la réponse aux incidents de sécurité. Bien qu'ils partagent une approche commune de surveillance continue, leurs objectifs et leurs domaines d'action diffèrent. Le tableau suivant résume les principales distinctions entre ces deux entités.

Aspect	NOC (Network Operations Center)	SOC (Security Operations Center)
<b>Objectif</b>	Assurer la performance et la disponibilité du réseau et des services IT.	Garantir la sécurité du système d'information et détecter les menaces.
<b>Focus principal</b>	Supervision technique (pannes, performances, disponibilité).	Surveillance de la sécurité (attaques, vulnérabilités, incidents).
<b>Données traitées</b>	Indicateurs de performance : CPU, RAM, réseau, services.	Logs de sécurité, alertes IDS/IPS, IOC, anomalies.
<b>Outils typiques</b>	Zabbix, Nagios, Prometheus.	ELK, Wazuh, Suricata, OpenCTI.
<b>Type d'alerte</b>	Défaillance technique ou arrêt de service.	Activité suspecte ou attaque détectée.
<b>Équipe concernée</b>	Administrateurs réseau et systèmes.	Analystes et ingénieurs en cybersécurité.
<b>Approche</b>	Préventive et réactive.	Proactive et réactive.
<b>Objectif final</b>	Maintenir la continuité des services.	Protéger les actifs et les données sensibles.

TABLE 2.2 – Tableau comparatif NOC vs SOC

### 2.2.5 Elements clés

Dans cette section, nous définissons quelques éléments clés, à savoir SIEM, Vulnérabilité, Menace , Attaque.

SIEM :

systèmes de gestion des événements et des informations de sécurité. Par définition, les SIEM sont des systèmes centralisés qui offrent une visibilité totale sur l'activité de votre réseau et vous permettent ainsi de réagir aux menaces en temps réel. Les SIEM permettent de collecter, de lire et de catégoriser les données machine d'une grande diversité de sources, puis analysent celles-ci pour produire des informations qui vous permettront d'agir. Les SIEM sont couramment employés par les grandes entreprises et les organisations gouvernementales pour assurer la surveillance de leurs réseaux informatiques, repérer les tentatives de piratage, les attaques de logiciels malveillants, les violations de données, ainsi que toute autre activité malveillante.

Vulnérabilité :

Les vulnérabilités représentent des faiblesses au sein d'un système informatique qui, si elles sont exploitées par des attaquants, peuvent compromettre la sécurité de ce système. Ces vulnérabilités peuvent résulter d'erreurs de conception, de bugs logiciels, de configurations incorrectes ou d'une mauvaise utilisation par les utilisateurs .

Menace :

Selon la norme ISO 27001, une menace se définit comme une cause potentielle d'un incident qui pourrait endommager ou détruire un actif, ou perturber les activités informatiques. En revanche, la source d'une menace désigne l'intention et les techniques utilisées pour exploiter délibérément cette menace afin de provoquer des dommages.

Attaque :

Une attaque informatique est une action malveillante menée par un individu ou un programme dans le but de compromettre la sécurité d'un système d'information. Elle vise généralement à accéder, modifier, voler ou détruire des données, ou encore à perturber le fonctionnement normal d'un service ou d'un réseau.

## 2.3 Etude comparative

Pour concevoir et mettre en place une solution NOC intégrée à un SOC open source, il est essentiel de sélectionner une combinaison cohérente d'outils capables d'assurer la supervision, la collecte et corrélation des logs, la détection d'intrusions et la gestion du renseignement sur les menaces. Cependant, face à la grande diversité d'outils open source disponibles sur le marché, le choix des composants les plus adaptés constitue une étape cruciale et parfois complexe. Ainsi, une étude comparative approfondie s'impose afin d'identifier les solutions les plus performantes, les plus compatibles et les mieux adaptées aux besoins techniques et fonctionnels de notre projet.

L'outil retenu doit non seulement répondre aux exigences techniques (performance, compatibilité, facilité d'intégration), mais aussi offrir une souplesse d'exploitation et une pérennité dans le temps.

### 2.3.1 Démarche de l'étude comparative

Le choix d'une combinaison homogène et interopérable d'outils de supervision et de sécurité constitue un facteur clé de réussite pour la mise en place d'une architecture NOC/SOC intégrée. Compte tenu de la diversité des solutions open source existantes, notre démarche repose sur une approche structurée qui comprend plusieurs étapes :

**Énumérer les outils open source les plus populaires** dans chaque catégorie (supervision, logs, IDS/IPS, Threat Intelligence, visualisation).

**Selectionner les solutions les plus pertinentes** en fonction de leur adoption, de leur stabilité et de leur compatibilité avec les autres composants.

**Élaborer des critères de comparaison rigoureux**, tels que la facilité d'installation, la compatibilité, les fonctionnalités, la performance, la sécurité et la communauté de support.

**Comparer les outils choisis** selon ces critères à travers des tableaux comparatifs détaillés, afin de dégager les forces et les limites de chaque solution.

**Synthétiser les résultats** pour identifier les outils les mieux adaptés à la conception de notre architecture intégrée NOC/SOC.

Dans la suite de ce chapitre, nous présenterons une étude comparative complète pour chaque catégorie d'outils, en suivant la démarche décrite ci-dessus.

### 2.3.2 Outils de supervision Open source

Pour une étude comparative des outils de supervision open source, nous allons examiner trois solutions largement adoptées dans le secteur : Nagios, Zabbix, et Prometheus.

#### Nagios

Nagios, ou Nagios Core est un logiciel ordonnanceur qui surveille les systèmes, les réseaux et l'infrastructure. Nagios offre des services de surveillance et d'alerte pour les serveurs, les commutateurs, les applications et les services. Il alerte les utilisateurs en cas d'incidents et les avertit une deuxième fois lorsque le problème a été résolu. Nagios a été conçu à l'origine pour fonctionner sous Linux, mais il fonctionne aussi bien sur d'autres variantes d'Unix.

#### Zabbix

Zabbix est un logiciel qui supervise de nombreux paramètres réseaux ainsi que la santé et l'intégrité des serveurs. Zabbix utilise un mécanisme de notification flexible qui permet aux utilisateurs de configurer une base d'alerte e-mail pour pratiquement tous les événements. Cela permet une réponse rapide aux problèmes serveurs.

Zabbix offre un excellent reporting et des fonctionnalités de visualisation de données basées sur les données stockées. Cela rend Zabbix idéal "for capacity planning". Zabbix supporte à la fois "polling et trapping". Tous les rapports et statistiques, comme la configuration de paramètres, sont accessibles par l'interface web. L'interface web veille à ce que le statut de votre réseau et de vos serveurs puisse être évalué depuis n'importe quel endroit. Correctement configuré, Zabbix peut jouer un rôle important dans la supervision de l'infrastructure IT. Ceci est également vrai pour les petites organisations avec peu de serveurs ainsi que pour les grandes entreprises avec une multitude de serveurs.

#### Prometheus

Prometheus est une solution de supervision novatrice. Créé à l'origine par Soundcloud en 2012, le projet a rejoint la Cloud Native Computing Foundation en 2016 en tant que deuxième projet hébergé, après Kubernetes. Basée sur la collecte de métriques et leur stockage dans une base de données maison de type séries temporelles (time series), la solution Prometheus propose une surveillance d'événements robuste avec une gestion des alertes sur seuil.

La restitution des données est assurée grâce au puissant langage de requête maison PromQL. La solution offre une interface WEB pour la restitution, mais la production de tableaux de bord avancée se fait couramment avec Grafana.

Parmi les outils de supervision, nous avons Nagios, Zabbix, et Prometheus. Pour faciliter le choix entre ces trois outils, un tableau comparatif a été élaboré pour évaluer leur facilité d'utilisation, leurs fonctionnalités, leur scalabilité, leur support communautaire, leur intégration, et leur coût.

Le tableau 2.1 compare Nagios, Zabbix, et Prometheus selon ces différents critères.

Après comparaison des solutions Zabbix, Nagios, et Prometheus, nous avons choisi Zabbix comme outil de supervision principal.

Zabbix combine une interface intuitive, une gestion centralisée des alertes, et une excellente compatibilité multi-plateforme (Linux, Windows, services applicatifs).

Critère	<b>Nagios®</b>	<b>ZABBIX</b>	 Prometheus
Fonctionnalités	Surveillance basique des systèmes et réseaux, notifications par e-mail/SMS, extensible via plugins.	Surveillance complète des serveurs, réseaux, et applications, notifications avancées, templates prédefinis	Spécialisé dans la surveillance des séries temporelles, adapté pour la surveillance d'infrastructures dynamiques (micro services, containers).
Facilité d'utilisation	Interface utilisateur moins intuitive, configuration manuelle souvent complexe.	Interface utilisateur plus moderne, configuration par interface graphique, apprentissage rapide.	interface orientée vers les développeurs, configuration via des fichiers YAML, courbe d'apprentissage plus élevée.
Scalabilité	Scalabilité limitée sans modifications significatives.	Très scalable, adapté aux grandes infrastructures avec des milliers de nœuds	Conçu pour une scalabilité massive, particulièrement dans les environnements cloud natifs.
Intégrations	Large choix de plugins, mais parfois complexes à intégrer.	Large choix d'intégrations natives et via API.	Forte intégration avec l'écosystème cloud natif (Kubernetes, Docker)
Coût	Gratuit (open-source), support commercial payant.	Gratuit (open-source), options de support commercial payant.	Gratuit (open-source), support commercial disponible via des entreprises partenaires.

TABLE 2.3 – Tableau comparatif des outils de supervision Open Source

Il offre une intégration fluide avec Kibana, permettant d'unifier la supervision opérationnelle (NOC) avec la surveillance de sécurité (SOC) au sein d'une même architecture cohérente et automatisée.

### 2.3.3 Outils de Collecte et d'Analyse des Logs (SIEM)

Pour une étude comparative des outils de collecte et d'analyse des logs open source, nous allons examiner trois solutions largement utilisées dans les environnements SOC : ELK Stack, Graylog, et Wazuh. Ces outils permettent la centralisation, la corrélation et la visualisation des événements systèmes et réseau afin d'assurer une meilleure détection

des incidents et une analyse approfondie des journaux de sécurité.

#### ELK Stack (Elasticsearch, Logstash, Kibana)

Une plateforme open source composée de trois outils :

- Elasticsearch pour l'indexation et la recherche rapide des logs,
- Logstash pour la collecte et le traitement des données
- Kibana pour la visualisation et la corrélation des événements.

Elle est largement utilisée comme base SIEM pour la centralisation et l'analyse des journaux d'activité système, réseau et sécurité.

#### Graylog

Outil open source de gestion et d'analyse de logs reposant sur Elasticsearch. Il permet la collecte centralisée, la recherche rapide et la création de tableaux de bord simples à configurer. Graylog est reconnu pour sa facilité de déploiement et sa gestion efficace des logs système et applicatifs.

#### Wazuh

Extension open source de sécurité basée sur l'ancien OSSEC. Wazuh intègre la collecte de logs, la corrélation d'événements, la surveillance de l'intégrité des fichiers et la détection d'intrusions.

Il est souvent couplé à ELK pour former une solution SIEM complète et automatisée.

Le tableau ci-dessous résume la comparaison des outils de collecte et d'analyse des logs open source .

Critère			
Fonctionnalités	Collecte, indexation et visualisation des logs	Gestion centralisée de logs	SIEM complet (logs + sécurité)
Facilité de déploiement	Moyenne (plusieurs composants)	Simple à moyenne	Moyenne à complexe
Scalabilité / Performance	Excellente, adaptée aux grands volumes	Bonnes	Bonnes mais gourmande
Corrélation d'événements	Oui, via requêtes Elasticsearch	Oui, via règles de corrélation	Oui, corrélation avancée de sécurité.
Coût	Gratuit (open-source)	Gratuit (open-source)	Gratuit (open-source)
Alerting	Watcher / ElastAlert	Alertes intégrées	Alertes centralisées

TABLE 2.4 – Tableau comparatif des outils de collecte et d'analyse des logs Open Sourc

Après avoir analysé et comparé les différentes solutions disponibles, nous avons choisi d'adopter la stack ELK (Elasticsearch, Logstash, Kibana). Ce choix s'explique par sa grande flexibilité, sa capacité à traiter de gros volumes de logs en temps réel, ainsi que son intégration fluide avec Suricata et OpenCTI. Contrairement à Graylog ou Wazuh, ELK offre une architecture hautement personnalisable et extensible, parfaitement adaptée à un environnement NOC/SOC intégré, où la corrélation et la visualisation avancée des événements sont essentielles.

### 2.3.4 Outils de Détection d’Intrusion (IDS/IPS)

Dans le cadre de la détection et de la prévention des intrusions réseau, plusieurs solutions open source se distinguent par leur efficacité et leur robustesse. Nous allons comparer Suricata, Snort, et Zeek, trois outils IDS/IPS offrant des approches complémentaires pour l’analyse du trafic, la détection d’anomalies et la prévention d’attaques.

#### Suricata

IDS/IPS open source développé par l’Open Information Security Foundation (OISF). Il effectue une inspection approfondie du trafic réseau (DPI), détecte les attaques en temps réel, et peut générer des logs de sécurité enrichis exportables vers ELK. Suricata est multi-threadé et supporte de nombreux protocoles modernes.

#### Snort

IDS/IPS développé par Cisco, très connu dans le monde de la cybersécurité. Il permet la détection d’intrusions réseau via des règles de signatures, tout en offrant un bon compromis entre performance et simplicité. C’est l’une des solutions IDS les plus largement utilisées.

#### Zeek (ex-Bro)

Outil d’analyse réseau axé sur la surveillance comportementale. Contrairement à Snort et Suricata (basés sur les signatures), Zeek analyse les modèles d’activité réseau pour identifier les comportements suspects. Il est souvent combiné à ELK pour enrichir la corrélation des alertes.

Le tableau ci-dessous résume la comparaison des des outils de collecte et d’analyse des logs open source .

Critère	 SURICATA	 SNORT	 zeek.
Type	IDS / IPS / NSM complet	IDS/IPS basé sur signatures	Outil d’analyse de trafic et de comportements
Performance	Très élevée	Moyenne	Moyenne à complexe
Compatibilité ELK	Excellente (sortie JSON, EVE log)	Moyenne	Bonnes
Coût	Gratuit (open-source)	Gratuit (open-source)	Gratuit (open-source)
Alerting	Sortie vers SIEM (ELK, Wazuh)	Sortie texte simple	Scripts personnalisés

TABLE 2.5 – Tableau comparatif des outils de Détection d’Intrusion Open Source

Suite à la comparaison entre Suricata, Snort, et Zeek, nous avons retenu Suricata comme solution principale de détection d’intrusion.

Suricata se distingue par son moteur multi-threadé performant, son support des protocoles récents, et sa compatibilité native avec la stack ELK, permettant une corrélation automatique des alertes dans le SIEM.

De plus, sa flexibilité et ses fonctionnalités avancées en IDS/IPS en font un choix idéal pour détecter et prévenir les attaques réseau dans notre architecture.

### 2.3.5 Outils de Threat Intelligence

Pour renforcer la capacité d'analyse et de corrélation des menaces, il est essentiel d'intégrer une plateforme de Threat Intelligence (CTI). Dans cette partie, nous comparerons OpenCTI et MISP deux outils open source permettant de collecter, structurer, et partager les renseignements sur les menaces afin d'améliorer la réponse aux incidents de sécurité.

**OpenCTI (Open Cyber Threat Intelligence Platform)**

Plateforme open source développée par l'ANSSI et le CERT-EU, destinée à centraliser, analyser et partager les renseignements sur les menaces (IoC, TTP, campagnes, acteurs). Elle permet une corrélation automatisée avec d'autres outils (ELK, Suricata, TheHive) pour contextualiser les alertes de sécurité.

**MISP (Malware Information Sharing Platform)**

Plateforme open source dédiée au partage d'indicateurs de compromission (IoC) entre communautés. Elle facilite l'échange de données sur les menaces, les attaques, et les vulnérabilités. MISP est souvent utilisée en complément d'un SIEM ou d'un IDS.

graphicx array [table]xcolor float

Critère	 OPENCTI	
<b>Type</b>	Plateforme de détection d'intrusion (IDS/IPS) et moteur de corrélation des signatures.	Plateforme de Threat Intelligence et corrélation d'indicateurs (TI).
<b>Visualisation</b>	Tableaux de logs, statistiques, flux et rapports (journaux détaillés).	Vues relationnelles / graphes d'entités (entités, liens, campagnes).
<b>Intégration SIEM / IDS</b>	S'intègre naturellement avec ELK, Wazuh, et outils SIEM via output/connector.	S'intègre via API, connecteurs vers SIEM, TIPs et outils externes.
<b>Facilité d'utilisation</b>	Moyenne — interface axée logs/analyses techniques.	Relativement facile — interface dédiée à l'analyse TI et la corrélation.
<b>Coût</b>	Gratuit (open source) — certaines distributions support payant.	Gratuit (open source) — options d'hébergement ou support commerciaux possibles.

TABLE 2.6 – Tableau comparatif des outils de Threat Intelligence Open Source

Après l'évaluation d'OpenCTI et MISP, le choix s'est porté sur OpenCTI pour sa structure de données riche et interconnectée et sa capacité à centraliser et contextualiser

les menaces.

OpenCTI facilite l'analyse stratégique en reliant les indicateurs de compromission (IoC) aux campagnes, tactiques et acteurs de menace.

Son intégration native avec ELK et Suricata en fait une solution cohérente pour compléter notre SOC avec une dimension Threat Intelligence avancée.

### 2.3.6 Outils de Visualisation et Tableaux de Bord

La visualisation joue un rôle central dans toute architecture NOC/SOC, car elle permet de suivre l'état de l'infrastructure et de corrélérer les événements de sécurité en temps réel. Nous allons comparer trois solutions open source populaires : Kibana et Grafana, , afin d'identifier celle qui offre la meilleure intégration, flexibilité et ergonomie pour la supervision et la sécurité.

#### Kibana

Composant de la stack ELK, Kibana est une interface web qui permet de visualiser, explorer et corrélérer les données collectées dans Elasticsearch. Très utilisée dans les environnements SOC/NOC, elle offre des dashboards dynamiques et interactifs adaptés à la supervision et à la cybersécurité.

#### Grafana

Outil open source de visualisation de données multi-sources. Il permet de créer des tableaux de bord personnalisés à partir de nombreuses bases (Prometheus, InfluxDB, MySQL, Elasticsearch...). Grafana est particulièrement utilisé dans la supervision réseau et système (NOC). graphicx array [table]xcolor float

<b>Critère</b>	 kibana	
<b>Type</b>	Plateforme de détection d'intrusion (IDS/IPS) et moteur de corrélation des signatures.	Plateforme de Threat Intelligence et corrélation d'indicateurs (TI).
<b>Source de données principale</b>	Elasticsearch	Multi-sources (Prometheus, InfluxDB, MySQL, Elasticsearch, etc.)
<b>Coût</b>	Gratuit (open source)	Gratuit (open source)

TABLE 2.7 – Tableau comparatif des outils de Visualisation et Tableaux de Bord Open Source

Suite à la comparaison entre Kibana et Grafana, nous avons opté pour Kibana comme outil principal de visualisation. Ce choix repose sur sa compatibilité directe avec Elasticsearch, sa puissante capacité d'exploration de données, et sa souplesse dans la création de tableaux de bord interactifs. Contrairement à Grafana, Kibana offre une intégration native avec le SIEM ELK et les alertes de sécurité, permettant une corrélation centralisée entre la supervision du NOC et la détection du SOC.

### 2.3.7 Synthèse et choix

Après avoir réalisé l'ensemble des analyses comparatives, nous avons retenu les outils les plus adaptés pour la conception d'une solution NOC intégrée à un SOC open source. Les choix finaux ont été effectués en tenant compte de la performance, de la facilité d'intégration, de la compatibilité entre les composants, et de la pertinence fonctionnelle de chaque outil au regard des besoins du projet.

