الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE BLIDA 1
Faculté de Technologie
Département d'Électronique

MEMOIRE DE MASTER
En Télécommunication

Spécialité : Réseaux & Télécommunications

RT13

THÈME :

Détection proactive des réseaux anonymes via l'analyse comportementale et signature

Réalisé par
Saidani Imene
Lazergui Rihab

Encadré par
Mr MEHDI Merouane

Juin 2025

Dédicace

Je dédie ce projet :

À mes **très chers parents**,

Aucune dédicace ne saurait exprimer tout le respect, l'amour éternel et la reconnaissance que je vous porte pour les sacrifices que vous avez consentis en faveur de mon instruction et de mon bien-être. Je vous remercie pour tout le soutien et l'amour que vous m'avez donnés depuis mon enfance, et j'espère que votre bénédiction m'accompagnera toujours.

À mes **grands-parents** bien-aimés (que Dieu ait leur âme), Bien que vous ne soyez plus parmi nous, votre amour et les valeurs que vous m'avez transmises continuent de guider mes pas.

À mes adorables **frères**, ainsi qu'à ma précieuse sœur **Djamila**, pour leur soutien.

À ma chère **belle-sœur** et à mes petits neveux, **Wassim** et **Djamel**.

À ma tante **Karima** et à sa famille, Merci pour votre tendresse et votre soutien constant.

À mon cher binôme, **Rihab**, pour son entente, sa patience et toutes les années que nous avons partagées ensemble.

À mes fidèles amies, **Ikram**, **Nadjet**, **Meriem** et **Hadil**, ainsi qu'à mes formidables collègues, **Hadir**, **Khadidja** et **Rania**, avec qui j'ai partagé des moments inoubliables tout au long de ce parcours. Votre soutien, votre esprit d'équipe et votre bonne humeur ont rendu cette expérience aussi enrichissante que mémorable.

À toute ma grande et petite famille.

IMENE

Dédicace

Je dédie ce projet :

À ma **chère mère**,

Ton amour inconditionnel, ton courage et les innombrables sacrifices que tu as faits pour moi sont à l'origine de tout ce que je suis devenu. Tu as souvent mis de côté ton propre confort pour m'assurer un avenir meilleur, et je ne pourrai jamais te remercier assez. Que ce travail soit un modeste témoignage de ma gratitude profonde.

À ma **sœur**,

Ton soutien constant, tes encouragements et ta présence bienveillante m'ont toujours donné la force d'avancer. Merci d'avoir toujours cru en moi, même quand moi-même j'en doutais.

À mon binôme et amie, **Imene**,

Merci pour ton écoute, ta patience et ta collaboration tout au long de ces années. Ton implication, ta générosité et notre belle entente ont grandement contribué à faire de ce parcours une expérience enrichissante et pleine de sens.

À mes amies de cœur, **Ihsane**, **Amina**, **Assia**, **Chaima**, **Nada** et **Souha**,

Merci pour votre amitié fidèle, vos mots réconfortants et votre présence précieuse à chaque étape de ce parcours. Votre bienveillance et vos encouragements ont été pour moi une source constante de force et de motivation.

À mes collègues exceptionnelles, **Hadir**, **Khadidja** et **Rania**,

Partager cette aventure avec vous a été un vrai privilège. Merci pour votre esprit d'équipe, votre soutien sans faille et tous ces moments de complicité qui ont illuminé notre chemin commun. Ce travail porte aussi la trace de notre belle collaboration.

À toute ma grande et petite famille.

RIHAB

Remerciement

Nous tenons tout d'abord à exprimer notre profonde gratitude à Allah, le Tout-Puissant, pour nous avoir accordé la santé, la force et la détermination nécessaires à la réalisation de ce projet.

Nous remercions chaleureusement notre promoteur, Monsieur **Mehdi Merouane**, pour ses précieux conseils, son suivi rigoureux et son soutien constant tout au long de ce travail. Ses encouragements et son expertise ont été essentiels à la réussite de ce projet.

Nous souhaitons également remercier l'ensemble des membres du jury pour le temps qu'ils nous consacrent et pour l'honneur qu'ils nous font en acceptant d'évaluer notre travail.

Nous n'oublions pas nos collègues, nos amis et nos familles, pour leur soutien inconditionnel, leurs encouragements et leur patience tout au long de cette aventure. Leur présence nous a donné la force de persévérer dans les moments les plus difficiles.

Enfin, nous remercions sincèrement toutes les personnes qui, de près ou de loin, nous ont apporté leur aide et ont contribué, par leurs conseils ou leur soutien moral, à la réussite de ce projet.

ملخص

يُعدّ استخدام الشبكات المجهولة مثل **تور و فريينت و جوندونيم** تهديدًا أمنيًا في البيئات المهنية، لما توفره من إمكانيات لتجاوز الرقابة وتسريب البيانات. يهدف هذا العمل إلى **كشف استخدام هذه الشبكات** اعتمادًا على أداة **سوريكاتا** ، من خلال **تحليل حركة الشبكة واستخلاص البصمات الرقمية** باستخدام **وايرشارك** ، ثم صياغة قواعد مخصصة للكشف الفوري. اكدت الاختبارات إمكانية التعرف على بعض تدفقات الشبكات المجهولة في شبكة محلية محاكاة.

Abstract

The use of anonymous networks such as **Tor Freenet,** and **JonDonym** poses risks to enterprise security by enabling data leakage and access to illegal content. This project proposes a **proactive detection method** using **Suricata**, based on **in-depth traffic analysis** with **Wireshark**. **Precise digital fingerprints** (TLS, ports, packet sizes) were extracted and converted into custom rules for real-time detection. Tests on a simulated network confirmed the effectiveness of the approach.

Résumé

L'utilisation de réseaux anonymes tels que **Tor**, **Freenet** et **JonDonym** peut poser des risques pour la sécurité des entreprises, notamment en facilitant la fuite de données ou l'accès à des contenus illégaux. Ce projet vise à **détecter l'usage de ces réseaux anonymes** dans un environnement professionnel à l'aide de **Suricata**. L'approche adoptée repose sur une **analyse approfondie du trafic réseau** avec **Wireshark**. Des **empreintes numériques précises** ont été extraites (TLS, ports, tailles de paquets) et traduites en règles personnalisées pour une détection en temps réel. Les tests menés sur un réseau simulé confirment l'efficacité de cette approche.