Le tableau ci-dessous récapitule les outils sélectionnés pour chaque fonction principale de l'architecture :

Fonctionnalité	Outil choisi	Rôle principal dans le projet
Supervision et monitoring (NOC)	<b>Zabbix</b>	Supervision de l'infrastructure, suivi des ressources (CPU, RAM, réseau), gestion des alertes et disponibilité des services.
Collecte et analyse des logs (SIEM)	<b>ELK Stack</b> (Elasticsearch, Logstash, Kibana)	Centralisation, indexation et visualisation des logs provenant des serveurs, applications et outils de sécurité.
Détection d'intrusions (IDS/IPS)	<b>Suricata</b>	Analyse du trafic réseau en temps réel, détection d'activités malveillantes et génération d'alertes de sécurité.
Threat Intelligence	<b>OpenCTI</b>	Centralisation et corrélation des indicateurs de compromission (IoC) pour enrichir l'analyse et contextualiser les menaces.
Visualisation et tableaux de bord	<b>Kibana</b>	Création de tableaux de bord dynamiques et interactifs pour la visualisation des données issues des outils de supervision et de sécurité.

TABLE 2.8 – Tableau récapitulatif des outils choisis

Ces outils, tous open source, offrent une complémentarité fonctionnelle permettant de mettre en œuvre une solution globale, intégrée et économique.

Ainsi, Zabbix assure la supervision technique (NOC), ELK centralise et corrèle les logs, Suricata détecte les intrusions, OpenCTI enrichit la connaissance sur les menaces, et Kibana offre une visualisation unifiée, constituant le point central du pilotage de la plateforme NOC/SOC.

### 2.3.8 Architecture finale

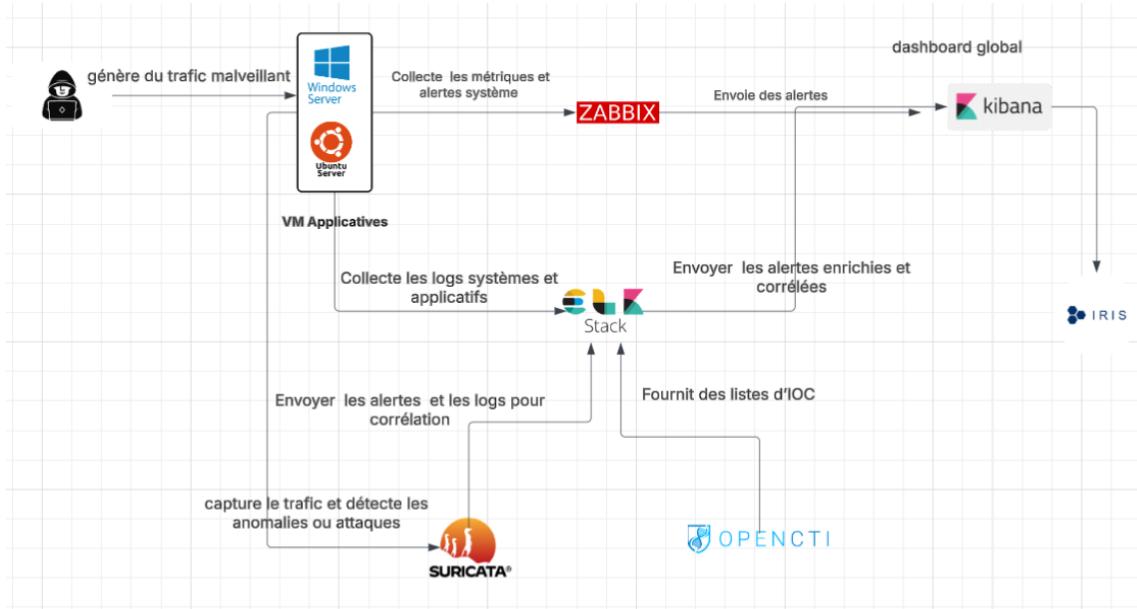


FIGURE 2.3 – Architecture finale après les choix des outils

## 2.4 Conclusion

Dans ce chapitre, nous avons présenté les concepts fondamentaux liés à la supervision et à la sécurité opérationnelle au sein d'un environnement NOC/SOC intégré. Nous avons ensuite réalisé une analyse comparative des principaux outils open source permettant d'assurer la surveillance, la détection et la corrélation des événements. Enfin, nous avons synthétisé nos choix technologiques en retenant les solutions les plus adaptées aux besoins de notre projet, afin de garantir une supervision centralisée et une analyse de sécurité efficace.

# chapitre 3

## Sprint 1 : Mise en place de la supervision

### 3.1 Introduction

Ce chapitre est consacré à la mise en place d'un NOC orienté supervision de l'infrastructure. Durant ce sprint, nous avons déployé une solution de supervision centralisée basée sur Zabbix, permettant de collecter, analyser et visualiser les métriques essentielles des serveurs Linux, Windows et des équipements réseau.

### 3.2 Sprint backlog

ID	Tâche	Priorité	Critères d'acceptation
T1.1	En tant qu'un admin, je veux créer et configurer toutes les machines virtuelles nécessaires	Élevée	Chaque VM est opérationnelle avec son système d'exploitation installé et connectée au réseau virtuel.
T1.2	Je veux configurer le réseau interne pour la communication.	Élevée	Toutes les VMs doivent pouvoir communiquer entre elles, tout en ayant une isolation externe.
T1.3	En tant qu'un admin, je veux installer et configurer un serveur Zabbix pour la supervision des agents .	Élevée	Zabbix doit détecter automatiquement les hôtes, collecter les métriques et générer des alertes
T1.4	En tant qu'un admin, je veux déployer les agents zabbix sur toutes les Vms	Moyenne	Suricata doit générer des alertes lors de comportements suspects détectés.
T1.4	En tant qu'un administrateur, je veux configurer une tableau de bord de supervision	Moyenne	Les indicateurs critiques (CPU, mémoire) sont visualisables en temps réel.

TABLE 3.9 – sprint backlog 1

## 3.3 Environnement de travail

### 3.3.1 Environnement matériel :

Le tableau représente l'ensemble des ressources utilisées ainsi que leurs caractéristiques.

Machine virtuelle	OS	RAM	vCPU
Zabbix server	Ubuntu 22.04	4 Go	2
Ubuntu Server	Ubuntu Server 22.04	4 Go	2
Windows Server	Windows Server 2019	4 Go	2

TABLE 3.10 – caractéristiques matérielles et logicielles

### 3.3.2 Environnement logiciel :

Pour la mise en œuvre de notre solution, nous avons choisi d'utiliser l'hyperviseur VMware. Cet outil nous a permis de créer, configurer et gérer plusieurs machines virtuelles.

#### Définition de VMware

VMware est une plateforme de virtualisation qui permet de créer et de gérer des machines virtuelles (VM) sur un même hôte physique. Elle offre un environnement isolé pour chaque VM, simulant des ordinateurs complets avec leur propre système d'exploitation.



FIGURE 3.4 – Logo VMware

## 3.4 Mise en place de Zabbix

### 3.4.1 Installation zabbix Server

Dans cette partie, nous avons procédé à la mise en place de la plateforme Zabbix afin d'assurer la supervision centralisée de l'ensemble de nos machines virtuelles et services déployés dans l'environnement du projet.

L'installation de Zabbix a été effectuée à partir des dépôts officiels, garantissant ainsi l'utilisation d'une version stable, sécurisée et pleinement compatible avec notre infrastructure. Une fois le serveur configuré et les agents déployés, nous avons pu accéder à l'interface Web de Zabbix, qui constitue le centre de monitoring et de gestion des alertes.

La figure ci-dessous présente le tableau de bord principal de Zabbix après l'installation,

mettant en évidence l'interface intuitive permettant le suivi en temps réel de l'état des hôtes, des performances système et des événements critiques détectés au sein de l'infrastructure.

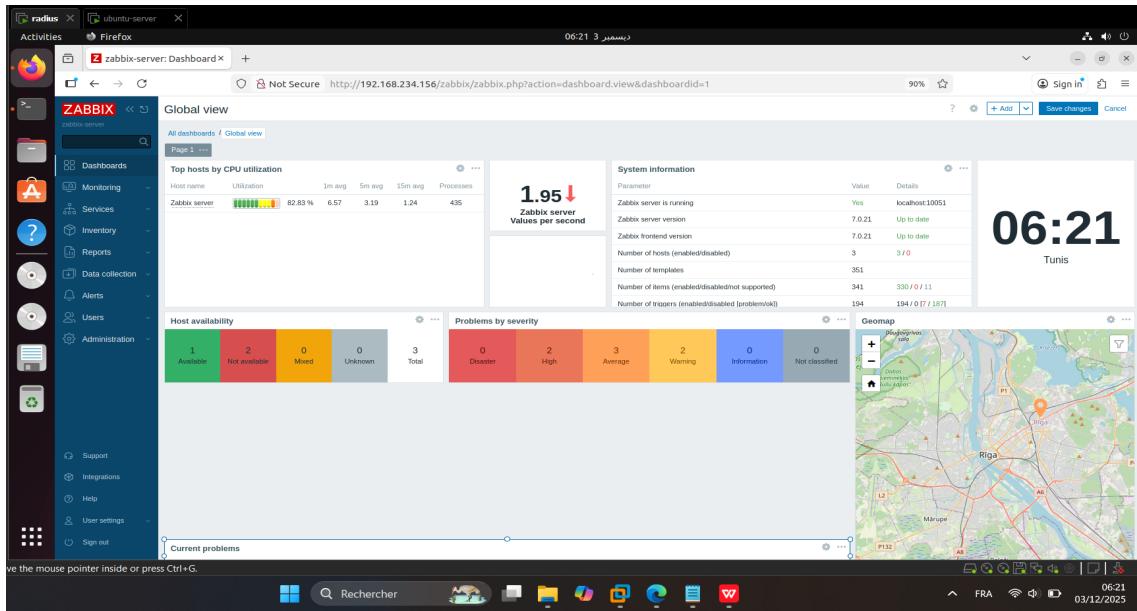


FIGURE 3.5 – Dashboard zabbix

### 3.4.2 Ajout des agents

Une fois le serveur Zabbix installé, nous avons procédé à l'installation de l'agent Zabbix sur la machine virtuelle Ubuntu Server et Windows Server destinée à être supervisée.

#### Pour la machine Ubuntu server :

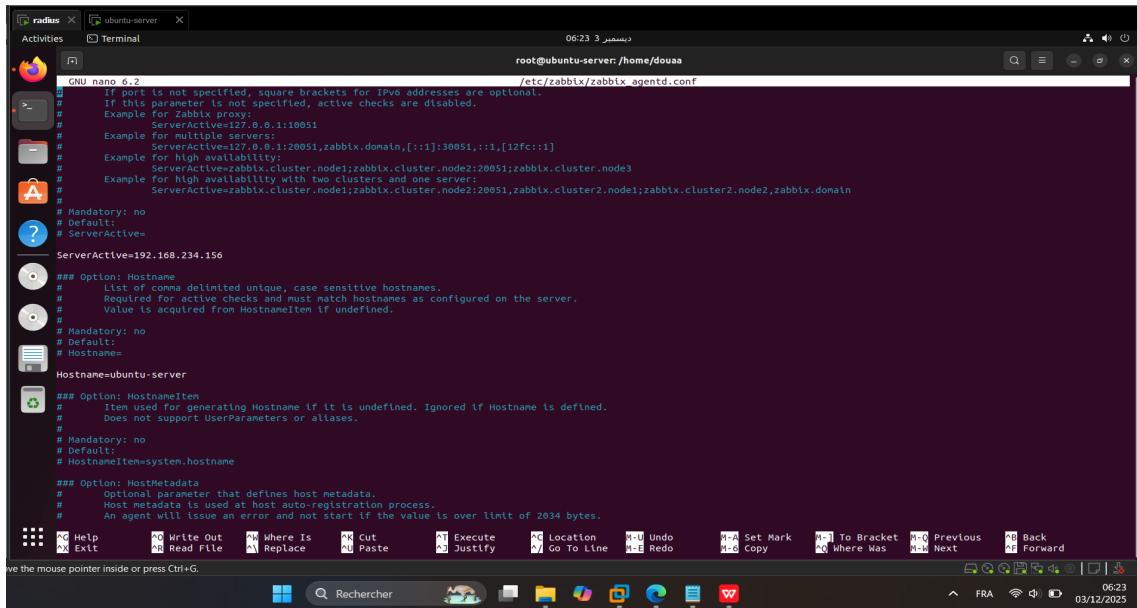
Sur la machine Ubuntu Server, l'agent Zabbix a été installé à partir du dépôt officiel fourni par l'éditeur, garantissant ainsi une version stable et compatible avec notre serveur Zabbix. Après l'installation, le fichier de configuration de l'agent a été ajusté pour spécifier l'adresse IP du serveur Zabbix, permettant ainsi une communication fiable entre les deux composants. Une fois le service démarré et activé, l'agent a été automatiquement détecté depuis l'interface Zabbix et ajouté comme nouvel hôte supervisé. Dans ce fichier, il est important de s'assurer que les paramètres suivants sont correctement définis :

- ServerActive=<adresse-IP-serveur-Zabbix> : Remplacer <adresse-IP-serveur-Zabbix> par l'adresse IP du serveur Zabbix.

Hostname=<nom-hôte> : Remplacer <nom-hôte> par le nom de la machine que nous voulons surveiller.

#### Pour la machine Windows Server

Pour la machine Windows Server, nous avons utilisé l'agent Zabbix dédié à l'environnement Windows, disponible sous forme d'exécutable. L'installation a été réalisée via



```

GNU nano 6.2
# If port is not specified, square brackets for IPv6 addresses are optional.
# If this parameter is not specified, active checks are disabled.
# Example for Zabbix proxy:
#   ServerActive=[127.0.0.1]:10051
#   Example for multiple servers:
#     ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,:1,[12fc::1]
#   Example for high availability with two clusters and one server:
#     ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
#
# Mandatory: no
# Default:
# ServerActive=
ServerActive=192.168.234.156

## Option: Hostname
# List of comma delimited unique, case sensitive hostnames.
# Required for active checks and must match hostnames as configured on the server.
# Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostnames=
Hostname=ubuntu-server

## Option: HostnameItem
# Item used for generating Hostname if it is undefined. Ignored if Hostname is defined.
# Does not support UserParameters or aliases.
#
# Mandatory: no
# Default:
# HostnameItem=system.hostname

## Option: HostMetadata
# Optional parameter that defines host metadata.
# Host metadata is used at host auto-registration process.
# An agent will issue an error and not start if the value is over limit of 2034 bytes.

```

FIGURE 3.6 – fichier de configuration Zabbix-agent

l'assistant graphique, suivi de la configuration des paramètres de connexion au serveur Zabbix (adresse IP du serveur, nom de l'hôte, clé de connexion si nécessaire).

Une fois l'agent installé et son service lancé, la machine Windows a été enregistrée dans Zabbix, permettant la collecte automatique des métriques système propres à cet environnement (processus, services Windows, état réseau, logs, etc.).

Une fois que nous avons installé et configuré les agents sur nos machines Ubuntu et Windows Server, nous pouvons procéder à l'ajout de ces hôtes dans Zabbix. Cela nous permettra de surveiller l'état et les performances de ces machines ainsi que les services qui y sont déployés.

Dans cette partie, nous allons ajouter manuellement les nouveaux hôtes (Ubuntu et Windows Server), sur lesquels les agents ont été installés, afin de les intégrer dans Zabbix pour la surveillance de leur état et de leurs services.

Nous avons accédé à la page principale de Zabbix, puis cliqué sur "Data collection", suivi de "Hosts", et ensuite sur "Create host". Voici les étapes que nous avons suivies pour configurer l'hôte :

**Host name** : Nous avons entré le nom d'hôte de la machine à surveiller, correspondant au « hostname » identifié précédemment.

**Templates** : Nous avons sélectionné les modèles "Zabbix Agent" afin de surveiller les hôtes correctement.

**Host group** : Nous avons attribué l'hôte au groupe approprié.

**Interface** : Nous avons cliqué sur "Ajouter", sélectionné "Agent", puis saisi l'adresse IP de l'agent installé.

La figure illustre le processus d'ajout de l'hôte Ubuntu Server dans Zabbix :

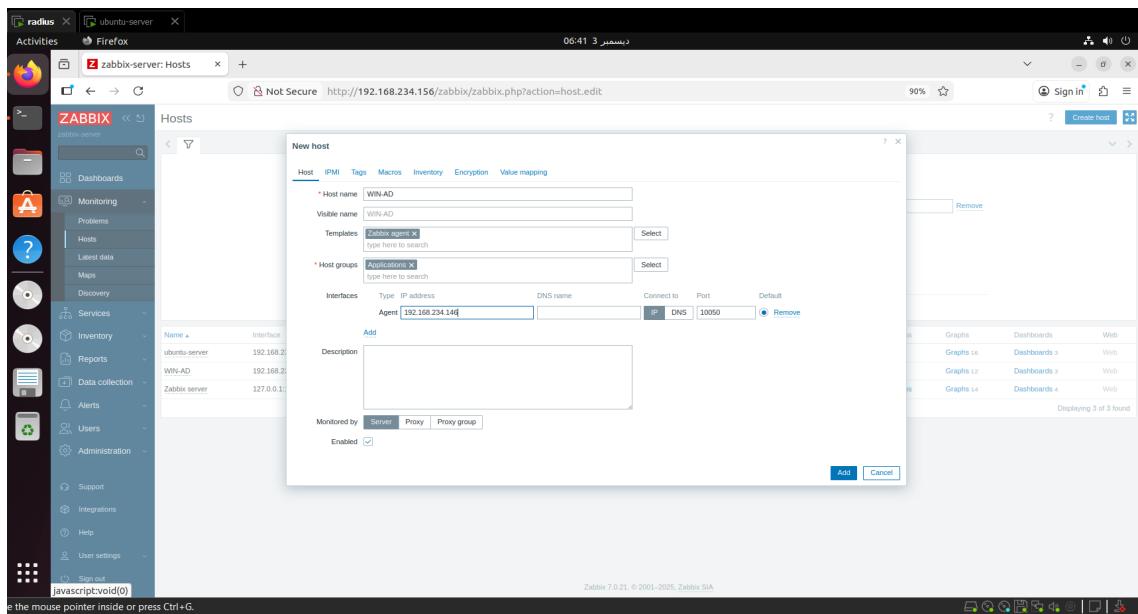


FIGURE 3.7 – le processus d’ajout de l’hôte Ubuntu server dans Zabbix

La figure illustre le processus d’ajout de l’hôte Windows Server dans Zabbix : Une fois

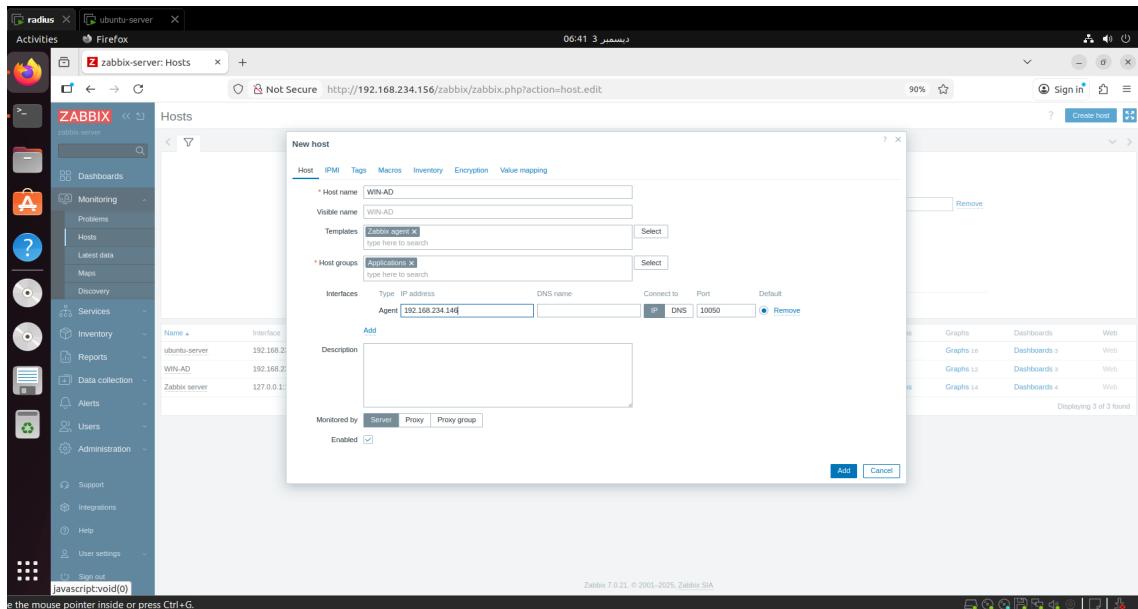


FIGURE 3.8 – le processus d’ajout de l’hôte windows server dans Zabbix

les deux agents ajoutés avec succès, nous avons accédé au tableau de bord de zabbix afin de vérifier son enregistrement et sa bonne communication avec le serveur zabbix. La figure ci-dessous illustre l’interface graphique de zabbix après l’ajout de les agents, confirmant que celui-ci est bien détecté et opérationnel au sein de l’infrastructure supervisée.

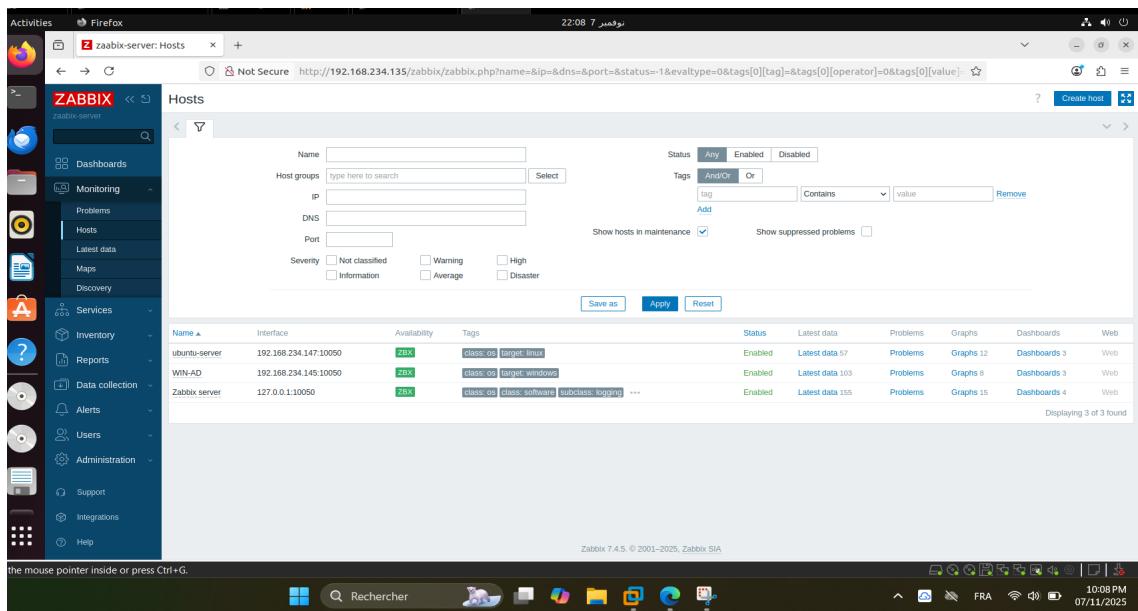


FIGURE 3.9 – l'état des hôtes sur zabbix

### 3.4.3 Supervision des hôtes dans Zabbix

#### Supervision du machine ubuntu server

Après avoir ajouté les hôtes dans Zabbix, nous pouvons commencer leur supervision active. Cela se fait principalement par la création d'items et de déclencheurs.

Les items (ou éléments) dans Zabbix nous permettent de définir les paramètres spécifiques que nous souhaitons surveiller pour chaque hôte. Cela inclut, par exemple, l'utilisation du processeur, la consommation de mémoire, et l'état des services critiques. Dans le cadre de notre projet, nous allons surveiller des paramètres essentiels tels que l'utilisation du processeur, la consommation de mémoire, ainsi que l'état de services importants comme SSH et tomcat. Ces éléments nous aideront à avoir une vue d'ensemble sur la performance et la disponibilité des machines supervisées.

Ces items serviront ensuite à la création de déclencheurs, qui joueront un rôle fondamental dans la supervision en temps réel. Un déclencheur dans Zabbix est une expression logique qui compare les données collectées par les items à des seuils que nous avons définis. Si ces seuils sont dépassés, des alertes et des notifications sont générées, nous avertisant ainsi de tout incident ou déviation. Cette approche garantit une réactivité accrue face aux problèmes d'infrastructure.

#### Supervision des ressources du système

La surveillance des ressources système est essentielle pour garantir le bon fonctionnement de notre infrastructure. Cela permet de détecter rapidement les anomalies, d'optimiser les performances, et de prévenir les pannes avant qu'elles n'affectent la productivité. En surveillant activement les ressources telles que le processeur, la mémoire, et les disques, nous pouvons identifier les goulets d'étranglement et ajuster les configurations en conséquence.

Création d'un template :

Un template dans Zabbix est un outil central qui simplifie la gestion des éléments de

surveillance (items), des déclencheurs (triggers), des graphiques, et d'autres paramètres. Il nous permet d'appliquer la même configuration à plusieurs hôtes sans avoir à la recréer pour chacun individuellement, ce qui nous fait gagner du temps et assure une uniformité dans la supervision. De plus, si une modification est nécessaire, elle est répliquée automatiquement à tous les hôtes utilisant le template, ce qui simplifie grandement la maintenance.

#### Ajout du template à l'hôte

L'ajout du template `GestionDesRessources` aux hôtes nous permet d'appliquer uniformément la même configuration de supervision, notamment pour surveiller la consommation CPU, sur plusieurs hôtes. Cela garantit une surveillance cohérente des ressources sur tous nos serveurs ou systèmes sans nécessiter une configuration manuelle pour chaque hôte. Pour ajouter le template au hôte Ubuntu server, nous avons accédé à Data Collection, puis à Hosts. Ensuite, nous avons sélectionné les hôtes concernés et, dans le champ Templates, nous avons ajouté le template `GestionDesRessources`.

#### Supervision de l'utilisation du CPU

##### Ajout d'un élément (Item) pour surveiller l'utilisation CPU

Après avoir créé notre template, nous avons ajouté un élément (item) pour surveiller l'utilisation du processeur (CPU) au sein de ce template. Voici les étapes que nous avons suivies :

- Nous avons sélectionné le template créé précédemment.
- Ensuite, nous avons cliqué sur "Create item" pour créer un nouvel élément de surveillance. Pour la configuration de cet élément :
  - Nom de l'élément : Nous avons saisi "consommationCPU".
  - Type : Nous avons choisi "Zabbix agent".
  - Clé (key) : La clé utilisée est `system.cpu.util[all,system]`. Cette clé permet à Zabbix de collecter l'utilisation totale du CPU par le système. En particulier, elle mesure la consommation CPU globale allouée au système.
  - Type de données : Nous avons sélectionné "Numeric (float)" pour indiquer que l'élément retourne des valeurs numériques simples.
  - Intervalle de mise à jour : 10 secondes, ce qui permet de recueillir des données en quasi temps réel sur l'utilisation du CPU.

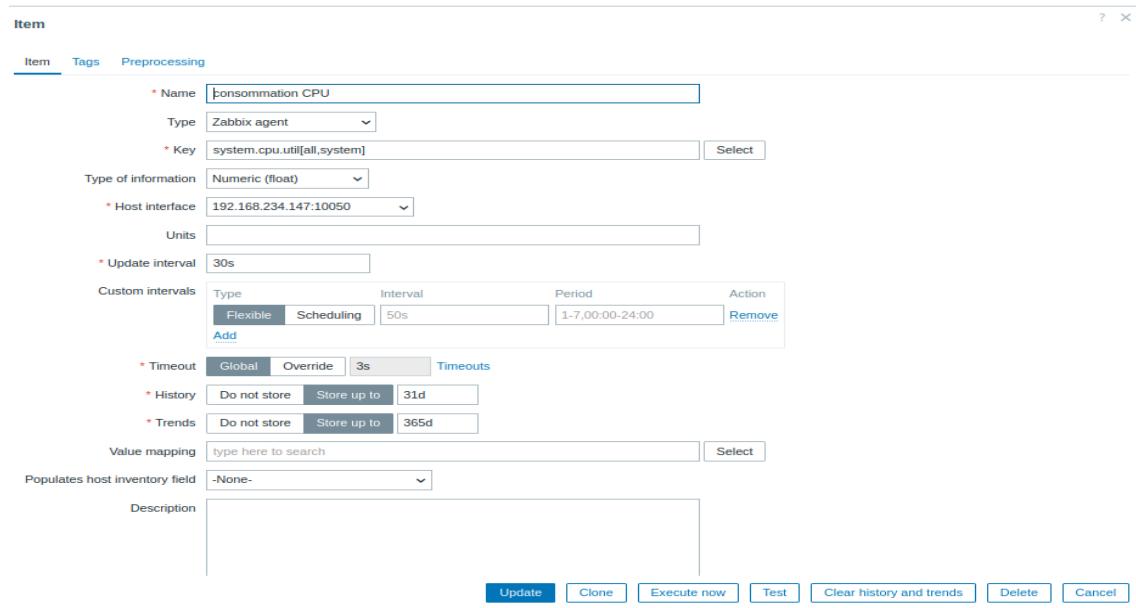


FIGURE 3.10 – Ajout d'un élément pour surveiller l'utilisation CPU

Création d'un déclencheur (Trigger) pour alerter sur une utilisation élevée du CPU. Après avoir créé notre item pour surveiller l'utilisation du CPU, nous avons mis en place un déclencheur qui nous alertera lorsque l'utilisation du CPU dépasse un seuil critique. Ce déclencheur est configuré pour surveiller à la fois la machine Ubuntu server, assurant ainsi une surveillance proactive de la charge CPU sur ces systèmes. Nous avons accédé à l'onglet "Triggers" (Déclencheurs) sous l'hôte concerné (Ubuntu Server ou Windows Server). Ensuite, nous avons cliqué sur "Create trigger" pour créer un nouveau déclencheur.

Lors de la configuration du déclencheur :

- Le nom saisi est "High CPU Usage" pour identifier clairement ce déclencheur. L'expression utilisée est `last(/GestionDesRessources/system.cpu.util[all,system])>70`. Cela signifie que si l'utilisation du CPU dépasse 70, une alerte sera déclenchée. Pour créer cette expression, nous avons : Sélectionné l'item "consommation CPU" précédemment créé puis choisi la fonction Last of (T), qui vérifie la dernière valeur reçue pour cet item et enfin nous avons définir la condition `>70` pour indiquer que l'alerte sera lancée lorsque l'utilisation du CPU dépasse 70.

- Nous avons défini la严重性 du déclencheur à "Warning", pour indiquer qu'une utilisation élevée du CPU nécessite une attention particulière.

La figure ci-dessous montre la configuration complète du déclencheur. Ce dernier est crucial pour nous permettre de recevoir une alerte dès que l'utilisation CPU atteint un seuil critique, ce qui nous permet d'agir rapidement et d'éviter des problèmes potentiels liés à une surcharge du processeur.

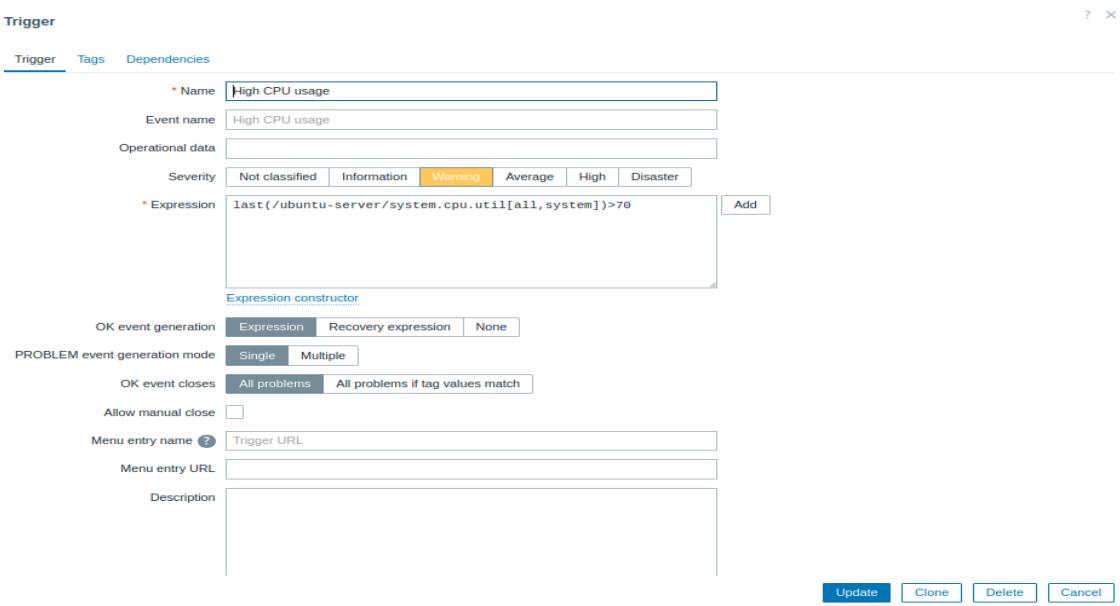


FIGURE 3.11 – Création d'un déclencheur (Trigger) pour surveiller l'utilisation CPU

Après avoir configuré l'élément et le déclencheur, nous avons vérifié leur bon fonctionnement en surveillant les valeurs collectées. Un test de charge a été réalisé à l'aide d'outils comme stress afin de simuler une utilisation intensive du CPU. Cela nous a permis de confirmer que le déclencheur s'active correctement lorsque l'utilisation du CPU dépasse le seuil de 70 que nous avons défini.

### Supervision de la mémoire RAM

La supervision de l'utilisation de la mémoire RAM est tout aussi essentielle que celle du CPU pour garantir une performance optimale des systèmes. La mémoire RAM joue un rôle clé dans l'exécution des programmes et des processus, et une consommation excessive peut entraîner des ralentissements, des blocages ou même des pannes système. En surveillant régulièrement l'utilisation de la RAM, nous pouvons détecter les pics de consommation, identifier les applications ou services qui utilisent de grandes quantités de mémoire, et ajuster les ressources ou configurations en conséquence. Cela nous aide à anticiper les besoins en mémoire, à optimiser l'allocation des ressources, et à garantir une meilleure stabilité du système, surtout lorsque plusieurs applications critiques fonctionnent en parallèle.

Ajout d'un élément (Item) pour l'utilisation de la mémoire RAM :

Pour la configuration de cet item de surveillance de la mémoire RAM, nous avons suivi les mêmes étapes que pour l'item de surveillance du CPU. Nous avons sélectionné le template GestionDesRessources. Ensuite, nous avons cliqué sur "Create item" pour créer un nouvel élément de surveillance. Nous avons configuré les paramètres tels que le nom de l'élément, le type d'agent (Zabbix agent) et l'intervalle de mise à jour à 30 secondes pour permettre un suivi fréquent et précis de l'utilisation de la RAM.

La Figure ci-dessous présente la configuration de l'élément de surveillance, illustrant en détail les paramètres que nous avons définis pour suivre l'utilisation des ressources du

système. Création du déclencheur pour alerter en cas de forte utilisation de la mémoire

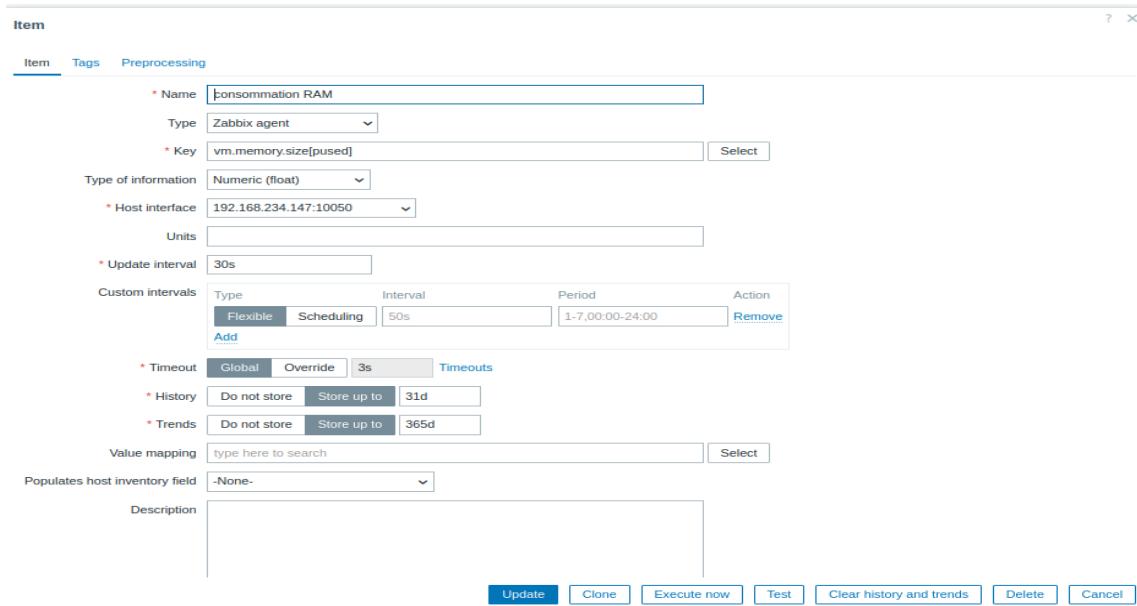


FIGURE 3.12 – Ajout d'un élément pour l'utilisation de la mémoire RAM

RAM

La création du déclencheur pour alerter en cas de forte utilisation de la mémoire RAM suit des étapes similaires à celles effectuées pour le CPU. Lors de la configuration du déclencheur :

- \* Nous avons saisi "AlerteRAM" comme nom du déclencheur , afin d'identifier clairement la situation de surveillance.
- \* Nous avons utilisé l'expression <Host Name> :vm.memory.size[pused].last()>75. Cela signifie que si l'utilisation de la mémoire RAM dépasse 75, une alerte sera déclenchée.
- \* Nous avons défini la Sévérité à "Warning" pour que cette alerte soit classée parmi les avertissements.

La figure ci-dessous illustre la configuration du déclencheur : Après avoir configuré l'élément et le déclencheur, nous avons vérifié leur bon fonctionnement en surveillant les valeurs collectées. Des tests de charge ont été réalisés à l'aide d'outils comme stress pour simuler une utilisation élevée de la mémoire RAM et observer si le déclencheur fonctionnait correctement.

La figure ci-dessous illustre le résultat de l'alerte que nous avons créée. Ce déclencheur, conçu pour surveiller l'utilisation excessive de la mémoire RAM, permet de générer une alerte dès que l'utilisation dépasse un seuil critique de 75 .

## Supervision du service tomcat

Pour superviser tomcat avec Zabbix, plusieurs méthodes sont disponibles, selon les aspects de Tomcat que nous souhaitons surveiller. Dans notre projet, nous avons opté pour la supervision du processus Tomcat sur notre machine Ubuntu server. Le processus Tomcat doit être surveillé en continu pour garantir qu'il est en cours d'exécution. Cela permet de s'assurer que le service web est toujours opérationnel et qu'aucune panne n'est passée inaperçue. En cas de dysfonctionnement, Zabbix déclenchera des alertes, facilitant une intervention rapide pour résoudre les problèmes.

The screenshot shows the 'Trigger' configuration page in Zabbix. The 'Trigger' tab is selected. The configuration includes:

- Name:** Alerte RAM
- Event name:** Alerte RAM
- Operational data:** (empty)
- Severity:** Not classified, Information, Warning, Average, High, Disaster
- Expression:** last(/ubuntu-server/vm.memory.size[pused])>75
- OK event generation:** Expression, Recovery expression, None
- PROBLEM event generation mode:** Single, Multiple
- OK event closes:** All problems, All problems if tag values match
- Allow manual close:** (checkbox)
- Menu entry name:** Trigger URL
- Menu entry URL:** (empty)
- Description:** (empty)

At the bottom right are buttons for **Update**, **Clone**, **Delete**, and **Cancel**.

FIGURE 3.13 – Création du déclencheur pour l'utilisation de la mémoire RAM

FIGURE 3.14 –

· Crédation d'un élément (item) pour surveiller le processus Tomcat :  
Comme mentionné précédemment, pour pouvoir surveiller le service Apache, nous devons créer un élément spécifique dans Zabbix. Pour ce faire, nous avons suivi les étapes suivantes :

Nous sommes allés dans Data Collection > Hosts, puis sélectionné l'hôte où Apache est installé (notre machine Ubuntu server). Ensuite, nous avons cliqué sur Items (Éléments) dans le menu de l'hôte et sur Create item pour créer un nouvel élément.

Pour la configuration de l'élément :

- Le nom saisi pour l'élément est tomcat process status.
- Pour la clé (key), nous avons utilisé proc.num[tomcat9]. Cette clé permet à Zabbix de vérifier le nombre de processus tomcat actifs sur le système. Elle est essentielle pour surveiller si le service tomcat est en cours d'exécution ou s'il s'est arrêté.
- Le type de données sélectionné est Numeric (unsigned), qui correspond à un type de données renvoyant des valeurs numériques simples.
- Nous avons défini l'intervalle de mise à jour à 60 secondes, garantissant ainsi une mise à jour régulière des données.