ACK Acknowledgment

ACL Access Control List

AES Advanced Encryption Standard

CBC Cipher Block Chaining

CHK Content Hash Key

DDoS Distributed Denial of Service

DHT Distributed Hash Table

DoS Denial of Service

FFS Freenet First Steps

FNP Freenet Protocol

FTP File Transfer Protocol

HIDS Host-based Intrusion Detection System

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

I2P Invisible Internet Project

IDS Intrusion Detection System

IP Internet Protocol

IPS Intrusion Prevention System

JAP Java Anon Proxy

JSON JavaScript Object Notation

KSK Keyword Signed Key

NAT Network Address Translation

netDB Network Database

NIDS Network-based Intrusion Detection System

NTCP New Input/Output Transmission Control Protocol

OISF Open Information Security Foundation

PCap Packet Capture

RSA Rivest–Shamir–Adleman

RTT Round-Trip-Time

SHA Secure Hash Algorithm

SIEM Security Information and Event Management

SSK Signed Subspace Key

SSU Secure Semireliable UDP

SSL Secure Sockets Layer

SYN Synchronize

TCP Transmission Control Protocol

TLS Transport Layer Security

TOR The Onion Router

TTL Time To Live

UDP User Datagram Protocol

UID User Identifier

URL Uniform Resource Locator

USK Updateable Subspace Key

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Avec le développement des technologies digitales et d'Internet, la problématique de la protection des données et de l'anonymat sur le web est devenue cruciale. Des technologies comme les réseaux anonymes, notamment Tor, JonDonym ou Freenet permettent aux utilisateurs de masquer leur identité en ligne. Bien que ces outils soient légitimes dans certains contextes (comme la lutte contre la censure), leur utilisation dans un cadre professionnel pose de graves questions de sécurité. En entreprise, ces réseaux peuvent être détournés pour contourner les politiques de filtrage, échapper la surveillance interne, ou encore accéder au Dark Web. Cette situation rend difficile pour les administrateurs réseau d'identifier et de bloquer ce type de trafic, qui ressemble souvent à du trafic web classique.

Dans ce contexte, la question suivante se pose : Comment détecter l'utilisation de réseaux anonymes dans un réseau d'entreprise ?

Afin de répondre à cette problématique, ce mémoire vise à détecter l'usage de réseaux anonymes en entreprise en s'appuyant sur leurs signatures spécifiques extraites à partir de l'analyse du trafic réseau. Pour cela, une démarche structurée a été adoptée dans les étapes suivantes :

- La réalisation d'une infrastructure réseau client-serveur qui intègre un serveur proxy simulant les restrictions.
- La génération de trafic anonyme en utilisant Tor, Freenet et JonDonym pour contourner les restrictions du proxy.
- Analyse des paquets réseau capturés avec l'analyseur de paquets Wireshark pour comparer le trafic classique généré par Firefox au trafic anonyme, afin d'identifier les signatures caracteristiques des réseaux anonymes.
- L'extraction de signatures spécifiques propres à chaque réseau anonyme.
- Enfin, l'implémentation de ces signatures sous forme de règles dans un système de détection d'intrusion (IDS), notamment Suricata, afin de détecter automatiquement et en temps réel les communications anonymes.

Ce mémoire est structuré en trois chapitres :

— Le premier chapitre introduit les réseaux anonymes, leur fonctionnement, leur usages légitimes et illégitimes, ainsi que leur risques.

— Le deuxième chapitre est consacré à l'analyse du trafic réseau et à l'extraction de signatures caractéristiques des réseaux anonymes.

— Le troisième chapitre décrit l'implémentation de notre système de détection, les tests réalisés et les résultats obtenus.

Enfin, une conclusion générale viendra synthétiser les résultats obtenus et les limites rencontrées.

Chapitre 1

Introduction aux Réseaux Anonymes et leurs Risques

1.1 Introduction

Il est devenu difficile d'utiliser Internet sans laisser de traces. Chaque site consulté, chaque recherche effectuée ou chaque fichier partagé peut être surveillé, analysé et parfois exploité à notre insu. Face à cette situation, de plus en plus d'utilisateurs cherchent des moyens de préserver leur anonymat en ligne.

Des outils comme Tor, I2P, JonDonym ou Freenet offrent une solution pour ceux qui veulent protéger leur vie privée et éviter la surveillance en ligne. Cependant, ces technologies ne sont pas toujours utilisées à des fins positives. Dans un cadre professionnel, elles peuvent aussi être détournées et représenter un danger pour les entreprises.

Certains employés pourraient les utiliser pour accéder au Dark Web, mettant ainsi en péril la sécurité des réseaux internes et augmentant le risque de fuite de données sensibles.

1.2 Anonymat, Confidentialité et Pseudonymat

Beaucoup de personnes confondent l'anonymat, la confidentialité et le pseudonymat, pensant qu'ils désignent la même chose. Bien qu'ils soient liés à la protection de l'identité et des données en ligne, ils ont des significations distinctes.

1.2.1 La confidentialité

La confidentialité en ligne consiste à protéger vos informations personnelles lorsque vous surfez sur Internet. Cela concerne votre nom, votre adresse et votre numéro de téléphone, mais aussi ce que vous faites en ligne, telles que les sites que vous visitez et les recherches que vous effectuez. [1]

La confidentialité vise à empêcher que vos données sensibles ne soient accessibles par des personnes non autorisées. [1]

1.2.2 L'anonymat

L'anonymat sur Internet consiste à cacher votre véritable identité, notamment votre adresse IP, lorsque vous naviguez en ligne. Lorsque vous êtes anonyme, votre identité ne peut pas être facilement retracée par les plateformes et services que vous utilisez. [1]

1.2.3 Pseudonymat

Le pseudonymat est une méthode intermédiaire entre la confidentialité et l'anonymat. Lorsque vous utilisez un pseudonyme en ligne, vous n'utilisez pas votre vrai nom mais un nom fictif pour interagir sur Internet. Les pseudonymes sont souvent utilisés sur les réseaux sociaux, les forums et d'autre espaces de discussion. Contrairement à l'anonymat, votre pseudo peut être associé à un compte spécifique, ce qui signifie que les administrateurs du site ou les autorités peuvent accéder à certains informations vous concernant. [1]

1.3 Les profondeurs de l'internet

Internet est un vaste réseau , bien qu'une seule partie limitée soit accessible à la majorité des utilisateurs. Alors que nous naviguons sur des sites web et effectuons des recherches sur ce qu'on appelle le surface web, il existe toute une partie cachée de l'internet souvent segmentée en web profond (Deep Web) et en web sombre (Dark Web), comme illustré dans la **figure 1.1**. [2]

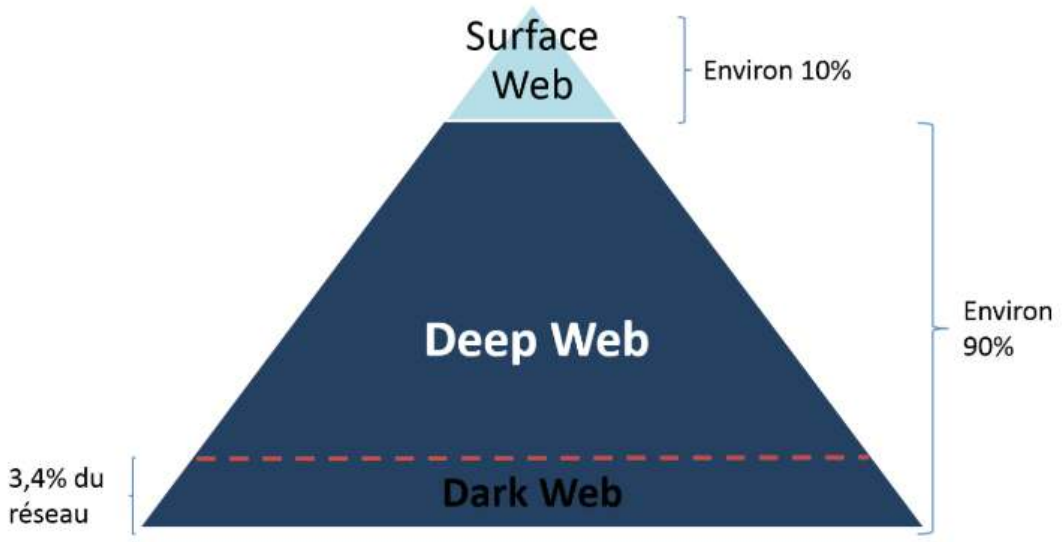


FIGURE 1.1 – Les profondeurs de l'Internet. [2]

1.3.1 Le Surface Web

Le Surface Web, parfois appelé le "web ouvert", est la couche d'Internet la plus familière pour la majorité des utilisateurs. On peut le considérer comme la partie accessible et publique d'internet, composée de sites web que les moteurs de recherche classiques (Google, Yahoo, Bing...) peuvent explorer et indexer. [3]

1.3.2 Le Deep Web

En descendant sous le Surface Web, nous entrons dans le Deep Web. Contrairement au contenu ouvert et accessible via les moteurs de recherche standards, le Deep Web comprend des pages qui ne sont ni indexées ni consultables par des moteurs de recherche standards. Cette couche englobe des bases de données privées, des sites protégées par identifiant, des dossiers académiques, des données, etc.

Bien qu'il soit souvent perçu comme mystérieux, la majorité du Deep Web est légale et sécurisée. Il est principalement protégé afin de préserver la confidentialité des utilisateurs et la sécurité des données sensibles. [3]

1.3.3 Le Dark Web

À la couche la plus profonde de l'Internet se trouve le Dark Web, une partie du Deep Web dissimulée et accessible uniquement via des outils spécifiques, tels que le

navigateur Tor. Contrairement au Deep Web, qui contient des données protégées, le Dark Web regroupe des sites au contenu chiffré, mettant l'accent sur l'anonymat et la confidentialité. [3]

Dans le Dark Web on trouve à la fois du contenu légal et illégal.

Selon l'enquête menée par les chercheurs **Daniel Moore** et **Thomas Rid**, publiée dans leur livre *Cryptopolitik and the Darknet*, où il ont analysé **5205** sites actifs sur le Dark Web et ont constaté que **2723** d'entre eux, soit environ la moitié du Dark Web, sont occupés par des contenus illicites. Parmi ces contenus on trouve : des activités financières illicites, la contrefaçon de médicaments, le trafic d'armes et de drogues, la fausse monnaie et même des bases de données volées en vente sur le Dark Web. [4]

1.4 Les réseaux anonymes

1.4.1 Définition

Les réseaux anonymes sont des systèmes conçus pour permettre des communications sans révéler l'identité des utilisateurs. Ils fonctionnent grâce à des applications décentralisées où chaque participant reste anonyme. L'objectif principal de ces outils est d'empêcher toute surveillance ou analyse du trafic en ligne.

Ils sont généralement développés en open source, ce qui les rend accessibles à tous. [5]

1.4.2 Les outils d'anonymat

L'anonymat en ligne est rendu possible grâce à divers outils, parmi ces outils on trouve : Tor, I2P, JonDonym et Freenet. Chacun de ces systèmes repose sur des principes cryptographiques avancés et des techniques de routage spécifiques pour assurer la confidentialité des utilisateurs.

1.4.2.1 Tor

Tor, acronyme de **The Onion Router**, est un logiciel open source permettant de naviguer anonymement sur Internet. Initialement développé par l'armée américaine, plus précisément par la **Navy**. Tor avait pour but de masquer les adresses IP militaires pour sécuriser les communications sensibles. Depuis, Tor est devenu un outil accessible à tous, utilisé par des millions de personnes dans le

monde pour protéger leur vie privée et contourner la censure. Les éléments graphiques officiels du projet Tor, y compris son logo, sont accessibles à l'adresse suivante : https://styleguide.torproject.org/brand-assets/.

Il n'est pas illégal d'utiliser Tor, mais s'en servir pour accéder au Dark Web ou à des plateformes de vente illégales est totalement interdit par la loi [6].

1.4.2.2 I2P : The Invesible Internet Project

Le projet Internet Invisible (I2P) est un réseau anonyme décentralisé, développé en Java, avec des principes similaires à Tor, mais conçu dés le départ comme un réseau sombre autonome. Le logo du projet I2P, est disponible à l'adresse suivante : https://geti2p.net/en/about/media.