.La figure ci-dessous montre la configuration de l'élément :

- Crédation d'un déclencheur pour alerter si tomcat s'arrête :

Après avoir créé notre item, nous avons procédé à la création d'un déclencheur pour nous alerter si le service Apache s'arrête. Voici les étapes que nous avons suivies :

Nous avons accédé à l'onglet "Triggers" (Déclencheurs) sous l'hôte concerné. Ensuite, nous avons cliqué sur "Create trigger" pour créer un nouveau déclencheur.

Lors de la configuration du déclencheur :

- Le nom saisi est Tomcat down, afin d'identifier clairement la situation à surveiller.
- Pour l'expression, nous avons utilisé <Host Name> :proc.num[tomcat9].last()=0 . Cette expression signifie que si le nombre de processus tomcat(proc.num[httpd]) est égal à 0 (service arrêté), une alerte sera déclenchée. Pour créer cette expression, nous avons cliqué sur

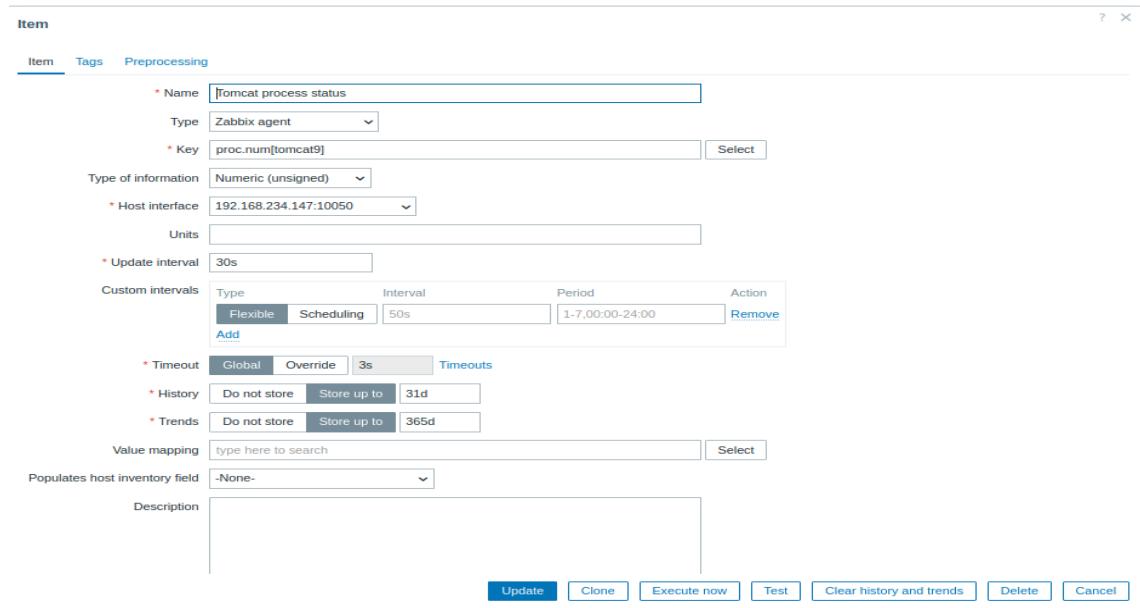


FIGURE 3.15 – Crédation d'un élément pour surveiller le processus tomcat

"Add", puis rempli la condition. Dans Item, nous avons sélectionné l'élément que nous avons créé précédemment, puis dans Function, nous avons choisi Last of (T), et enfin mis le résultat = 0 pour spécifier que l'alerte sera lancée si Apache est arrêté. Comme indiqué dans la figure suivante

- Enfin, nous avons défini la Sévérité à High, pour que l'alerte soit classée dans les avertissements.

La figure ci-dessous montre la configuration du déclencheur :

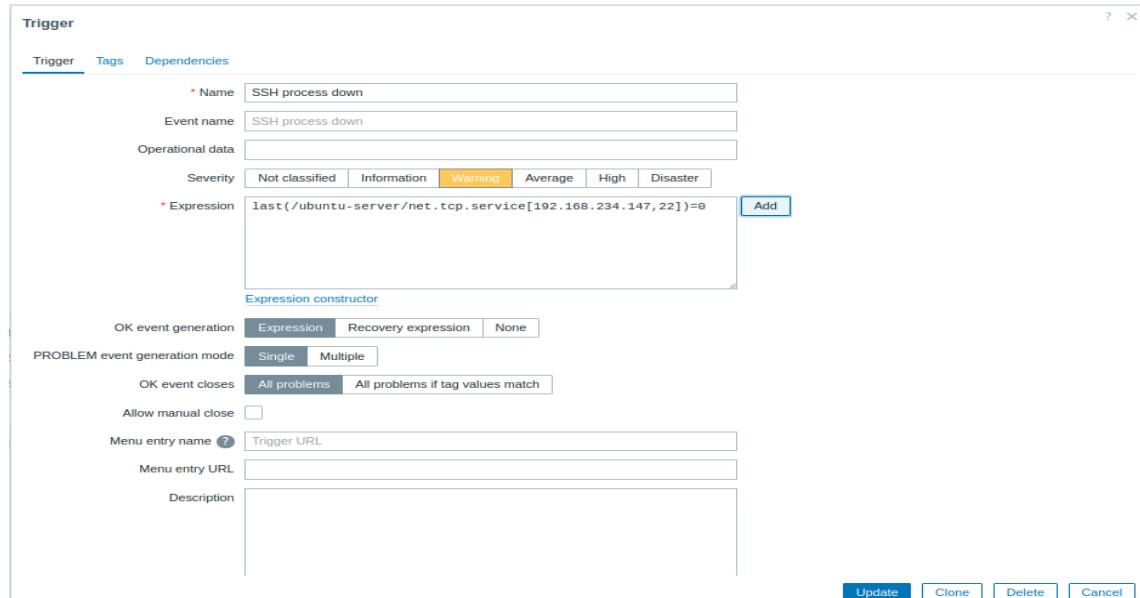


FIGURE 3.16 – Crédation d'un élément pour surveiller le processus tomcat

Une fois que nous avons terminé la configuration du déclencheur, nous avons cliqué sur "Add" pour valider et enregistrer notre déclencheur dans Zabbix.

La figure ci-dessous illustre le résultat de l'alerte que nous avons créée. Ce déclencheur,

mis en place pour surveiller l'état du service tomcat, a généré une alerte lorsque le processus tomcat s'est arrêté, conformément aux conditions définies dans notre déclencheur. Cela nous permet de réagir rapidement en cas de dysfonctionnement du service sur notre serveur Ubuntu.

Severity	Value	Name	Operational data	Expression	Status	Info	Tags
High	PROBLEM	Tomcat down		last(/ubuntu-server/proc.num@tomcat5)=0	Enabled		

FIGURE 3.17 – Résultat de l'alerte tomcat is not running on server ubuntu

## Supervision du service SSH

Pour surveiller le service SSH avec Zabbix, nous avons suivi une méthode qui combine la surveillance de la disponibilité du service SSH et la vérification des processus associés à SSH.

### Surveillance de la Disponibilité du Service SSH

- Création d'un Item pour vérifier la disponibilité du port SSH

Nous avons suivi les étapes suivantes pour créer un élément de surveillance du service SSH :

Nous sommes allés dans Data Collection > Hosts et avons sélectionné l'hôte où nous voulons surveiller le service SSH (notre machine Ubuntu server). Ensuite, Nous avons cliqué sur Items (Éléments) dans le menu de l'hôte, puis sur Create item pour créer un nouvel élément.

Pour la configuration de l'élément de surveillance du service SSH, nous avons mis en place les paramètres suivants :

- Nom : Nous avons choisi "ssh service availability" pour identifier clairement l'élément de surveillance.
- Clé (Key) : net.tcp.service[ssh,<ip-address>,22] (où <ip-address> est remplacée par l'adresse IP du serveur supervisé).
- Type d'information : Nous avons opté pour "Numeric (unsigned)".
- Intervalle de mise à jour : Nous avons fixé cet intervalle à 30 secondes (ou selon nos besoins).

La figure suivante présente la configuration de l'élément.

· Création d'un Déclencheur pour alerter en cas d'indisponibilité du service SSH :  
Après avoir créé notre item, nous avons configuré un déclencheur pour recevoir une alerte si le service SSH devient indisponible. Nous sommes allés dans l'onglet Triggers (Déclencheurs) sous l'hôte concerné. Ensuite, Nous avons cliqué sur Create trigger pour créer un nouveau déclencheur.

Pour la création du déclencheur visant à surveiller l'état du service SSH, nous avons configuré les paramètres suivants :

- Nom : Nous avons utilisé "SSH processDown" pour identifier le déclencheur.
- Expression : <Host Name> :net.tcp.service[ssh,<ip-address>,22].last()=0  
Cette expression signifie que si la dernière valeur renvoyée pour le service SSH est égale à 0, cela indique que le service SSH n'est pas disponible, et une alerte sera déclenchée.
- Gravité : Nous avons défini la gravité à "Warnings" (ou selon nos besoins).

Ce déclencheur est essentiel pour recevoir une alerte dès que le service SSH devient inaccessible, ce qui permet d'intervenir rapidement afin de résoudre le problème.

La figure ci-dessous présente la configuration du déclencheur :

The screenshot shows the 'Item' configuration page in Zabbix. The 'Name' is set to 'SSH process count', 'Type' is 'Zabbix agent', and 'Key' is 'proc.num[sshd]'. The 'Type of information' is 'Numeric (unsigned)'. The 'Host interface' is '192.168.234.147:10050'. The 'Update interval' is '50s'. Under 'Custom intervals', there is a single entry for 'Scheduling' with an interval of '50s' and a period from '1-7,00-00-24:00'. The 'Timeout' is set to '3s'. For 'History', 'Do not store' is selected with a store up to of '31d'. For 'Trends', 'Do not store' is selected with a store up to of '365d'. The 'Value mapping' section has a dropdown for 'Populates host inventory field' set to 'None'. The 'Description' field is empty. At the bottom are buttons for 'Update', 'Clone', 'Execute now', 'Test', 'Clear history and trends', 'Delete', and 'Cancel'.

FIGURE 3.18 – Crédation d'un Item pour vérifier la disponibilité du port SSH

The screenshot shows the 'Trigger' configuration page in Zabbix. The 'Name' is 'SSH process down', 'Event name' is 'SSH process down', and 'Operational data' is empty. The 'Severity' is 'Warning'. The 'Expression' is 'last(/ubuntu-server/proc.num[sshd])<1'. Below this is the 'Expression constructor' section. The 'OK event generation' is set to 'None'. The 'PROBLEM event generation mode' is 'Single'. The 'OK event closes' is 'All problems'. The 'Allow manual close' checkbox is unchecked. The 'Menu entry name' is 'Trigger URL', and the 'Menu entry URL' is empty. The 'Description' field is empty. At the bottom are buttons for 'Update', 'Clone', 'Delete', and 'Cancel'.

FIGURE 3.19 – Crédation d'un Déclencheur pour alerter en cas d'indisponibilité du service SSH

### Surveillance du Processus SSH

Pour surveiller le processus SSH sur l'hôte, nous avons suivi une approche similaire à celle utilisée pour la surveillance de la disponibilité du service SSH

- Création d'un Item pour le Processus SSH

Pour la configuration de l'élément de surveillance du processus SSH, nous avons mis en place les paramètres suivants :

- Nom : Nous avons nommé cet élément SSH process count pour indiquer clairement qu'il surveille le nombre de processus SSH actifs.

- Clé (Key) : Nous avons utilisé la clé proc.num[sshd], qui permet de compter le nombre de processus SSH en cours d'exécution sur l'hôte supervisé.
  - Type d'information : Nous avons choisi Numeric (unsigned), car l'élément retourne des valeurs numériques simples.
  - Intervalle de mise à jour : Nous avons défini l'intervalle à 50 secondes pour recueillir des données régulièrement et suivre l'état du processus SSH en temps réel.
- La figure ci-dessous présente la configuration de l'item :

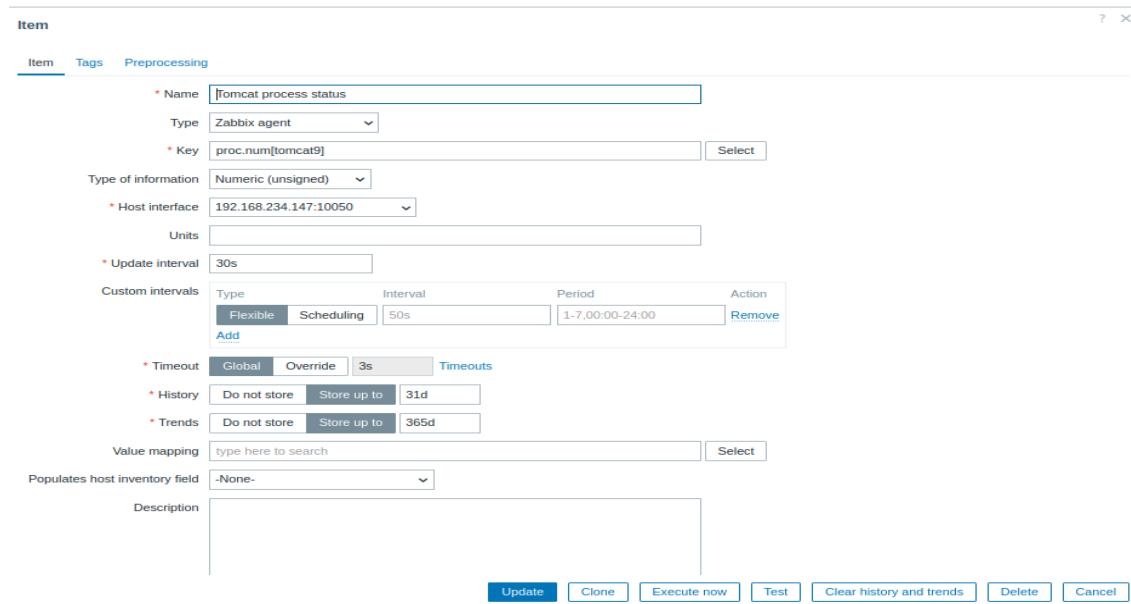


FIGURE 3.20 – Création d'un élément pour surveiller le processus SSH

- Crédit d'un Déclencheur pour le Processus SSH :

Pour créer le déclencheur qui surveillera l'état du service SSH, nous avons configuré les paramètres suivants :

- Nom du déclencheur : SSH process down
  - Expression : your-host :proc.num[sshd].last()<1 Cette expression vérifie si le nombre de processus SSH est inférieur à 1. Si aucun processus SSH n'est détecté, une alerte est générée.
  - Gravité : Warnings(ou selon nos besoins)
- Ce déclencheur nous permet d'être immédiatement alertés en cas d'arrêt du processus SSH, assurant une surveillance proactive et une intervention rapide en cas de problème. L'illustration ci-dessous montre la configuration du déclencheur :

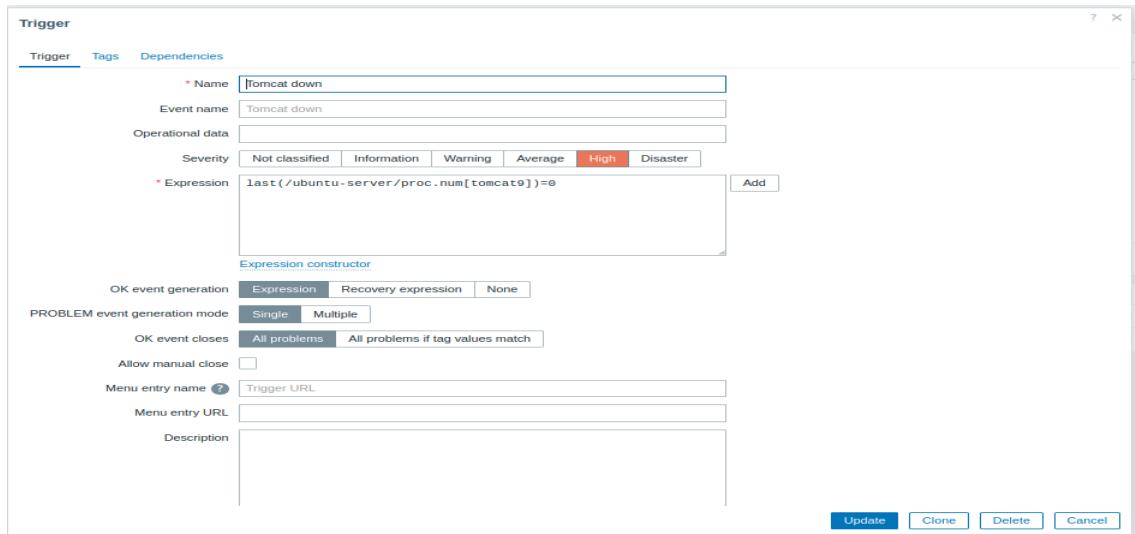


FIGURE 3.21 – Création d'un déclencheur pour surveiller le processus SSH

Pour tester les déclencheurs que nous avons configurés pour surveiller le service SSH, nous pouvons arrêter le service SSH sur l'hôte que nous surveillons. Nous simulons une situation où le service n'est pas disponible. Zabbix devrait détecter cette situation et déclencher les alertes associées aux déclencheurs que nous avons configurés, comme le montre la figure suivante :

Severity	Value	Name	Operational data	Expression	Status	Info	Tags
High	PROBLEM	Tomcat down		last(/ubuntu-server/proc.num[tomcat9])=0	Enabled		

FIGURE 3.22 –

## Pour la machine Windows Server

Le même principe de supervision des ressources a été appliqué à la machine Windows Server, afin de garantir une surveillance.

La supervision du CPU permet de suivre la charge globale du système, le pourcentage d'utilisation processeur ainsi que la charge par cœur. Concernant la mémoire RAM, Zabbix surveille la mémoire totale, la mémoire utilisée, la mémoire libre ainsi que le taux d'occupation.

## Collecte des métriques

Zabbix permet de collecter un ensemble détaillé de métriques système, offrant une vision de l'état de performance et de disponibilité de la machine. Ces métriques sont accessibles via la section Latest Data et mises à jour en temps réel.

Les métriques CPU incluent notamment l'utilisation globale du processeur, le temps processeur en mode utilisateur (CPU user time) et en mode privilégié, ainsi que la longueur de la file d'attente CPU. Concernant la mémoire, Zabbix surveille la consommation de la RAM , l'espace de swap disponible ainsi que le pourcentage de swap libre.

La supervision du stockage est assurée à travers plusieurs indicateurs tels que le taux d'utilisation des disques , les débits de lecture et d'écriture , ainsi que les temps d'attente

moyens des requêtes disque (Disk request avg waiting time).

L'illustration ci-dessous montre les métriques de l'hôte

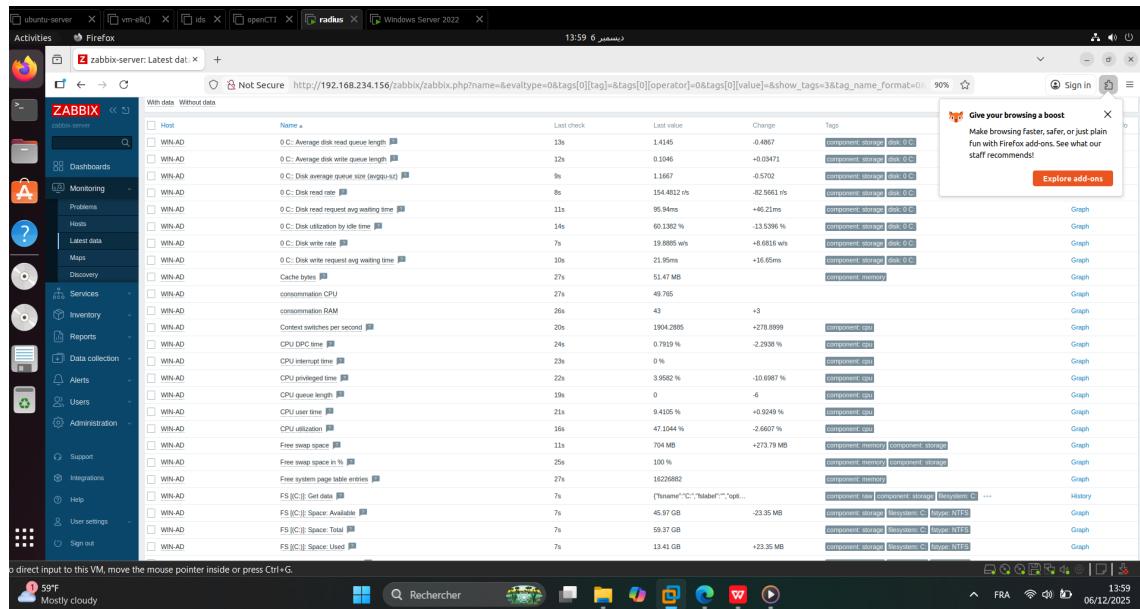


FIGURE 3.23 – Collecte des métriques de l'hôte

Nous avons mis en place un contrôle spécifique des services critiques de l'Active Directory à l'aide de Zabbix. Grâce à l'agent Zabbix installé sur le serveur Windows et au template dédié, Zabbix assure une surveillance en temps réel de l'état des services essentiels tels que DNS, DHCP, Kerberos (KDC), Netlogon et DFS Replication.