I2P est en réalité une Internet dans l'Internet. Une fois connectée, vous pouvez envoyer des e-mails, naviguer sur des sites web, accéder à des epsites qui sont des sites I2P internes, utiliser des blogs et des forums, participer à des discussions anonymes en temps réel, et bien plus encore. [7]

1.4.2.3 Freenet

Freenet est un logiciel décentralisé qui vous permet de partager anonymement des fichiers, parcourir et publier des freesites, qui sont des sites internes seulement accessibles via Freenet. [8] Le logo officiel du projet Freenet est consultable sur le site officiel : https://staging.freenetproject.org/fr/index.html En 2023 le projet Freenet a été renommé Hyphanet. [9]

1.4.2.4 JonDonym

JonDonym, anciennement connu sous le nom d'AN.ON, est un système d'anonymisation reposant sur le principe du chiffrement multiple en couches, de la distribution et du traitement du trafic. [10]

1.5 Fonctionnement general

1.5.1 Tor

Tor fonctionne grâce à un circuit bidirectionnel chiffré en multicouches, basé sur la technique du routage en oignon. Lorsqu'un utilisateur se connecte à Tor, un circuit est crée a travers généralement trois relais (nœuds) choisis aléatoirement, comme illustré dans la **figure 1.2**. Ce circuit comprend : [11]

— **Nœud d'entrée (Guard)** : Souvent appelé le nœud de garde, est le premier nœud auquel le client Tor se connecte. Le nœud d'entrée est capable de voir l'adresse IP de l'utilisateur, mais il est incapable de voir la destination finale.
— **Nœud intermédiaire (Relay)** : Il agit comme un relais de transition dans le circuit, il ne possède aucune information concernant l'origine initiale de la communication ni sa destination finale.
— **Nœud de sortie(Exit)** : Le nœud de sortie est le point où le trafic web quitte le réseau Tor, ce dernier est capable de savoir la destination externe à laquelle la requête est adressée.

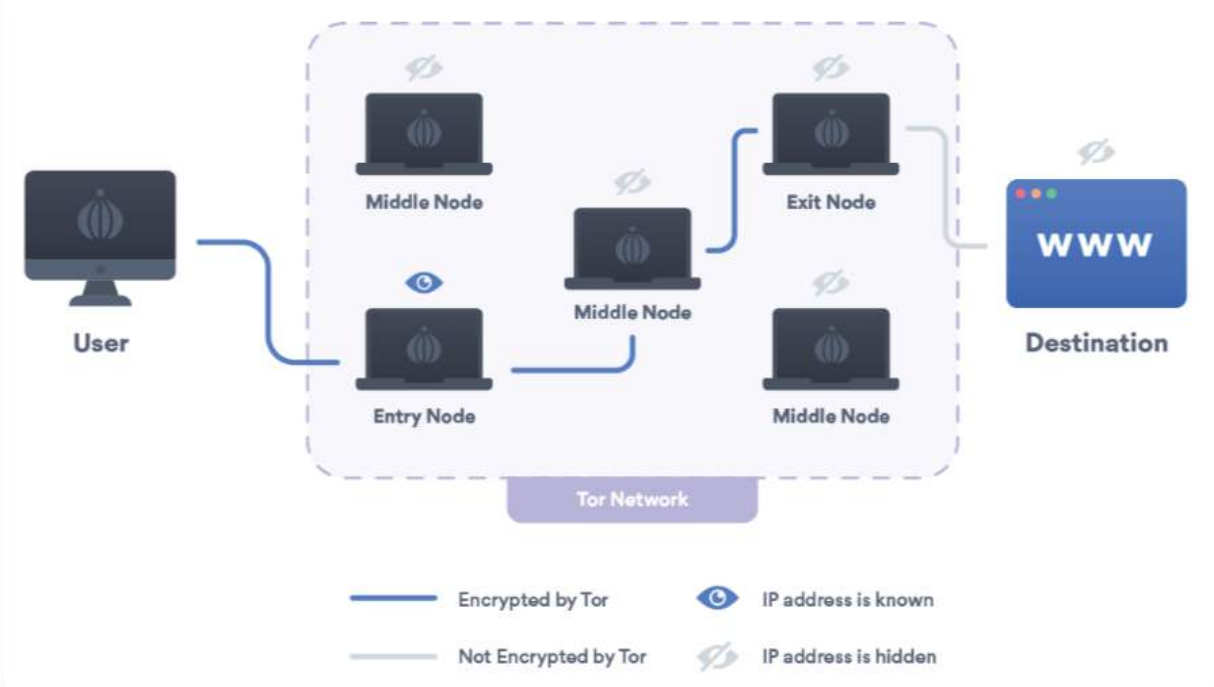


FIGURE 1.2 – Circuit Tor. [12]

1.5.1.1 Routage en oignon (Onion Routing)

L'idée principale du routage en oignon est d'encapsuler les données dans plusieurs couches de chiffrement, comme les couches d'un oignon. Chaque relais dans le réseau possède une paire de clés : une clé publique et une clé privée.

Lorsqu'un utilisateur envoie une requête via le réseau Tor, le message est d'abord chiffré avec la clé publique du nœud de sortie, puis une deuxième fois avec la clé publique du relais intermédiaire, et enfin une troisième fois avec celle du nœud d'entrée.

Chaque relais avec sa propre clé privé, ne peut déchiffrer que la couche qui lui destinée, sans jamais connaître l'intégralité du chemin ni le contenu du message,

comme illustré dans la **figure 1.3.** [13]

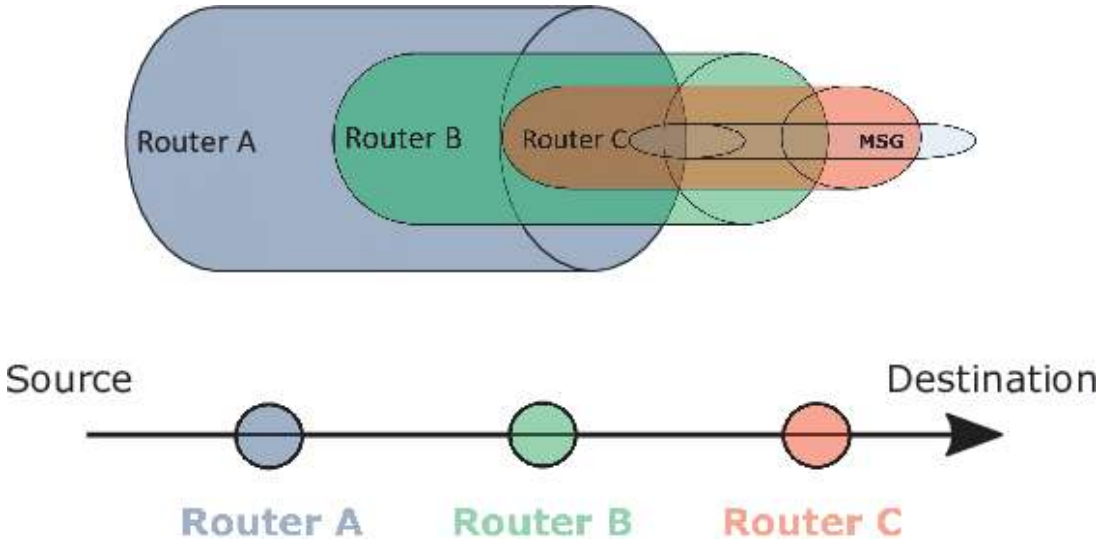


FIGURE 1.3 – Onion Routing. [14]

1.5.1.2 Services cachés

Les "services onion" (également appelés "services cachés") sont des sites web auxquels on ne peut accéder qu'au moyen du navigateur Tor. Ces sites web ont un long nom de domaine généré de manière aléatoire et se terminant par `.onion`.

La connexion à un service onion dans Tor fonctionne de manière très similaire à la connexion à un service de surface, mais votre trafic est acheminé à travers un total de **six nœuds** avant d'atteindre le serveur de destination. Cependant, comme auparavant, seuls trois de ces nœuds contribuent à votre anonymat, les trois autres nœuds protègent l'anonymat du service onion, en cachant la véritable IP et la localisation du site web de la même manière que le navigateur Tor cache les vôtres. [11]

1.5.1.3 Les Algorithmes de Chiffrement

— **Diffie-Hellman** : Deffie-Hellman génère une clé de session appelée "clé secrète", qui est une clé symétrique temporaire partagée entre le client Tor et chaque relais, sans être directement échangée. Cette clé est utilisée pour chiffrer les données avec AES-128 et change a chaque nouveau circuit pour renforcer la sécurité.

— **RSA (Rivest-Shamir-Adleman)** : est un algorithme asymétrique qui utilise une paire de clé : une clé publique pour chiffrer la clé secrète et une clé privée,

gardée par le relais, pour la déchiffrer. La clé privée RSA n'est jamais envoyée.

— **AES (Advanced-Encryption-Standard)** : AES-128 est un algorithme de chiffrement symétrique qui utilise la clé de session pour chiffrer et déchiffrer toutes les données échangées entre le client Tor et les relais.

— **SHA(Secure-Hash-Algorithm)** : est une fonction de hachage cryptographique utilisée dans Tor pour vérifier l'intégrité du message, le client Tor crée une empreinte SHA-1 du message avant de l'envoyer. Chaque relais vérifie le message qu'il reçois correspond bien a cette empreinte. [13]

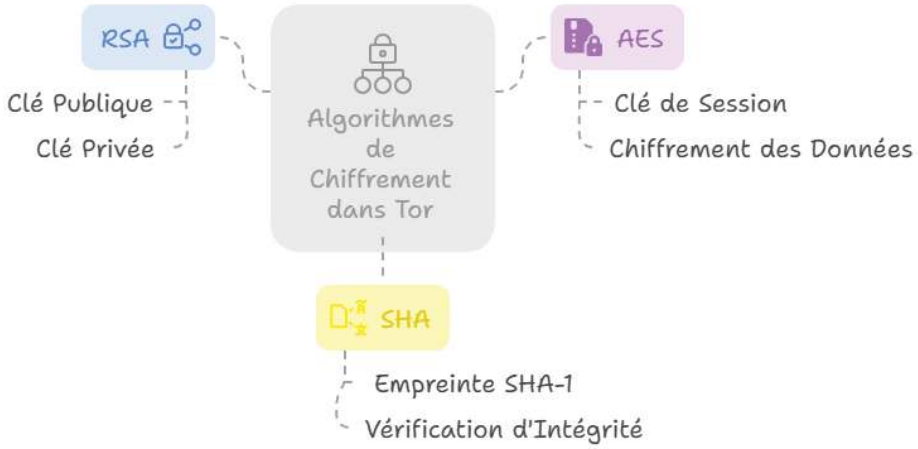


FIGURE 1.4 – Algorithmes de Chiffrement dans Tor. [13]

Tor combine la sécurité et la rapidité en utilisant AES et RSA ensemble, RSA est utilisé pour sécuriser la clé de session uniquement, ce qui évite de ralentir la connexion. AES est utilisé pour chiffrer les données, car il est très rapide et capable de gérer de grande quantité de données. [15]

FIGURE 1.5 – Utilisation conjointe de RSA et AES. [15]

1.5.2 I2P

1.5.2.1 Les tunnels

Un tunnel I2P est un ensemble de routeurs dans un ordre spécifique utilisé pour transmettre un message. Il se compose de :

— Passerelle (Gateway) : le premier routeur du tunnel.
— Participant : le routeur au milieu.
— Point de sortie (Endpoint) : le dernier routeur de la chaîne.

Il existe des tunnels sortants (Outbound tunnel) pour l'envoie et des tunnels entrants (Inbound tunnel) pour la réception des messages. [16]

1.5.2.2 Création des tunnels

— **Tunnel Sortant** :

1. Le routeur qui veut créer un tunnel recupére une liste de routeurs disponibles (routerInfos) depuis la NetDB.
2. Il sélectionne un nombre de routeurs généralement deux.
3. Il crée des messages de création de tunnel pour chaque participant du tunnel, lui même agissant comme passerelle.
4. Il envoie le message au prochain routeur qui est le participant lui même transmet le message au routeur suivant (point de sortie).
5. Le point de sortie envoie un message ACK au créateur du tunnel. Si tout est réussit, le tunnel est prêt a envoyer des messages à d'autre utilisateurs, comme illustré dans **la figure 1.6**.