Chaque service est vérifié périodiquement afin de détecter tout arrêt ou dysfonctionnement. En cas de changement d'état, notamment si un service passe en état Stopped, une alerte est automatiquement générée afin de permettre une intervention rapide de l'équipe. Cette supervision garantit la disponibilité et la continuité des services Active Directory. La figure ci-dessous montre les métriques de l'hôte

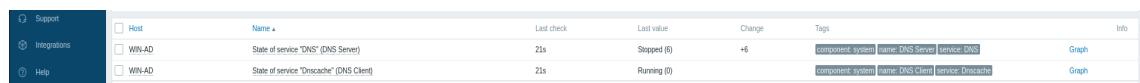


FIGURE 3.24 – Résultat du métrique

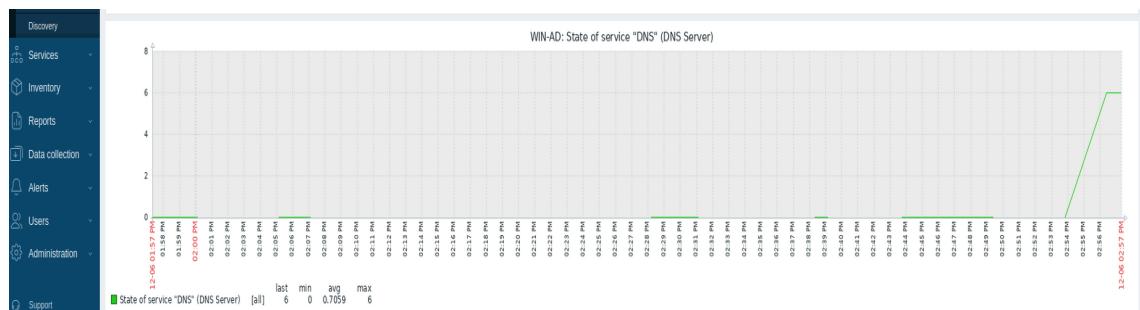


FIGURE 3.25 – Evaluation du service DNS

Nous avons vérifié leur bon fonctionnement en surveillant les valeurs collectées. Des tests de charge ont été réalisés à l'aide d'outils comme stress pour simuler une utilisation élevée de la CPU et observer si le déclencheur fonctionnait correctement. Ce graphique



FIGURE 3.26 – Résultat

affiche l'évaluation de la consommation de la CPU sous forme de courbe, permettant ainsi de visualiser l'évolution au fil du temps. Il aide à identifier les pics de consommation et à observer les tendances globales.



FIGURE 3.27 – Evaluation de la consommation de la CPU

La figure ci-dessous illustre le résultat de l'alerte que nous avons créée. Ce déclencheur, conçu pour surveiller l'utilisation excessive de la mémoire RAM, permet de générer une alerte dès que l'utilisation dépasse un seuil critique de 70

Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Update	Actions	Tags
02:38:22 PM	Warning		PROBLEM		WIN-AD	alert RAM	4m 44s	Update		class:os component:memory component:storage ...
02:38:19 PM	Warning		PROBLEM		WIN-AD	Windows: High swap space usage (less than 20% free)	4m 47s	Update		class:os component:memory component:storage ...

FIGURE 3.28 – le résultat de l'alerte alert RAM

## 3.5 Intégration Zabbix et Kibana

Pour permettre que les alertes de zabbix seront visualisé dans le kibana, il est essentiel de mettre en place une intégration entre les deux plateformes.

Dans un premier temps, nous avons généré un token d'authentification en créant un utilisateur dédié à l'intégration ELK, avec des droits de lecture sur les hôtes supervisés. Ce token est utilisé par Logstash pour interroger l'API.

```
root@douaa-virtual-machine:/home/douaa# curl -X POST -H "Content-Type: application/json" \
-d '{
  "jsonrpc": "2.0",
  "method": "user.login",
  "params": {
    "username": "Admin",
    "password": "zabbix"
  },
  "id": 1
}' \
http://192.168.234.156/zabbix/api_jsonrpc.php
["jsonrpc":"2.0","result":"210ed905903c1df4c9b14b16c6009d36","id":1}root@douaa-virtual-machine:/home/douaa#
```

FIGURE 3.29 – génération du token

Dans un deuxième temps, nous avons créé une pipeline Logstash dans /etc/logstash/conf.d/zabbix.conf pour interroger l'API Zabbix toutes les X minutes et envoyer les données vers Elasticsearch.

```
root@douaa-virtual-machine:/home/douaa
/etc/logstash/conf.d/zabbix.conf

Input {
  http_poller {
    urls => {
      zabbix_problems => {
        method => post
        url => "http://192.168.234.156/zabbix/api_jsonrpc.php"
        headers => {
          "Content-Type" => "application/json"
        }
        body => '{
          "jsonrpc": "2.0",
          "method": "problem.get",
          "params": { "output": "extend", "recent": true },
          "auth": "1929fc0bf10df70f6fb0d46db0a41056",
          "id": 1
        }'
      }
      request_timeout => 30
      schedule => [ cron => "*/* * * * *" ] # toutes les 1 minute pour tester
      codec => json [ target => "zabbix" ]
    }
  }
  filter {
    split field => "result"
    mutate {
      add_field => { "zabbix_item_name" => "%{[result][name]}" }
      add_field => { "zabbix_hostid" => "%{[result][hostid]}" }
      add_field => { "zabbix_value" => "%{[result][lastvalue]}" }
    }
  }
  output {
    stdout { codec => rubydebug } # debug dans les logs
    elasticsearch [
      hosts => [ "http://192.168.234.139:9200" ]
      index => "zabbix-api-alerts"
    ]
  }
}

GNU nano 6.2
```

FIGURE 3.30 – pipeline lsstssogstash dans /etc/logstash/conf.d/zabbix.conf

## 3.6 Conclusion

# chapitre 4

## Sprint 2 : Mise en place du SOC

### 4.1 Introduction

Après avoir installé et configuré Zabbix pour la supervision des performances et des services, nous avons procédé à la mise en place de la solution ELK (Elasticsearch, Logstash, Kibana) afin d'assurer la centralisation, l'analyse et la visualisation des logs dans le cadre de la partie SOC.

### 4.2 Sprint backlog

ID	Tâche	Priorité	Critères d'acceptation
T2.1	En tant qu'un administrateur, je veux créer et configurer toutes les machines virtuelles nécessaires	Élevée	Chaque VM est opérationnelle avec son système d'exploitation installé et connectée au réseau virtuel.
T2.2	Je veux configurer le réseau interne (NAT/Host-only) pour la communication inter-VMs.	Élevée	Toutes les VMs doivent pouvoir communiquer entre elles, tout en ayant une isolation externe.
T2.3	En tant qu'un administrateur, je veux installer ELK(Elasticsearch, Logstash, Kibana) pour la centralisation des logs	Élevée	Les logs système et applicatifs doivent être collectés et consultables via Kibana.
T2.4	Je veux déployer Suricata comme IDS/IPS pour la détection des anomalies réseau	Moyenne	Suricata doit générer des alertes lors de comportements suspects détectés.

TABLE 4.11 – sprint backlog 2

## 4.3 Environnement de travail

Machine virtuelle	OS	RAM	vCPU
ELK	Ubuntu 22.04	7 Go	2
Ubuntu Server	Ubuntu Server 22.04	4 Go	2
Windows Server	Windows Server 2019	4 Go	2
attaquant	Kali linux	2 Go	1

TABLE 4.12 – caractéristiques matérielles et logicielles

### 4.3.1 Environnement logiciel :

Pour la mise en œuvre de notre solution, nous avons choisi d'utiliser l'hyperviseur VMware. Cet outil nous a permis de créer, configurer et gérer plusieurs machines virtuelles

#### Définition de VMware

VMware est une plateforme de virtualisation qui permet de créer et de gérer des machines virtuelles (VM) sur un même hôte physique. Elle offre un environnement isolé pour chaque VM, simulant des ordinateurs complets avec leur propre système d'exploitation.



FIGURE 4.31 – Logo VMware

## 4.4 Mise en place de ELK

Après avoir installé et configuré Zabbix pour la supervision des performances et des services, nous avons procédé à la mise en place de la solution ELK (Elasticsearch, Logstash, Kibana) afin d'assurer la centralisation, l'analyse et la visualisation des logs dans le cadre de la partie SOC.

### 4.4.1 Mise en place de Elasticsearch

Elasticsearch représente le cœur de la plateforme ELK. Il s'agit du moteur d'indexation et de recherche distribué chargé de stocker l'ensemble des logs et événements envoyés par Filebeat.

L'installation a été réalisée via les dépôts officiels Elastic, garantissant une compatibilité

totale avec les autres composants de la stack.

### Configuration du elasticsearch

Une fois installé, nous pouvons modifier le fichier de configuration principal /etc/elasticsearch/elasticsearch.yml pour définir des paramètres :

Les paramètres essentiels configurés sont :

network.host : définit l'adresse IP de la VM, permettant l'accès externe au service.

http.port : précise le port d'écoute de l'API Elasticsearch (9200 par défaut).

la figure ci-dessous montre le fichier de configuration après avoir apporté les modifications. nous avons activé la sécurité native d'Elasticsearch et configuré le chiffrement TLS/SSL

FIGURE 4.32 –

au niveau du transport et de l'API HTTP.

Dans Elasticsearch, la fonctionnalité xpack.security.enabled permet l'authentification utilisateur, la gestion des rôles, le chiffrement des échanges et la sécurisation des API. nous avons activé TLS/SSL pour que kibana accéder à Elasticsearch via l'API HTTP. la figure ci-dessous montre le fichier de configuration.

```
# -----
# Enable security features
xpack.security.enabled: true

xpack.security.enrollment.enabled: false

# Enable encryption for HTTP API client connections, such as Kibana, Logstash, and Agents
xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/http.p12

# Enable encryption and mutual authentication between cluster nodes
xpack.security.transport.ssl:
  enabled: true
  verification_mode: certificate
  keystore.path: certs/transport.p12
  truststore.path: certs/transport.p12
# Create a new cluster with the current node only
# Additional nodes can still join the cluster later
cluster.initial_master_nodes: ["douaa-virtual-machine"]

# Allow HTTP API connections from anywhere
# Connections are encrypted and require user authentication
http.host: 0.0.0.0
```

FIGURE 4.33 – Activation du sécurité dans elasticsearch

#### 4.4.2 Mise en place de logstash

Logstash est le composant responsable de la collecte, du traitement et de la transformation des logs avant leur stockage. Dans notre projet, il joue rôle essentiel pour recevoir les logs provenant de Suricata via Filebeat, appliquer des filtres pour analyser les données

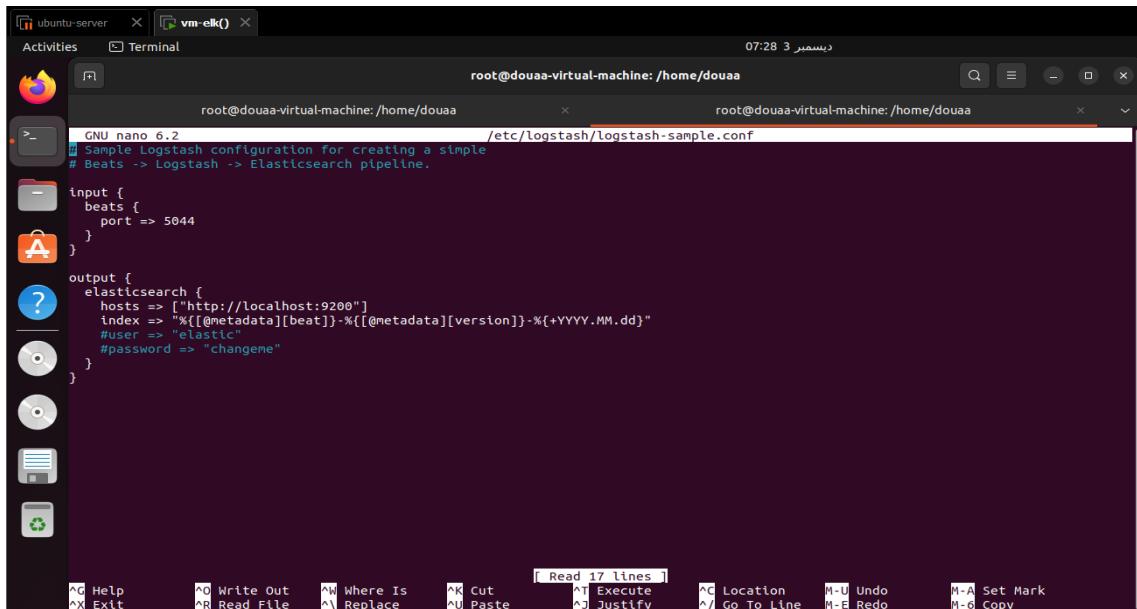
et les envoyer vers Elasticsearch dans un format structuré.

Nous avons créé un pipeline Logstash dans /etc/logstash/logstash-sample.conf cette pipeline définit :

input : réception des logs via Beats

output : envoi des données vers Elasticsearch

La figure ci-dessous montre la configuration



```

# Sample Logstash configuration for creating a simple
# Beats -> Logstash -> Elasticsearch pipeline.
input {
    beats {
        port => 5044
    }
}

output {
    elasticsearch {
        hosts => ["http://localhost:9200"]
        index => "%{@metadata}[beat]-%{@metadata}[version]-%{+YYYY.MM.dd}"
        #user => "elastic"
        #password => "changeme"
    }
}

```

FIGURE 4.34 – configuration du logstash

Une fois configuré, Logstash a été démarré et testé pour garantir l'intégration correcte avec Elasticsearch.

#### 4.4.3 Mise en place de kibana

Kibana constitue la partie visuelle de la stack ELK. C'est à travers cette interface que nous avons pu visualiser et analyser les logs collectés, ainsi que créer des dashboards personnalisés pour la détection d'incidents.

Dans le fichier /etc/kibana/kibana.yml, il est important de s'assurer que les paramètres suivants sont correctement définis :

server.publicBaseUrl : URL publique permettant l'accès à Kibana via un navigateur.

server.port : port d'écoute de l'interface web (5601)

elasticsearch.hosts : URL du service Elasticsearch, en HTTPS

elasticsearch.username, elasticsearch.password : identifiants permettant à Kibana de s'authentifier auprès d'Elasticsearch, nécessaires car la sécurité xPack a été activée.

Ces deux derniers paramètres sont les identifiants utilisés par Kibana pour s'authentifier auprès d'Elasticsearch.

```

root@dousa-virtual-machine:/home/dousa
GNU nano 6.2
server.rootPath: false
# Specifies the public URL at which Kibana is available for end users. If
# 'server.basePath' is configured this URL should end with the same basePath.
server.publicBaseUrl: "http://192.168.234.139:5601"
server.port: 5601
# The maximum payload size in bytes for incoming server requests.
#server.maxPayload: 1048576
# The Kibana server's name. This is used for display purposes.
#server.name: "your-hostname"
# ===== System: Kibana Server (optional) =====
# Enables SSL and paths to the PEM-format SSL certificate and key files, respectively.
# These settings enable SSL for outgoing requests from the Kibana server to the browser.
#server.ssl.enabled: false
#server.ssl.certificate: /path/to/your/server.crt
#server.ssl.key: /path/to/your/server.key
# ===== System: Elasticsearch =====
# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["https://localhost:9200"]
# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
elasticsearch.username: "kibana_system"
elasticsearch.password: "realzzxxv-qLMX2MjxJt"

```

FIGURE 4.35 – activation d’authentification dans kibana

Une fois le service démarré, nous avons accédé à l’interface web via un navigateur à l’adresse “<http://<IP de la VM>:5601>”

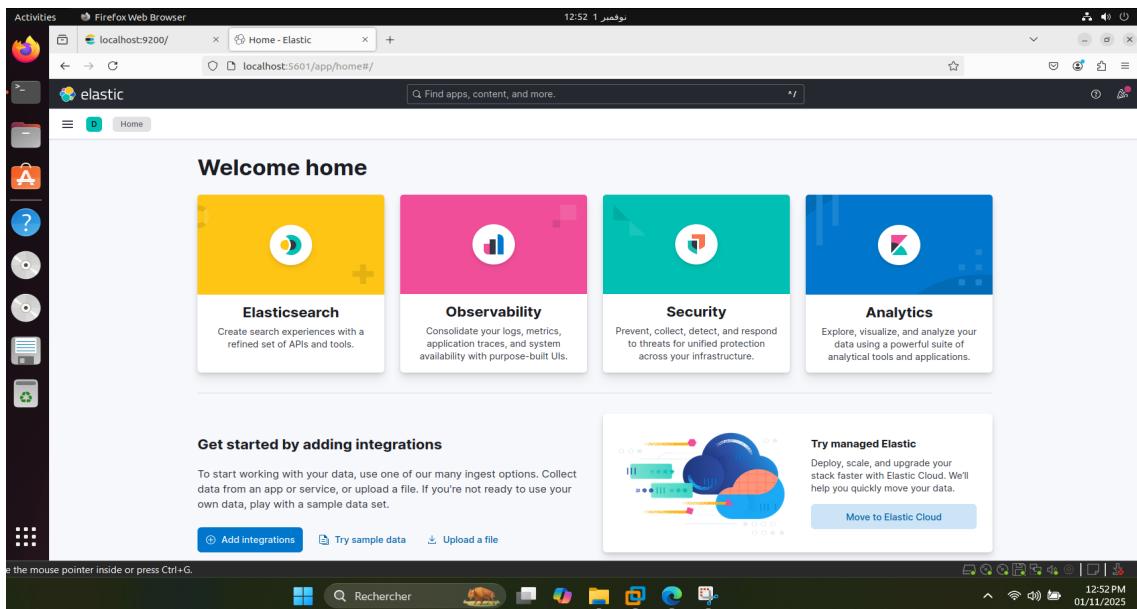


FIGURE 4.36 – Tableau de bord de kibana

## 4.5 Déploiement des agents de collecte

### 4.5.1 mise en place du filebeat dans la hote ubuntu server

Dans cette section, nous présentons la mise en place des agents de collecte de logs nécessaires à l’alimentation de notre plateforme SOC basée sur l’ELK Stack. Nous avons

choisi d'utiliser filebeat pour l'hôte Linux (Ubuntu Server) et winlogbeat pour l'hôte Windows Server, afin d'assurer une collecte fiable, normalisée et compatible avec Logstash.

## Installation de filebeat

Pour collecter les logs système et les transférer vers ELK, nous avons installé filebeat sur l'hôte Ubuntu Server. Nous avons commencé par ajouter le dépôt Elastic puis installé le paquet.