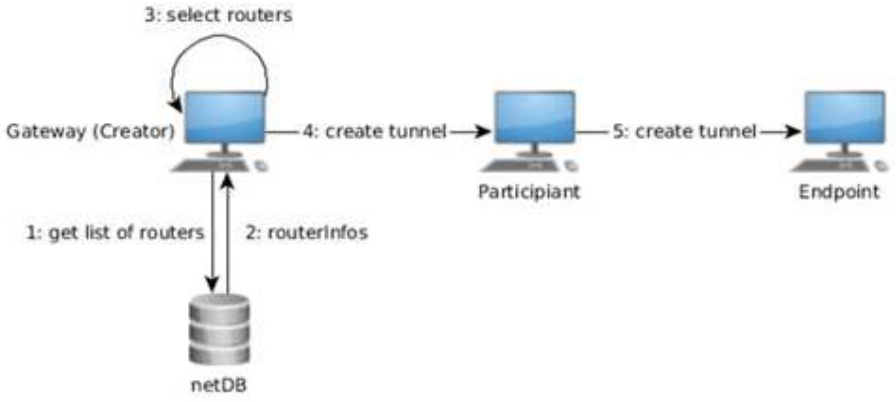


Figure 1.6 – Tunnel Sortant. [16]

— **Tunnel Entrant** :
La création d'un tunnel entrant suit le même processus sauf que le créateur agit comme le point de sortie et le dernier routeur devient la passerelle, qui agit comme adresse cible pour les messages destines au récepteur. Un Tunnel-ID unique est utilisé afin que la passerelle puisse relier les paquets entrants au destinataire prévu et les transférer. Comme dernière étape, le créateur publie l'adresse de la passerelle dans le NetBD avec le Tunnel-ID, comme le montre la **figure 1.7**. [16]

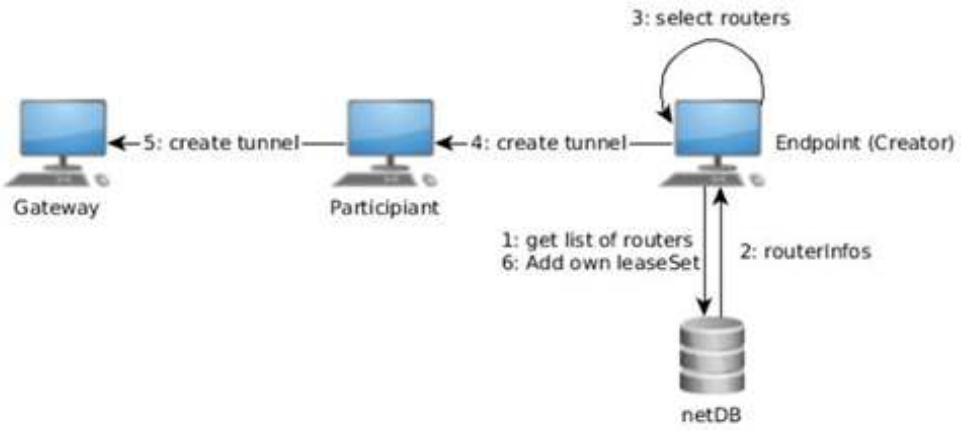


FIGURE 1.7 – Tunnel Entrant. [16]

1.5.2.3 Chiffrement et routage

I2P combine deux couches de chiffrement pour assurer l'anonymat.

— **Routage en ail (Garlic Routing)** : le chiffrement en ail aussi appelée routage en ail, fait référence au projet Onion Routing. Il encapsule les messages (Cloves) dans une grosse enveloppe chiffrée (Garlic Message), chaque message est chiffré individuellement avec AES-256. Ensuite le Garlic message est chiffré avec une autre couche, utilisant le chiffrement asymétrique ElGamal pour sécuriser la clé de session qui est générée par le client I2P (comme RSA dans Tor) et le chiffrement symétrique AES pour chiffrer le message d'ail, masquant ainsi la destination et les informations de contrôle. [16]

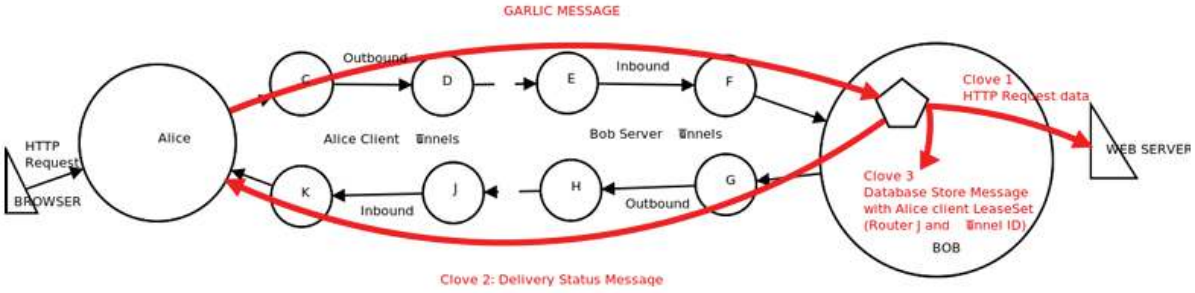


FIGURE 1.8 – Routage en Ail. [16]

— **Chiffrement des tunnels** : les tunnels ajoutent une couche supplémentaire AES-256/CBC pour chaque routeur traversé, pour protéger le chemin. [16]

1.5.2.4 NetDB

Dans la NetDB on trouve, le **RouterInfo** qui contient les informations sur un routeurs (Adresse IP, ports ...) et le **LeaseSet** qui est l'équivalent d'une adresse pour un service caché I2P. [16]

1.5.2.5 Protocoles de Transport

1. **SSU (Secure Semireliable UDP)** : est un protocole développé basé sur UDP. Il permet des retransmissions des paquets perdus, mais seulement un nombre limité de fois pour éviter de surcharger le réseau, donc semi-fiable. Il gère des tâches comme, comme la détection des adresses IP et la gestion des configuration NAT et des pare-feu.
2. **NTCP (NIO TCP)** : signifie New I/O TCP, il s'appuie sur Java TCP transport pour transférer les paquets de manière fiable. [16]

I2P combine les deux protocoles ensemble pour garantir une communication fluide et résiliente.