L'installation se fait rapidement, et le service filebeat est automatiquement activé et configuré pour démarrer au boot

```
root@ubuntu-server:/home/douaa# filebeat version
filebeat version 8.19.7 (amd64), libbeat 8.19.7 [18c81c4f9d725daac2b67c74cf2fd04738a96638 built 2025-11-07 12:25:18 +0000 UTC] (FIPS-distribution: false)
root@ubuntu-server:/home/douaa#
```

FIGURE 4.37 – Vérification d'installation du filebeat

## Configuration des modules filebeat

Filebeat propose plusieurs modules préconfigurés permettant d'activer facilement la collecte des journaux les plus courants. Dans notre cas, nous avons activé principalement le module system qui collecte des logs syslog, auth.log et des activités utilisateur et authentification. L'activation se fait via cette commande :

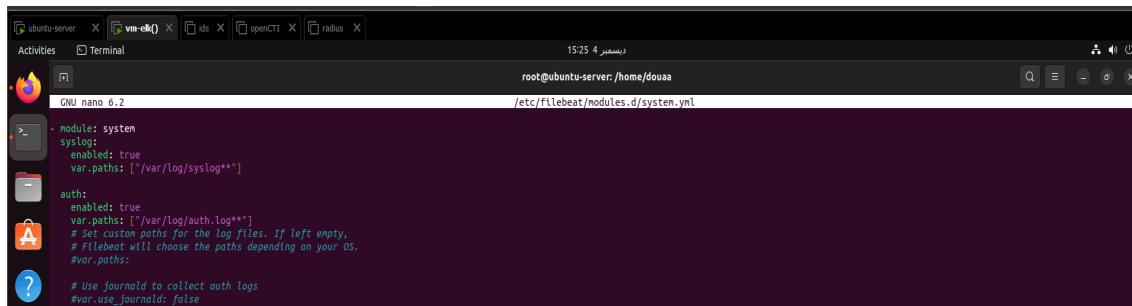


FIGURE 4.38 – configuration des modules dans filbeat

```
root@ubuntu-server:/home/douaa# filebeat modules enable system
Module system is already enabled
```

FIGURE 4.39 – activation des modules dans filbeat

Nous avons ensuite ajusté les chemins des fichiers logs si nécessaire, puis validé la configuration avec :

```
root@ubuntu-server:/home/douaa# filebeat test config
Config OK
```

FIGURE 4.40 – validation du config

```

root@ubuntu-server:/home/douaa
GNU nano 6.2
# -----
# Logstash Output
# -----
output.logstash:
  # The Logstash hosts
  hosts: ["http://192.168.234.139:5044"]
  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificateAuthorities: ["/etc/pki/root/ca.pem"]
  # Certificate for SSL client authentication
  #ssl.certificates: "/etc/pki/client/cert.pem"
  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"

```

FIGURE 4.41 – configuration d'envoie des logs

### Configuration de l'envoi des logs vers Logstash

Nous avons configuré filebeat pour envoyer les logs directement à logstash, afin de bénéficier du filtrage et de l'enrichissement des données avant leur indexation dans Elastic-search. Dans le fichier /etc/filebeat/filebeat.yml, nous avons : Nous avons ensuite démarré le service :

```

root@ubuntu-server:/home/douaa# systemctl restart filebeat
root@ubuntu-server:/home/douaa# systemctl enable filebeat

```

FIGURE 4.42 – Démarrage du filebeat

## 4.5.2 mise en place du WinLogbeat ds la hot Windows server

### Installation de Winlogbeat

Pour la collecte des journaux sous Windows Server, nous avons installé winlogbeat, l'agent elastic dédié aux environnements Windows. Après avoir téléchargé le paquet .zip depuis le site officiel d'Elastic, son contenu a été extrait dans un répertoire. Ensuite, un PowerShell a été ouvert en mode administrateur afin d'exécuter le script ".\service-winlogbeat.ps1", ce qui a permis de créer un service Windows chargé de démarrer automatiquement avec le système et d'assurer la transmission continue des journaux vers la plateforme Elastic.

```

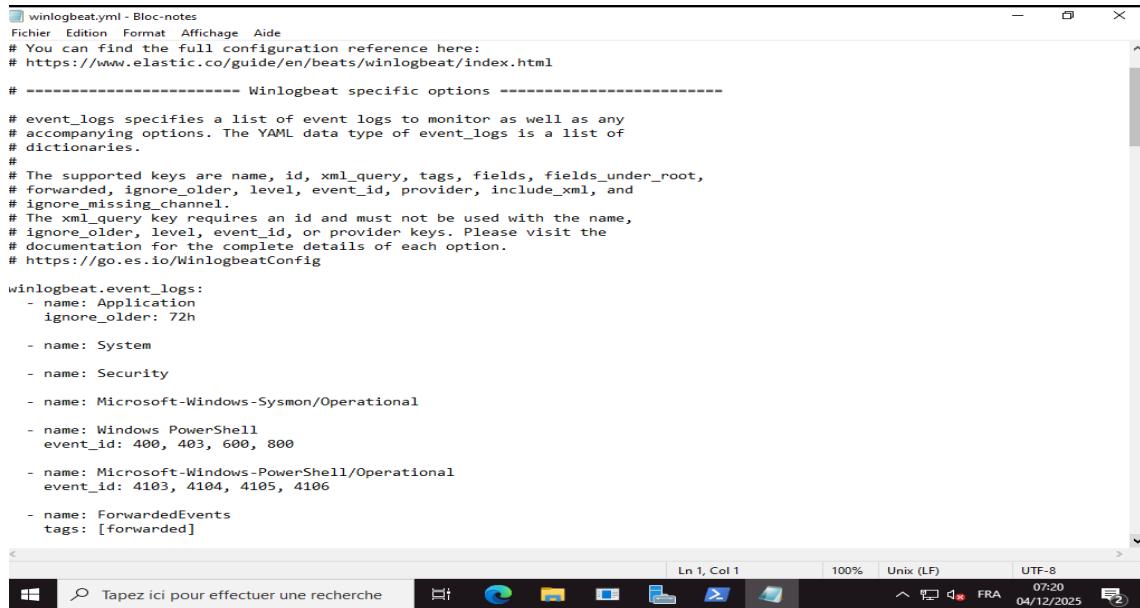
PS C:\Program Files (x86)\Winlogbeat> .\install-service-winlogbeat.ps1

```

FIGURE 4.43 – Installation du winlogbeat

### Choix des journaux windows collectés

Winlogbeat permet de choisir précisément les journaux que nous souhaitons intégrer dans notre SOC. Dans notre configuration, nous avons choisi de collecter les journaux critiques pour l'analyse de sécurité, notamment : les logs de sécurité tel que les tentatives de connexion, les logs systèmes comme l'arrêts de service et les erreurs système et les logs applicatifs



```

winlogbeat.yml - Bloc-notes
Fichier Edition Format Affichage Aide
# You can find the full configuration reference here:
# https://www.elastic.co/guide/en/beats/winlogbeat/index.html

# ----- Winlogbeat specific options -----
# event_logs specifies a list of event logs to monitor as well as any
# accompanying options. The YAML data type of event_logs is a list of
# dictionaries.
#
# The supported keys are name, id, xml_query, tags, fields, fields_under_root,
# forwarded, ignore_older, level, event_id, provider, include_xml, and
# ignore_missing_channel.
#
# The xml_query key requires an id and must not be used with the name,
# ignore_older, level, event_id, or provider keys. Please visit the
# documentation for the complete details of each option.
# https://go.es.io/WinlogbeatConfig

winlogbeat.event_logs:
- name: Application
  ignore_older: 72h
- name: System
- name: Security
- name: Microsoft-Windows-Sysmon/Operational
- name: Windows PowerShell
  event_id: 400, 403, 600, 800
- name: Microsoft-Windows-PowerShell/Operational
  event_id: 4103, 4104, 4105, 4106
- name: ForwardedEvents
  tags: [forwarded]

Tapez ici pour effectuer une recherche

```

FIGURE 4.44 – Choix des logs windows collectés

### Configuration de l'envoi vers Logstash

Comme pour filebeat, nous avons choisi de faire transiter les logs windows par logstash afin de les filtrer et les normaliser. Dans le fichier winlogbeat.yml, nous avons configuré :

```

# ----- Logstash Output -----
output.logstash:
  # The Logstash hosts
  hosts: ["http://192.168.234.139:5044"]

```

FIGURE 4.45 – configuration du winlogbeat

## 4.6 Mise en place du Suricata

Dans le cadre de la partie SOC, Suricata a été déployé comme IDS (Intrusion Detection System). Ce composant joue un rôle essentiel dans l'architecture de sécurité : il surveille le trafic réseau en temps réel, identifie les comportements suspects et génère des logs enrichis. Ces logs, produits au format JSON, sont ensuite transmis à la stack ELK via Filebeat pour être analysés, corrélés et visualisés dans Kibana. Le fichier principal de configuration de Suricata se trouve dans /etc/suricata/suricata.yaml. Nous avons apporté plusieurs modifications afin d'adapter Suricata à notre infrastructure :

### 4.6.1 Activation de la sortie JSON (eve.json)

Nous avons activé la sortie EVE JSON afin que Suricata génère des logs structurés. Ces logs contiennent les alertes, les statistiques et les événements réseau, et sont indispensables pour l'intégration avec ELK :

```

Activities Terminal Des 3 16:50
root@douaa-virtual-machine: /home/douaa
GNU nano 6.2 /etc/suricata/suricata.yaml

# Configure the type of alert (and other) logging you would like.
outputs:
- fast:
    enabled: yes
    filename: fast.log
    append: yes
    #filletype: regular # 'regular', 'unix_stream' or 'unix_dgram'

# Extensible Event Format (nicknamed EVE) event log in JSON format
- eve-log:
    enabled: yes
    filetype: regular #regular/syslog/unix_dgram/unix_stream/redis
    filename: /var/log/suricata/eve.json
    # Enable for multi-threaded eve.json output; output files are amended with
    # with an identifier, e.g., eve.9.json
    #threaded: false
    #prefix: "@cee: " # prefix to prepend to each log entry
    # the following are valid when type: syslog above
    #identity: "suricata"
    #facility: local5
    #level: Info ## possible levels: Emergency, Alert, Critical,
    #          ## Error, Warning, Notice, Info, Debug
    #ethernet: no # log ethernet header in events when available
    #redis:
    #    server: 127.0.0.1
    #    port: 6379
    #    async: true ## if redis replies are read asynchronously
    #    mode: list ## possible values: list|lpush (default), rpush, channel|publish
    #          ## lpush and rpush are using a Redis list, "list" is an alias for lpush
    #          ## publish is using a Redis channel, "channel" is an alias for publish
    #    key: suricata ## key or channel to use (default to suricata)
    #Redis pipelining set up. This will enable to only do a query every
    # 'batch-size' events. This should lower the latency induced by network
    # connection at the cost of some memory. There is no flushing implemented
    # so this setting should be reserved to high traffic suricata deployments.

```

FIGURE 4.46 – activation de la sortie json

#### 4.6.2 Définition de l'interface réseau à surveiller

Nous avons identifié l'interface réseau utilisée par nos machines virtuelles (ens33) et l'avons spécifiée dans la configuration afin que Suricata capture le trafic pertinent :

```

Activities Terminal Des 3 16:51
root@douaa-virtual-machine: /home/douaa
GNU nano 6.2 /etc/suricata/suricata.yaml

af-packet:
- interface: ens33
  # Number of receive threads. "auto" uses the number of cores
  #threads: auto
  # Default clusterid. AF_PACKET will load balance packets based on flow.
  #Clusterid: 0
  # Default AE_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
  # This is only supported for Linux kernel > 3.1
  # Possible values are:
  # * Cluster_flow: all packets of a given flow are sent to the same socket
  # * Cluster_cpu: all packets treated in kernel by a CPU are sent to the same socket
  # * Cluster_qm: all packets linked by network card to a RSS queue are sent to the same
  #   socket. Requires at least Linux 3.14
  # * Cluster_bpf: eBPF file load balancing. See doc/userguide/capture-hardware/ebpf-xdp.rst for
  #   more info.
  # Recommended modes are cluster_flow on most boxes and cluster_cpu or cluster_qm on system
  # with capture card using RSS (requires cpu affinity tuning and system IRQ tuning)
  cluster-type: cluster_flow
  # In some fragmentation cases, the hash can not be computed. If "defrag" is set
  # to yes, the kernel will do the needed defragmentation before sending the packets.
  defrag: yes
  # To use the ring feature of AF_PACKET, set 'use-mmap' to yes
  #use-mmap: yes
  # Lock memory map to avoid it being swapped. Be careful that over
  # subscribing could lock your system
  #mmap-locked: yes
  # Use tpacket_v3 capture mode, only active if use-mmap is true
  # Don't use it in IPS or TAP mode as it causes severe latency
  #tpacket-v3: yes
  # Ring size will be computed with respect to "max-pending-packets" and number
  # of threads. You can set manually the ring size in number of packets by setting
  # the following value. If you are using flow "cluster-type" and have really network
  # intensive single-flow you may want to set the "ring-size" independently of the number
  # of threads:
  #ring-size: 2048
  # Block size is used by tpacket_v3 only. It should set to a value high enough to contain

```

FIGURE 4.47 – définition d'interface

#### 4.6.3 Mise à jour des règles de détection

Afin de bénéficier des dernières signatures de détection, nous avons mis à jour les règles Suricata . Après avoir modifié le fichier de configuration et ajouté nos règles, nous avons redémarré le service Suricata

FIGURE 4.48 – Rédemarrage du suricata

## 4.7 Intégration Suricata et ELK

Cette intégration vise à permettre Suricata à transmettre ses journaux (alertes, logs réseau, anomalies) vers la pile ELK pour assurer leur centralisation, visualisation et corrélation.

Cette chaîne de collecte nous offre la possibilité de transformer des événements bruts en informations exploitables et de les visualiser sur des tableaux de bord Kibana spécifiquement dédiés.

### 4.7.1 mise en place du filebeat :

Nous avons opté pour la mise en place de filebeat en tant qu'agent de collecte des journaux. Il s'agit d'un outil léger appartenant à la suite Elastic, conçu pour la surveillance en temps réel des fichiers de log. Filebeat transmet ces journaux vers Logstash ou directement vers Elasticsearch.

Dans notre contexte, filebeat joue un rôle essentiel : il collecte les logs générés par Suricata, notamment le fichier eve.json, et les transmet vers la stack ELK pour analyse et visualisation dans Kibana. Nous avons installé filebeat sur la machine Ubuntu hébergeant Suricata, en utilisant les dépôts officiels Elastic. Une fois l'installation terminée, le service filebeat est disponible et prêt à être configuré.

#### Configuration du module Suricata

Le fichier principal de configuration se trouve dans /etc/filebeat/filebeat.yml. Nous avons activé le module Suricata dans filebeat afin de bénéficier d'un enrichissement automatique des logs et d'une structuration adaptée. Le fichier de configuration se trouve dans /etc/filebeat/modules.d/suricata.yml  
Voici les paramètres que nous avons définis :

enabled : true : active le module Suricata.

var-paths : indique le chemin du fichier eve.json généré par Suricata.

Cette configuration permet à Filebeat de parser les logs JSON de Suricata et de les en-

```
GNU nano 6.2                               /etc/filebeat/modules.d/suricata.yml
Module: suricata
# Docs: https://www.elastic.co/guide/en/beats/filebeat/8.19/filebeat-module-suricata.html

- module: suricata
  # All logs
  eve:
    enabled: true
    var_paths: ["/var/log/suricata/eve.json"]
```

FIGURE 4.49 – configuration des modules

richir avec des champs spécifiques.

Ensuite, nous avons configuré filebeat pour envoyer les logs vers Elasticsearch :

### Configuration de la sortie vers Elasticsearch

Toujours dans le fichier filebeat.yml, nous avons défini la sortie vers Elasticsearch, afin que les logs collectés soient indexés directement dans notre moteur de recherche

```
GNU nano 6.2                               /etc/filebeat/filebeat.yml
# Configure what output to use when sending the data collected by the beat.

# ----- Elasticsearch Output -----
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.234.139:9200"]
```

FIGURE 4.50 – configuration de la sortie vers elasticsearch

### Configuration de la sortie vers Kibana

Dans le fichier principal /etc/filebeat/filebeat.yml, nous avons défini les paramètres de connexion à Kibana pour permettre le chargement des dashboards et la visualisation des données host : correspond à l'URL de notre instance Kibana.

```
GNU nano 6.2                               /etc/filebeat/filebeat.yml
# Versions, this URL points to the dashboard archive on the artifacts.elastic.co
# website.
#setup.dashboards.url:

# ====== Kibana ======
# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:
  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "http://192.168.234.139:5601"

  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded. By default,
  # the Default Space will be used.
  #space.id:
```

FIGURE 4.51 – configuration de la sortie vers kibana

L'adresse IP correspond à la machine.

Le port 5601 est celui utilisé par kibana.

## 4.8 Collecte, normalisation et corrélation des logs

Nous décrivons le processus que nous avons mis en place pour assurer une centralisation des logs, leur transformation et normalisation au sein de la stack ELK, ainsi que leur exploitation à travers des visualisations et des corrélations de sécurité dans Kibana. L'objectif est d'unifier les données (Linux, Windows, réseau) provenant de la hôte ubuntu server, windows server et l'IDS suricata afin d'obtenir une vision pour la détection d'incidents.

## 4.8.1 Collecte des différentes sources de logs

### Logs système Linux – Filebeat

La figure ci-dessous présente un extrait des logs collectés depuis la VM Ubuntu Server, visibles dans Kibana via l'interface Discover.

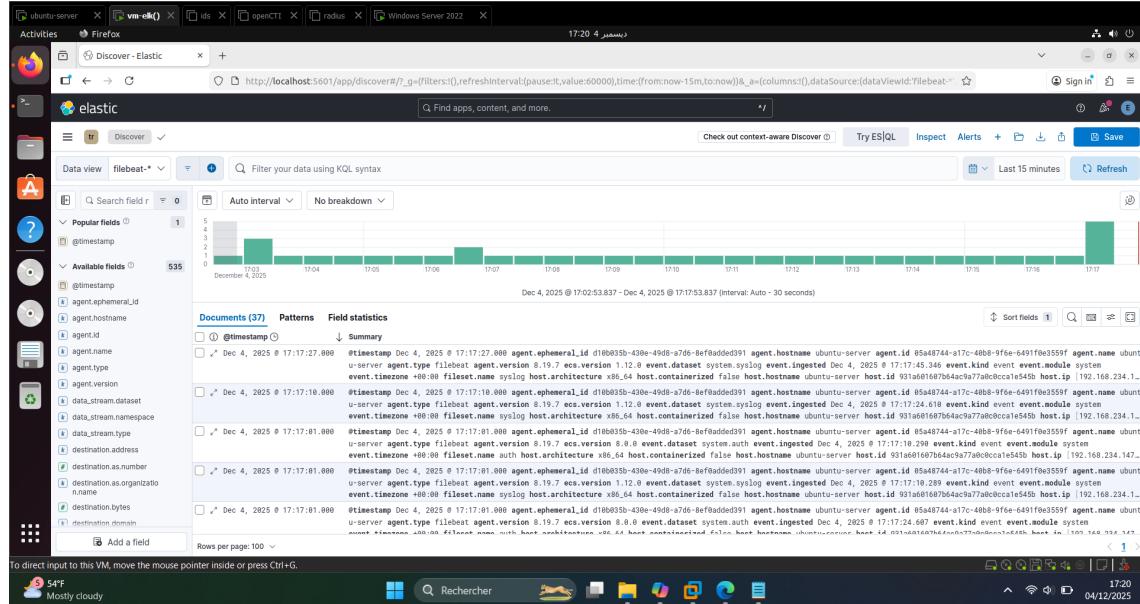


FIGURE 4.52 – extrait des logs du serveur ubuntu

### Logs windows server

La figure ci-dessous présente un extrait des logs collectés depuis la VM windows server, visibles dans Kibana via l'interface Discover.

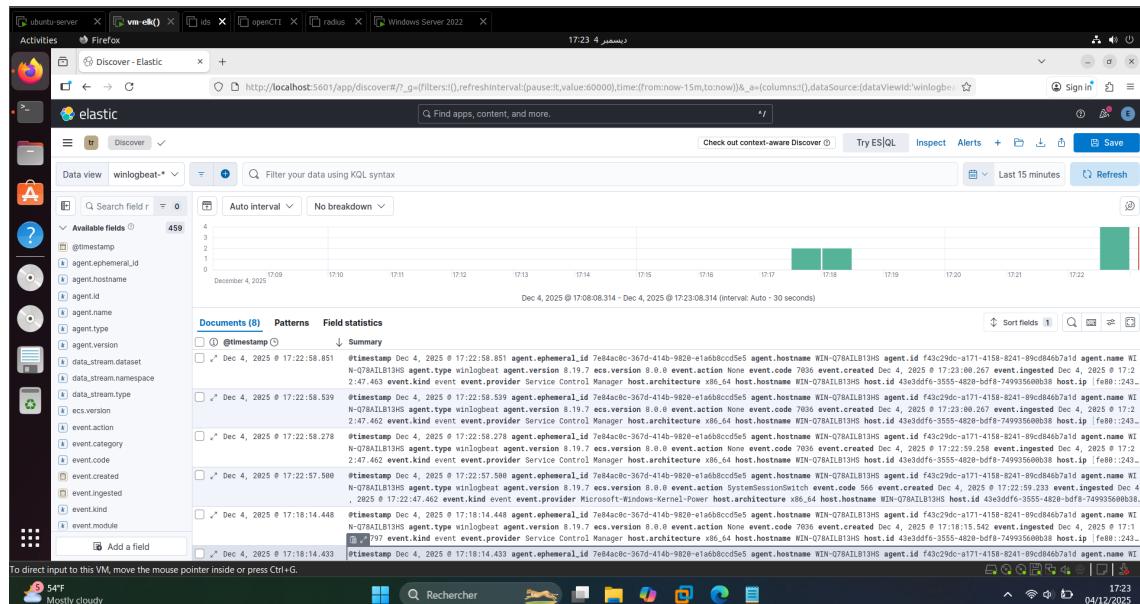


FIGURE 4.53 – extrait des logs du serveur windows

## Logs réseau

La figure ci-dessous présente un extrait des logs collectés depuis la VM ids suricata, visibles dans Kibana via l'interface Discover.