1.5.3 Freenet

Freenet repose sur une architecture décentralisée et anonyme ou chaque utilisateur, appelé nœud est connecté à une vingtaine d'autres utilisateurs.

Lorsqu'un utilisateur envoie un fichier sur Freenet, celui-ci est d'abord découpé en morceaux "fragments" puis chiffré. Ces fragments sont ensuite distribués et stockés de

manière décentralisée sur l'espace disque d'autres utilisateurs du réseau, qui agissent comme des nœuds. La **figure 1.9** illustre la structure de ces noeuds. [5]

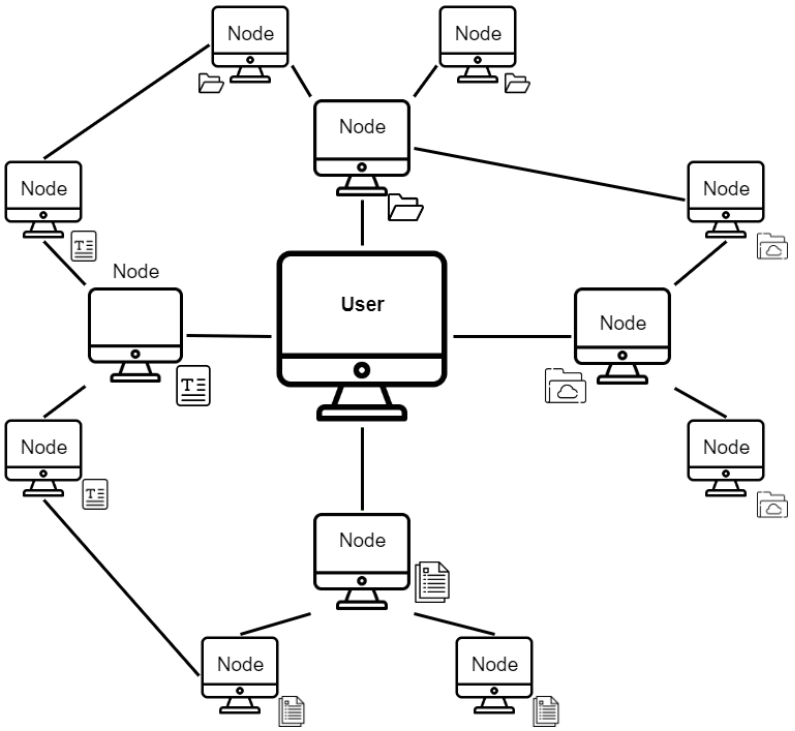


FIGURE 1.9 – Structure des nœuds dans Freenet. [17]

Lorsqu'un utilisateur souhaite accéder à un fichier, il ne sais pas où le fichier est situé. Il envoie donc une requête à ses voisins, s'ils ne possèdent pas le fichier, transmettent la requête à leurs propres voisins et ainsi de suite, formant une recherche en cascade. Dés qu'un nœud trouve le fichier ou un de ses fragments, il le renvoi en remontant la chaîne jusqu'à l'utilisateur initial de la requête, comme illustré dans la **figure 1.10**. [18]

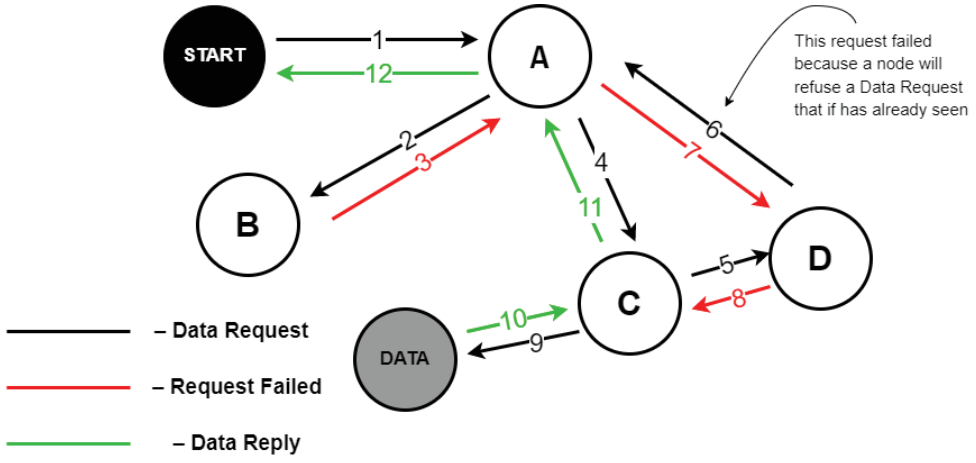


FIGURE 1.10 – Recherche en Cascade. [17]

La particularité de Freenet est que pendant cette "remontée", chaque utilisateur de la chaîne va garder ce fichier en cache. Le fichier se démultiplie. [18]

Techniquement, Freenet fonctionne indépendamment du protocole de transport : bien que les premières versions utilisaient TCP, les versions récentes utilisent UDP, rendant son trafic moins identifiable. La communication entre les nœuds utilise un protocole spécifique : le Freenet Protocol (FNP).

Chaque nœud dispose d'une identité unique, comprenant une clé publique et un port UDP, qui sont partagés avec les nœuds voisins pour établir des canaux sécurisés. Les messages échangés (requêtes, données, probes) incluent un identifiant unique (UID) permettant d'éviter les boucles dans le routage.