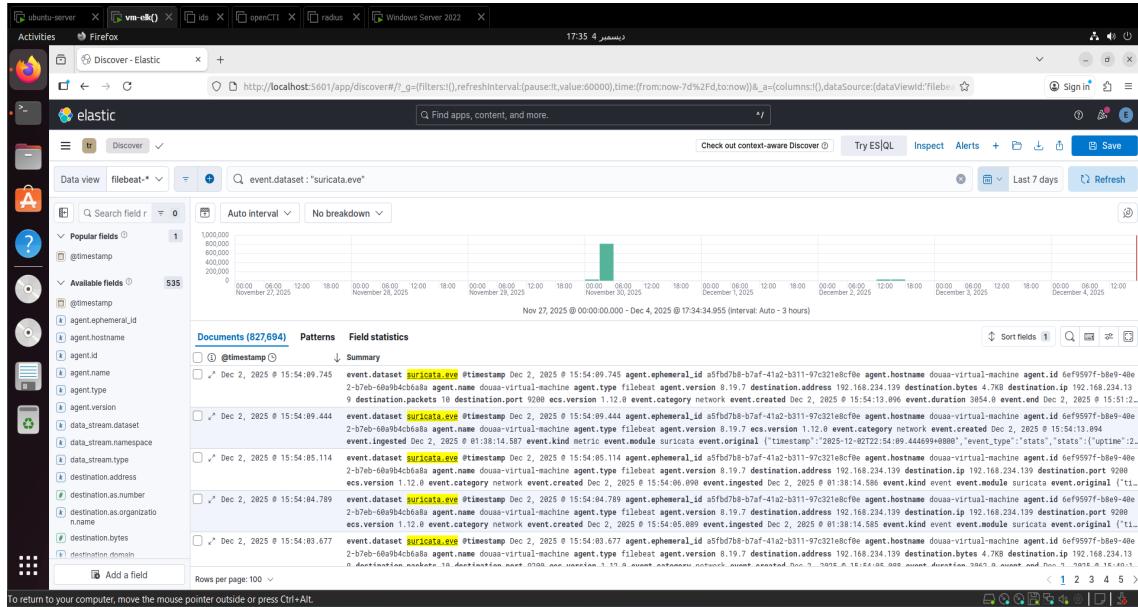


FIGURE 4.54 – extrait des logs de l'ids

### 4.8.2 Parsing et traitement

7.2 parsing et traitement Après la collecte brute réalisée par filebeat, winlogbeat et Suricata, la première étape du traitement consiste à vérifier que les logs sont correctement parsés, c'est-à-dire décodés, structurés et enrichis avant leur indexation dans Elasticsearch. Grâce aux pipelines intégrés et au standard ECS (Elastic Common Schema), chaque log est automatiquement organisé en champs normalisés (host., event., process., network., etc.), facilitant ensuite la corrélation et la détection.

## Logs système Linux

Filebeat a été déployé sur la machine « ubuntu-server » afin de collecter les logs système présents dans /var/log/syslog. La capture ci-dessous montre que les logs Linux ont été correctement parsés par le module system de filebeat.

Table JSON	
<input type="text"/> Search field names or values	
Field	Value
agent.ephemeral_id	769221e8-07af-40b7-8cf8-22dfdc1025a8
agent.hostname	ubuntu-server
agent.id	05a48744-a17c-40b8-9f6e-6491f0e3559f
agent.name	ubuntu-server
agent.type	filebeat
agent.version	8.19.7
ecs.version	1.12.0
event.dataset	system.syslog
event.ingested	Nov 28, 2025 @ 12:34:49.097
event.kind	event
event.module	system
event.timezone	+00:00
fileset.name	syslog
host.architecture	x86_64
host.containerized	false
host.hostname	ubuntu-server
host.id	931a601607b64ac9a77a0c0cca1e545b
host.ip	[192.168.234.147, fe80::20c:29ff:febf:973c]
host.mac	00-0C-29-FB-97-3C

FIGURE 4.55 – logs du serveur ubuntu parsés

## Logs réseau

Le fichier de sortie de Suricata /var/log/suricata/eve.json est lui aussi collecté par filebeat.

La capture suivante montre un événement réseau passé via le pipeline Suricata.

Discover .ds-filebeat-8.19.7-2025.11.18-000001#j5IK35oBsCb4Zcg5l540	
Table JSON	
1	{
2	"_index": ".ds-filebeat-8.19.7-2025.11.18-000001",
3	"_id": "j5IK35oBsCb4Zcg5l540",
4	"_version": 1,
5	"_source": {
6	"agent": {
7	"name": "douaa-virtual-machine",
8	"id": "6ef9597f-b8e9-40e2-b7eb-60a9b4cb6a8a",
9	"ephemeral_id": "a5fb7b8-b7af-41a2-b311-97c321e8cf0e",
10	"type": "filebeat",
11	"version": "8.19.7"
12	},
13	"log": {
14	"file": {
15	"path": "/var/log/suricata/eve.json"
16	},
17	"offset": 460122497
18	},
19	"destination": {
20	"address": "192.168.234.139",
21	"port": 9200,
22	"bytes": 4862,
23	"ip": "192.168.234.139",
24	"packets": 10
25	},
26	"source": {
27	"address": "192.168.234.146",

FIGURE 4.56 – logs du réseau parsés

## Logs windows server

La capture ci-dessous montre que les logs windows ont été correctement parsés par winlogbeat.

FIGURE 4.57 – logs du serveur windows parsés

### 4.8.3 Normalisation des données dans Elasticsearch

Elastic Common Schema (ECS) est un schéma standardisé qui définit comment les champs doivent être organisés dans Elasticsearch. Grâce à ECS, toutes les machines ont un champ `host.*`, tous les événements ont un champ `event.*`, tous les logs réseau ont un champ `source.*` et `destination.*`

### 4.8.4 Visualisations dans Kibana

Avant d'exploiter les données, il est nécessaire de créer une Data View basée sur l'index contenant les logs (ex : `suricata-*`, `winlogbeat-*`, `filebeat-*`). Cette étape permet à Kibana d'interpréter correctement les champs, en particulier le champ temporel (`@timestamp`), indispensable pour les analyses chronologiques.

## 4.9 Test et résultat

### Dans la machine windows server

Nous avons exécuté un test d'intrusion simulant une attaque brute-force sur le service SSH du serveur windows.

```
(kali㉿kali)-[~]
$ hydra -t 4 -V -l Administrator -P passwords.txt rdp://192.168.234.145
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-12-02 08:07:29
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 10 login tries (l:1:p:10), -3 tries per task
[DATA] attacking rdp://192.168.234.145:145:3389
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "123456" - 1 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "admin123" - 2 of 10 [child 1] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "password" - 3 of 10 [child 2] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "azerty" - 4 of 10 [child 3] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "root" - 5 of 10 [child 1] (0/0)
[ERROR] freerd: The connection failed to establish.
[ERROR] freerd: The connection failed to establish.
[RE-ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "123456" - 5 of 10 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "azerty" - 5 of 10 [child 3] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "toor" - 6 of 10 [child 3] (0/0)
[ERROR] freerd: The connection failed to establish.
[ERROR] freerd: The connection failed to establish.
[RE-ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "password" - 6 of 10 [child 2] (0/0)
[RE-ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "123456" - 6 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "administrator" - 7 of 10 [child 0] (0/0)
[ERROR] freerd: The connection failed to establish.
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "welcome" - 8 of 10 [child 2] (0/0)
[RE-ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "root" - 8 of 10 [child 1] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "qwerty" - 9 of 10 [child 1] (0/0)
[ATTEMPT] target 192.168.234.145 - login "Administrator" - pass "test123" - 10 of 10 [child 3] (0/0)
[ERROR] freerd: The connection failed to establish.
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-12-02 08:07:34
```

FIGURE 4.58 – attaque brute-force simulé

Par défaut, Suricata a plusieurs règles mais pour ce projet nous avons créé des règles tel que règle contre brute force. Il envoie la donnée au format json, ensuite transférée vers Logstash puis Elasticsearch. La figure ci-dessus montre l’alerte générée par Suricata lors de l’attaque brute-force SSH. Chaque tentative suspecte est identifiée et classifiée selon les règles IDS activées.

```
root@douaa-virtual-machine:/home/douaa
tail -f /var/log/suricata/fast.log
11/30/2025-10:28:10.240993 [**] [1:1000001:1] SSH brute force attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.158:47008 -> 192.168.234.147:22
11/30/2025-10:28:10.3713531 [**] [1:1000001:1] SSH brute force attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.158:63932 -> 192.168.234.147:22
11/30/2025-10:46:58.672618 [**] [1:1000001:1] SSH brute force attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.158:41471 -> 192.168.234.147:22
11/30/2025-11:11:15.549798 [**] [1:1000001:1] SSH brute force attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.158:11366 -> 192.168.234.147:22
12/02/2025-20:33:42.446968 [**] [1:1000009:1] RDP SYN test [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.158:41818 -> 192.168.234.145:3389
12/02/2025-20:48:03.446322 [**] [1:1000009:1] RDP SYN test [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.1:56982 -> 192.168.234.145:3389
12/02/2025-20:54:54.157428 [**] [1:1000009:1] RDP SYN test [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.1:43597 -> 192.168.234.145:3389
12/02/2025-20:57:55.235871 [**] [1:1000009:1] RDP SYN test [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.1:52843 -> 192.168.234.145:3389
12/02/2025-21:06:07.426933 [**] [1:1000009:1] RDP SYN test [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.158:41998 -> 192.168.234.145:3389
12/02/2025-21:07:31.686149 [**] [1:1000009:1] RDP SYN test [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] [TCP] 192.168.234.158:50288 -> 192.168.234.145:3389
```

FIGURE 4.59 – alerte générée par suricata

En plus de Suricata, nous avons créé une règle de détection dans kibana, basée sur les logs SSH collectés par winlogbeat.

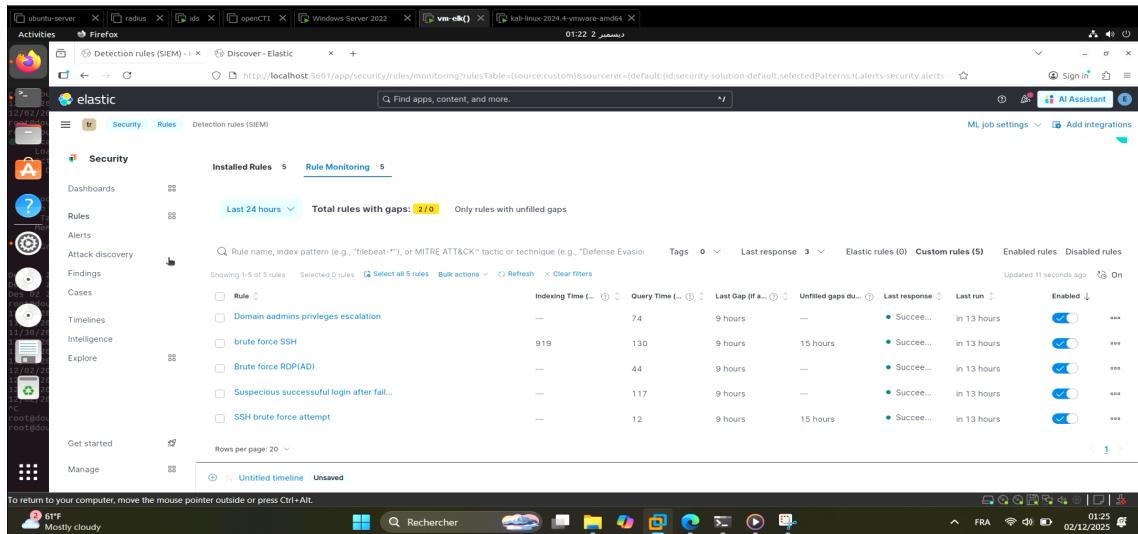


FIGURE 4.60 – création du règle

Une alerte est automatiquement affichée dans Kibana SIEM (Security → Alerts) Dans

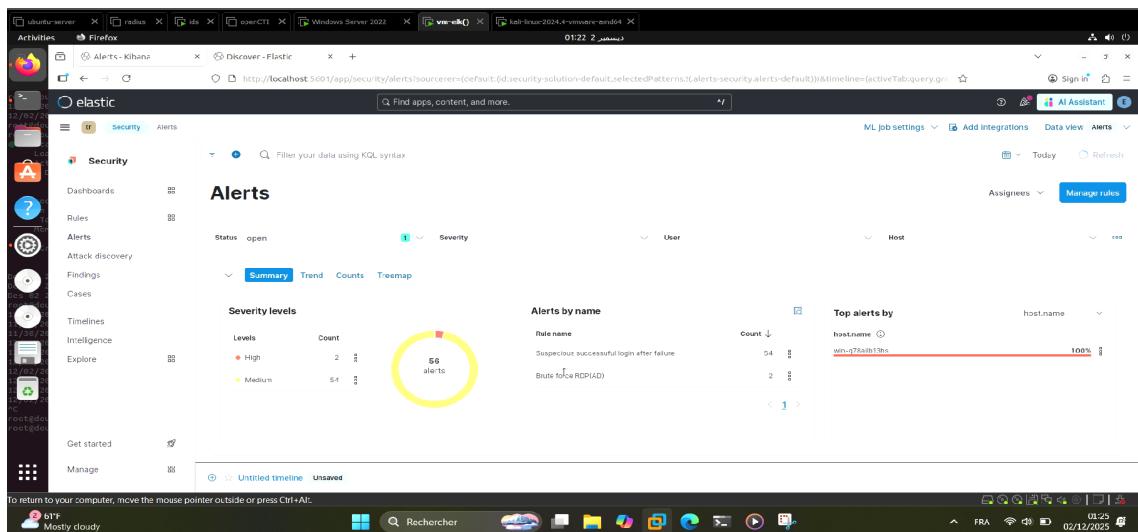


FIGURE 4.61 – alerte affichée dans kibana

### la machine ubutnu server

Nous avons exécuté un test d'intrusion simulant une attaque brute-force sur le service SSH du serveur ubuntu.

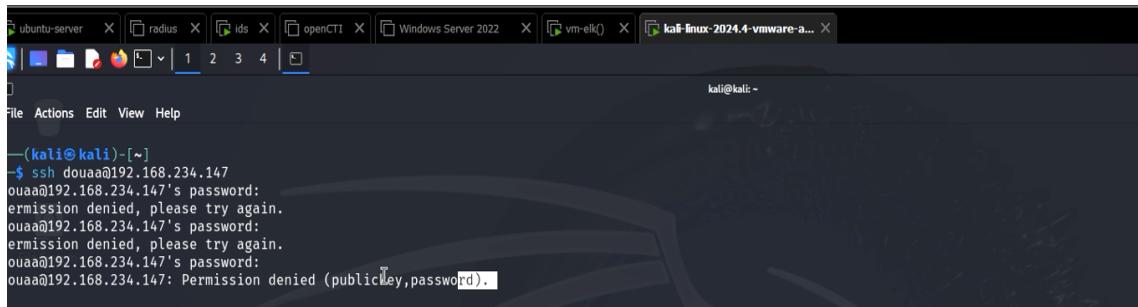


FIGURE 4.62 – attaque brute-force dans le serveur ubuntu

Par défaut, Suricata a plusieurs règles mais pour ce projet nous avons créé des règles tel que règle contre brute force. Il envoie la donnée au format eve json, ensuite transférée vers Logstash puis Elasticsearch. La figure ci-dessus montre l'alerte générée par Suricata lors de l'attaque brute-force SSH. Chaque tentative suspecte est identifiée et classifiée selon les règles IDS activées.

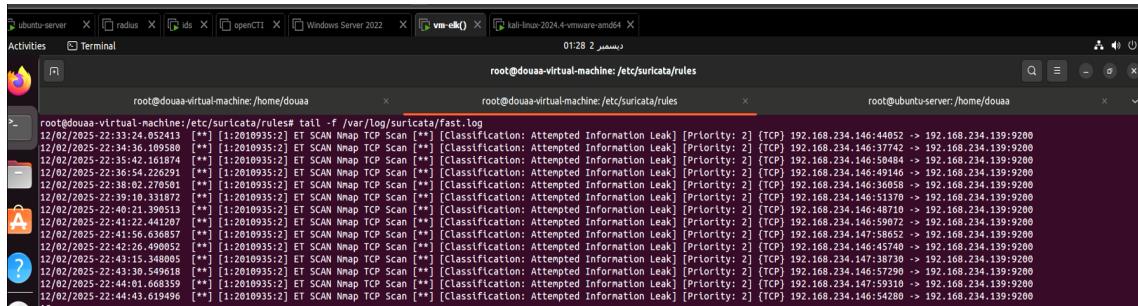


FIGURE 4.63 – alerte généré par suricata

En plus de Suricata, nous avons créé une règle de détection dans kibana, basée sur les logs SSH collectés par filebeat.

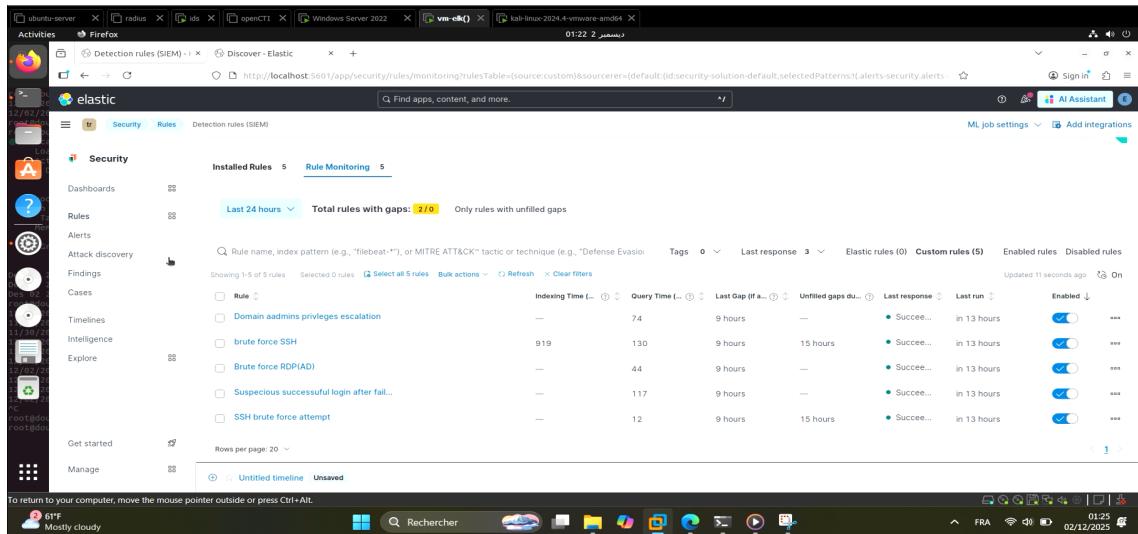


FIGURE 4.64 – création du règle de détection dans kibana

Une alerte est automatiquement affichée dans Kibana SIEM (Security → Alerts)

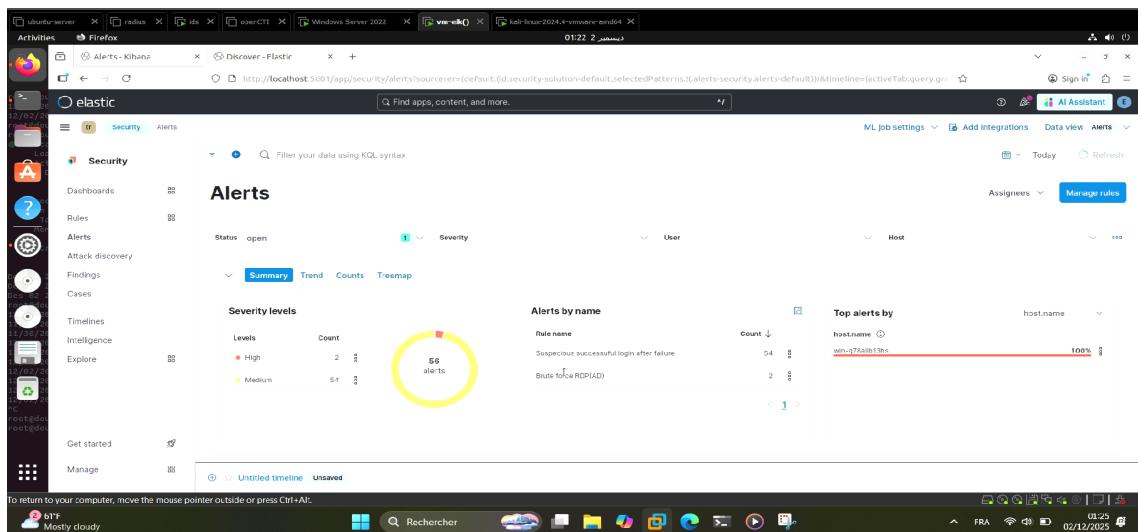


FIGURE 4.65 – alerte affichée

## 4.10 Conclusion

La mise en place de la solution SOC a permis de démontrer concrètement la capacité de l’infrastructure à collecter, centraliser et analyser les événements de sécurité. Grâce à l’intégration de l’ids Suricata et le stack ELK, nous avons validé un flux complet allant de la détection des incidents jusqu’à leur visualisation dans des dashboards unifiés.

# chapitre 5

## Sprint 3 : Mise en place du threat intelligence

### 5.1 Introduction

Après avoir installé et configuré ELK et Suricata en tant que solution de supervision et détection d'intrusion au sein du SOC, nous avons procédé à l'installation d'OpenCTI afin d'enrichir notre plateforme avec une base de renseignement sur les menaces.

### 5.2 Sprint backlog

ID	Tâche	Priorité	Critères d'acceptation
T2.1	En tant qu'un administrateur, je veux créer et configurer toutes les machines virtuelles nécessaires	Élevée	Chaque VM est opérationnelle avec son système d'exploitation installé et connectée au réseau virtuel.
T2.2	Je veux configurer le réseau interne (NAT/Host-only) pour la communication inter-VMs.	Élevée	Toutes les VMs doivent pouvoir communiquer entre elles, tout en ayant une isolation externe.
T2.3	Je veux installer et configurer OpenCTI pour la threat intelligence	Élevée	OpenCTI doit collecter et afficher des indicateurs de compromission (IoC).
T2.4	Je veux intégrer OpenCTI avec ELK pour enrichir les détections.	Moyenne	Les IoC d'OpenCTI doivent apparaître corrélés avec les logs dans ELK.

TABLE 5.13 – sprint backlog 3

## 5.3 Environnement de travail

### 5.3.1 Environnement matériel :

Le tableau représente l'ensemble des ressources utilisées ainsi que leurs caractéristiques.

Machine virtuelle	OS	RAM	vCPU
OpenCTI	Ubuntu 22.04	8 Go	4
attaquant	Kali linux	2 Go	1

TABLE 5.14 – caractéristiques matérielles et logicielles

### 5.3.2 Environnement logiciel :

Pour la mise en œuvre de notre solution, nous avons choisi d'utiliser l'hyperviseur VMware. Cet outil nous a permis de créer, configurer et gérer plusieurs machines virtuelles.

#### Définition de VMware

VMware est une plateforme de virtualisation qui permet de créer et de gérer des machines virtuelles (VM) sur un même hôte physique. Elle offre un environnement isolé pour chaque VM, simulant des ordinateurs complets avec leur propre système d'exploitation.



FIGURE 5.66 – Logo VMware

## 5.4 Mise en place du Open CTI

Avant de procéder à la configuration d'OpenCTI, nous avons déployer la plateforme à l'aide de Docker. L'utilisation de Docker présente plusieurs avantages : elle permet de simplifier le déploiement, d'assurer une portabilité maximale et de garantir un environnement isolé et reproductible pour chaque composant. Grâce à Docker Compose, nous pouvons orchestrer facilement les différents services nécessaires au fonctionnement d'OpenCTI (base de données, moteur de recherche, API, interface web, etc.) sans avoir à gérer manuellement leurs dépendances. Cette approche nous offre une solution flexible et rapide à mettre en œuvre, tout en respectant les bonnes pratiques de déploiement moderne.

### 5.4.1 Configuration du OpenCTI

Nous avons commencé par configurer le fichier docker-compose.yml afin de définir les variables essentielles pour l'administration d'OpenCTI, nous avons utilisé un fichier .env pour centraliser les variables d'environnement nécessaires au bon fonctionnement de la plateforme. Chaque variable définie dans .env est automatiquement injectée dans le fichier docker-compose.yml, ce qui simplifie la gestion des paramètres sensibles et techniques. Voici les principales variables que nous avons définies :

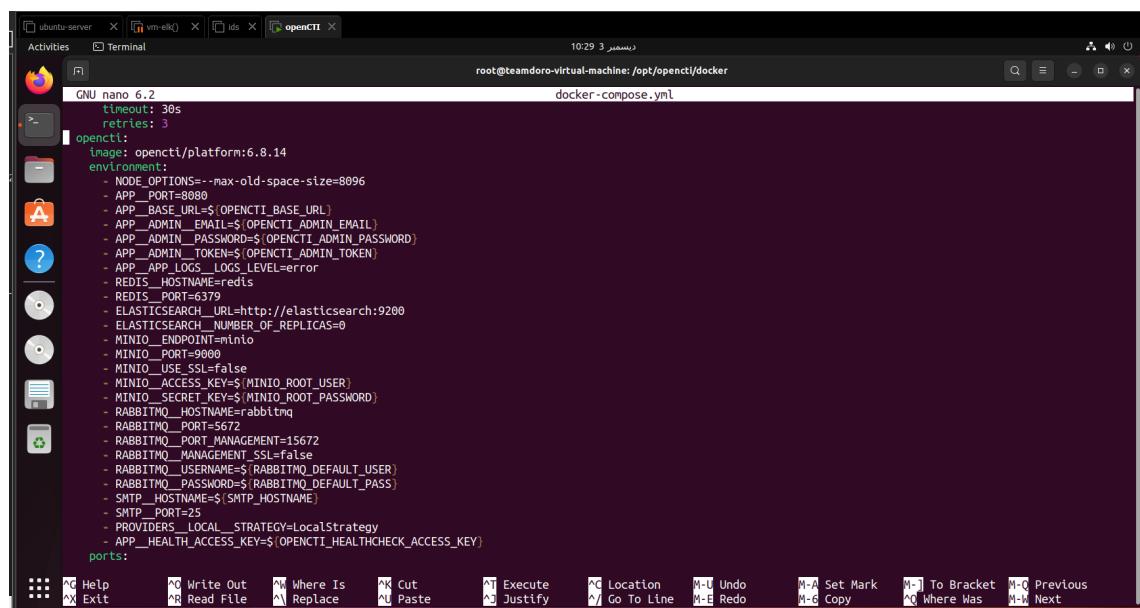
**OPENCTI-ADMIN-EMAIL** : adresse e-mail de l'administrateur principal.

**OPENCTIADMIN-PASSWORD** : mot de passe associé au compte admin.

**OPENCTI-ADMIN-TOKEN** : jeton d'authentification utilisé par les connecteurs pour interagir avec l'API OpenCTI.

**OPENCTI-BASE-URL** : URL publique d'accès à l'interface web d'OpenCTI.

**OPENCTIHEALTHCHECK-ACCESS-KEY** : clé utilisée pour les vérifications de santé du service.



```
root@teamdoro-virtual-machine:/opt/opencti/docker
GNU nano 6.2
version: '3'
services:
  opencti:
    image: opencti/platform:6.8.14
    environment:
      - NODE_OPTIONS=--max-old-space-size=8096
      - APP_PORT=8080
      - APP_BASE_URL=${OPENCTI_BASE_URL}
      - APP_ADMIN_EMAIL=${OPENCTI_ADMIN_EMAIL}
      - APP_ADMIN_PASSWORD=${OPENCTI_ADMIN_PASSWORD}
      - APP_ADMIN_TOKEN=${OPENCTI_ADMIN_TOKEN}
      - APP_APP_LOGS_LOGS_LEVEL=error
      - REDIS_HOSTNAME=redis
      - REDIS_PORT=6379
      - ELASTICSEARCH_URL=http://elasticsearch:9200
      - ELASTICSEARCH_NUMBER_OF_REPLICAS=0
      - MINIO_ENDPOINT=minio
      - MINIO_PORT=9000
      - MINIO_USE_SSL=false
      - MINIO_ACCESS_KEY=${MINIO_ROOT_USER}
      - MINIO_SECRET_KEY=${MINIO_ROOT_PASSWORD}
      - RABBITMQ_HOSTNAME=rabbitmq
      - RABBITMQ_PORT=5672
      - RABBITMQ_MANAGEMENT_PORT=15672
      - RABBITMQ_MANAGEMENT_SSL=false
      - RABBITMQ_USERNAME=${RABBITMQ_DEFAULT_USER}
      - RABBITMQ_PASSWORD=${RABBITMQ_DEFAULT_PASS}
      - SMTP_HOSTNAME=${SMTP_HOSTNAME}
      - SMTP_PORT=25
      - PROVIDERS_LOCAL_STRATEGY=LocalStrategy
      - APP_HEALTH_ACCESS_KEY=${OPENCTI_HEALTHCHECK_ACCESS_KEY}
    ports:
```

FIGURE 5.67 – Configuration du fichier docker-compose

La figure suivante illustre le fichier docker-compose.yml après la modification des paramètres dans le fichier .env

```
GNU nano 6.2 .env
OPENCTI_ADMIN_EMAIL=admin@opencti.io
OPENCTI_ADMIN_PASSWORD=admin123
OPENCTI_ADMIN_TOKEN=f7011f4f-37d0-46f6-bd99-a59631bfebfd
OPENCTI_BASE_URL=http://192.168.234.148:8080
OPENCTI_HEALTHCHECK_ACCESS_KEY=f44010a3-418b-4f11-9cc0-273f5db2d0b6
MINIO_ROOT_USER=opencti
MINIO_ROOT_PASSWORD=admin123
RABBITMQ_DEFAULT_USER=opencti
RABBITMQ_DEFAULT_PASS=admin123
CONNECTOR_EXPORT_FILE_STIX_ID=dd817c8b-abae-460a-9ebc-97b1551e70e6
CONNECTOR_EXPORT_FILE_CSV_ID=7ba187fb-fde8-4063-92b5-c3da34060dd7
CONNECTOR_EXPORT_FILE_TXT_ID=ca715d9c-bd64-4351-91db-33a8d728a58b
CONNECTOR_IMPORT_FILE_STIX_ID=72327164-0b35-482b-b5d6-a5a3f76b845f
CONNECTOR_IMPORT_DOCUMENT_ID=c3970f8a-ce4b-4497-a381-20b7256f56f0
CONNECTOR_ANALYSIS_ID=4dffdf77c-ec11-4abe-bca7-fd997f79fa36
KTM_COMPOSER_ID=8215614c-7139-422e-b825-b20fd2a13a23
SMTP_HOSTNAME=localhost
ELASTIC_MEMORY_SIZE=4G
COMPOSE_PROJECT_NAME=opencti
```

FIGURE 5.68 – Configuration des variables

Une fois la configuration terminée, nous avons lancé le service OpenCTI :

```
Status: Downloaded newer image for filigran/xtm-composer:1.0.0
Creating opencti_redis_1 ... done
Creating opencti_rsa-key-generator_1 ... done
Creating opencti_rabbitmq_1 ... done
Creating opencti_minio_1 ... done
Creating opencti_elasticsearch_1 ... done
Creating opencti_opencti_1 ... done
Creating opencti_worker_1 ... done
Creating opencti_worker_2 ... done
Creating opencti_worker_3 ... done
Creating opencti_connector-analysis_1 ... done
Creating opencti_connector-import-document_1 ... done
Creating opencti_connector-export-file-txt_1 ... done
Creating opencti_connector-import-file-stix_1 ... done
Creating opencti_xtm-composer_1 ... done
Creating opencti_connector-export-file-csv_1 ... done
Creating opencti_connector-export-file-stix_1 ... done
```

FIGURE 5.69 – Lancement du service OpenCTI

Nous avons ensuite accédé à l'interface web via l'URL :

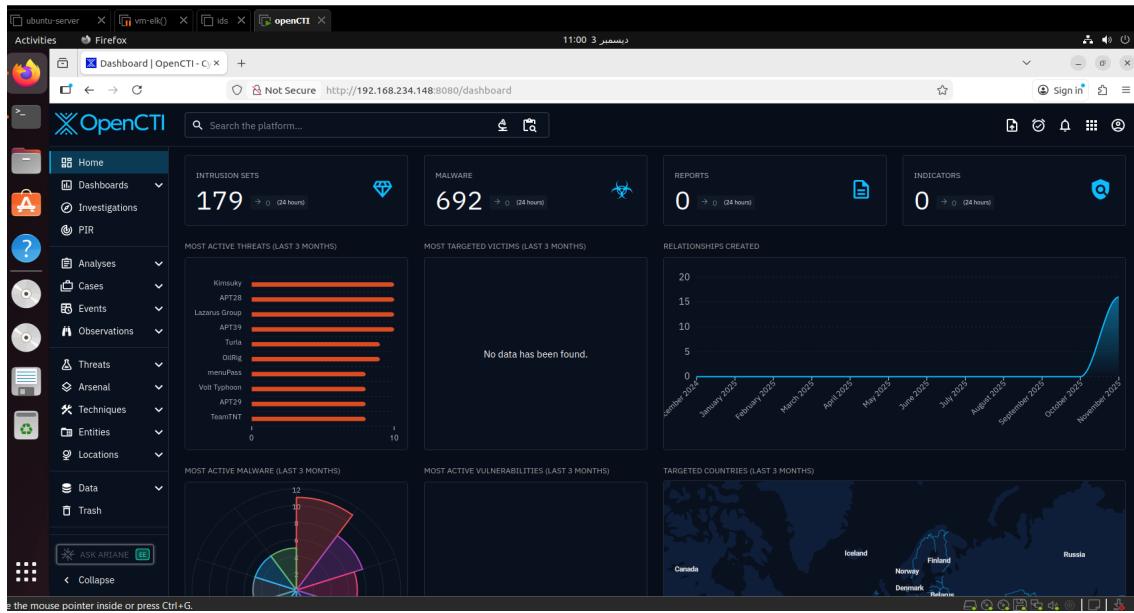


FIGURE 5.70 – Tableau de bord du OpenCTI

### 5.4.2 Mise en place du MITRE ATTACK

Une fois la plateforme OpenCTI opérationnelle et accessible via son interface web, nous avons procédé à l'enrichissement de la solution par l'ajout d'un connecteur de Threat Intelligence. Dans le cadre de notre SOC, nous avons choisi d'intégrer le connecteur MITRE ATTACK, qui constitue une référence mondiale en matière de classification des tactiques, techniques et procédures (TTPs) utilisées par les attaquants.

Le connecteur MITRE ATTACK a été déployé sous forme de conteneur Docker, conformément à l'architecture retenue pour OpenCTI. Sa configuration repose également sur l'utilisation de variables d'environnement définies dans le fichier .env. Les paramètres principaux configurés sont les suivants :

```
GNU nano 4.2
OPENCTI_ADMIN_EMAIL=admin@opencti.lo
OPENCTI_ADMIN_PASSWORD=dmin123
OPENCTI_ADMIN_TOKEN=2cfb3409-272e-4d69-a2a3-9fdc1d96dc5b
OPENCTI_BASE_URL=http://192.168.234.148:8080
OPENCTI_HEALTHCHECK_ACCESS_KEY=75547dbc-39cd-4da2-8416-65addb75a01a
MINIO_ROOT_USER=opencti
MINIO_ROOT_PASSWORD=admin123
RABBITMQ_DEFAULT_USER=opencti
RABBITMQ_DEFAULT_PASS=opencti
CONNECTOR_EXPORT_FILE_STIX_ID=d9d917c8b-abae-460a-9ebc-97b1551e70e6
CONNECTOR_EXPORT_FILE_CSV_ID=7ba187fb-fde8-4063-92b5-c3da346e6d47
CONNECTOR_EXPORT_FILE_TXT_ID=c4715d9c-bd64-4351-91db-33a8d728a58b
CONNECTOR_IMPORT_FILE_STIX_ID=72327164-0835-482b-b5d6-a5a3f76b845f
CONNECTOR_IMPORT_DOCUMENT_ID=c3970f8a-ce4b-4497-a3b1-20b7256f56f0
CONNECTOR_ANALYSIS_ID=4dfdf77c-ec11-4abe-bca7-fd997f79fa36
XTM_COMPOSER_ID=b215614c-7139-422e-b825-b28fd2a13a3a
SMTP_HOSTNAME=localhost
ELASTIC_MEMORY_SIZE=4G
COMPOSE_PROJECT_NAME=opencti
```

FIGURE 5.71 – Configuration des paramètres de MITRE ATTACK

Une fois le connecteur lancé, celui-ci récupère automatiquement les données MITRE ATTACK depuis les sources officielles et les injecte dans OpenCTI. Ces informations incluent notamment :

- Les tactiques (Initial Access, Execution, Persistence, etc.)
- Les techniques et sous-techniques
- Les relations entre techniques, menaces et indicateurs

## 5.5 Conclusion

La mise en place d'OpenCTI constitue une étape clé dans l'évolution de notre SOC vers une approche plus intelligente et orientée renseignement sur les menaces.

L'intégration du connecteur MITRE ATTACK a considérablement enrichi OpenCTI en apportant une base structurée et standardisée des tactiques, techniques et procédures utilisées par les attaquants.

# Conclusion Générale

À travers les trois chapitres de ce projet, nous avons mis en place une architecture complète de supervision et de sécurité des systèmes d'information, articulée autour des concepts NOC et SOC.

Dans un premier temps, la solution NOC a permis d'assurer la disponibilité, la performance et la stabilité de l'infrastructure grâce à une supervision proactive des ressources système, des services critiques et des hôtes Linux et Windows. L'utilisation de Zabbix a offert une visibilité en temps réel sur l'état de l'infrastructure et a permis de détecter rapidement les incidents pouvant impacter la continuité de service.

Dans un second temps, la mise en place du SOC a renforcé la posture de sécurité globale de l'environnement en intégrant une solution de collecte, d'analyse et de corrélation des logs basée sur la stack ELK, complétée par Suricata en tant que système de détection d'intrusion réseau. Cette approche a permis de détecter des activités uses, d'analyser les événements de sécurité et de valider l'efficacité des mécanismes de détection à travers des scénarios d'attaque concrets, tels que les tentatives de force brute.

Enfin, l'intégration d'OpenCTI a apporté une dimension stratégique à la solution en introduisant le renseignement sur les menaces (Threat Intelligence). Grâce à l'importation des données MITRE ATTACK, les alertes et incidents peuvent être replacés dans un contexte plus large, facilitant la compréhension des tactiques, techniques et procédures utilisées par les attaquants. Cette approche permet de transformer les alertes techniques en informations exploitables par les analystes SOC.

Plusieurs axes d'amélioration et d'évolution peuvent être envisagés afin d'augmenter la maturité de la solution déployée. Tout d'abord, une intégration plus poussée entre OpenCTI et ELK pourrait être mise en œuvre afin d'enrichir automatiquement les logs et alertes avec des indicateurs de compromission (IOC), des techniques MITRE associées et des informations de contexte issues de la Threat Intelligence. Cette corrélation renforcerait la précision des analyses et permettrait une priorisation plus efficace des alertes. Par ailleurs, l'ajout d'une solution de ticketing et de gestion des incidents telle qu'IRIS représenterait une évolution majeure vers un SOC opérationnel. L'intégration d'IRIS permettrait de transformer automatiquement les alertes critiques en incidents, de tracer les actions des analystes, de centraliser les investigations et d'améliorer la coordination des réponses aux incidents.