Le réseau utilise un modèle de table de hachage distribuée (DHT). Chaque nœud possède une position (location) aléatoire dans un espace circulaire [0,1), et les contenus sont stockés dans les nœuds les plus proches (en termes de position) de la clé associée au contenu. [19]

Freenet repose sur plusieurs types de clés cryptographiques pour organiser les contenus :

- **CHK (Content Hash Key)** : basée sur le hachage du contenu, elle permet la vérification d'intégrité.
- **KSK (Keyword Signed Key)** : dérivée d'une chaîne lisible, elle permet des recherches par mot-clé.
- **SSK (Signed Subspace Key)** : utilisée pour les contenus évolutifs, avec une signature de l'auteur.
- **USK (Updatable Subspace Key)** : version améliorée de la SSK, permettant d'obtenir automatiquement la dernière version d'un contenu. [19]

1.5.3.1 Darknet et Opennet

Le réseau Freenet dispose deux méthodes de fonctionnement :

- **Le mode Opennet** : dans ce mode, chaque utilisateur se connecte à d'autre utilisateurs inconnus. C'est le mode le plus vulnérable.
- **Le mode Darknet** : les utilisateurs ne se connectent qu'a des utilisateurs qu'ils ont explicitement indiqués, seuls les contacts de confiance peuvent établir une connexion, ce qui renforce l'anonymat. [18]

1.5.4 JonDonym

Le service d'anonymisation JonDonym repose sur le principe de chiffrement multiple. Il se compose de plusieurs cascades de mix sélectionnables par l'utilisateur. Une cascade est constituée de deux ou trois serveurs de mixage avec un chiffrement séparé pour chacun (**RSA-1024** pour le chiffrement asymétrique et **AES-128** pour le chiffrement symétrique). Ces serveurs exploités par des organisations indépendantes et non reliées entre elles, ou par des particuliers, qui publient tous leurs identité.

Sans JonDonym, l'adresse IP de l'utilisateur est directement exposée aux sites web, aux fournisseurs d'accès et au traqueurs.

Avec JonDonym, le trafic est protégé, car il est acheminé via la cascade où chaque serveur ne connaît qu'une partie des informations. De plus, tous les utilisateurs partagent la même adresse IP finale, ce qui empêche toute corrélation.

JonDonym s'appuie sur deux composants principaux pour garantir une sécurité et un anonymat maximum :

JonDo (JAP) : JonDo est un client proxy, il gère le chiffrement et le transfert du trafic à travers la cascade de serveurs mix.

JonDoFox (JonDoBrowser) : une version modifiée de Mozilla Firefox, spécialement configurée pour la navigation anonyme. Il envoie tout le trafic web via le client JonDo. [10]

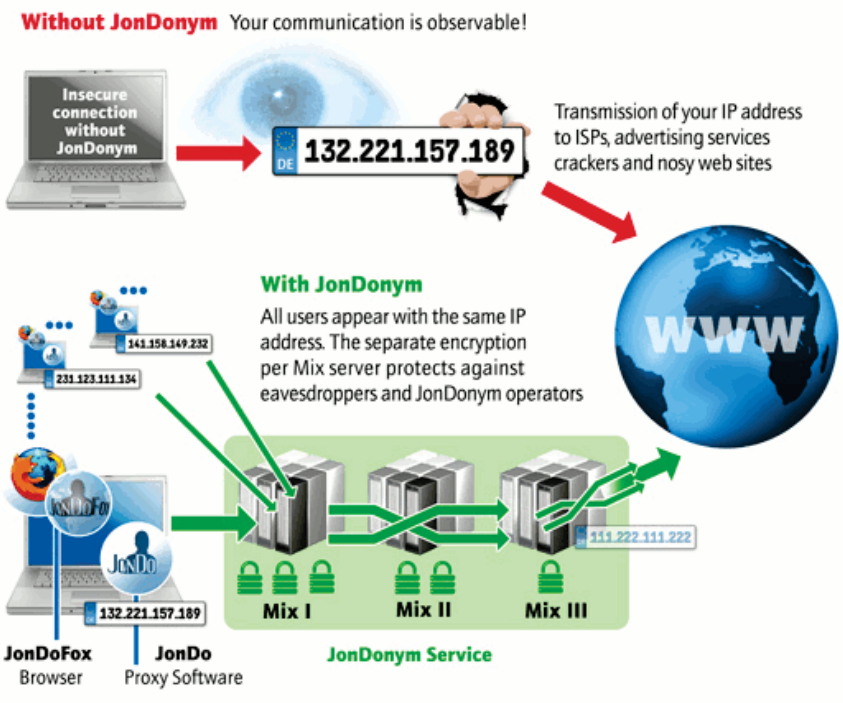


Figure 1.11 – Fonctionnement de JonDonym. [10]

1.6 Usages légitimes et illégitimes des réseaux anonymes

1.6.1 Usages légitimes

1. **Protection de la vie privée** : l'anonymat permet aux individus de garder leurs informations personnelles privées et de réduire le risque de vol d'identité, de surveillance en ligne et de suivi non désiré.
2. **Liberté d'Expression** : l'anonymat permet au Internautes d'exprimer leurs opinions et leurs idées sans craindre de sanction ou de répercussion. Cela est important dans les régions où la liberté d'expression est restreinte.
3. **Protection des lanceurs d'alerte** : l'anonymat encourage la révélation de la corruption, des abus de pouvoir ou des violations des droits humains tout en offrant une protection aux lanceurs d'alerte. [20]

1.6.2 Usages illégitimes

La commission européenne recense trois catégories d'activités criminelles pouvant être commis sous le couvert de l'anonymat : [21]

1. **Cybercriminalité** : les cybercriminels utilisent l'anonymat pour exploiter des attaques par déni de service (DoS), qui consiste à saturer ou bloquer les systèmes informatiques, rendant ainsi les services inaccessibles.
2. **Fraude en ligne** : il s'agit d'escroqueries menées sur internet.
3. **Contenus illégaux** : l'anonymat peut être détourné pour diffuser des contenus nuisibles (pédophilie, racisme ou la xénophobie).

Parmi ces usages illégitimes, plusieurs s'appuient sur les possibilités offertes par le Dark Web, déjà abordé dans la **section 1.3.3**. Ce denier, étroitement lié au réseau **Tor**, facilite l'accès à des contenus et activités illicites. [22] . Un exemple concret est détaillé dans la **section 1.7.4**.

1.7 Typologie des risques liés aux réseaux anonymes

1.7.1 Risques techniques et vulnérabilité de réseau

Risques liés aux nœuds de sortie et aux failles de sécurité : dans le cas de Tor, les nœuds de sorties peuvent être contrôlés par des attaquants, leur permettant ainsi d'espionner le trafic non chiffré. De plus, des attaques tels que l'attaque *Sybil*, découverte en 2014, peuvent désanonymiser les utilisateurs. [23]

Fichiers non modérés : dans Freenet, la publication des fichiers n'est pas modérée, ce qui permet la diffusion de logiciels malveillants ou de contenus dangereux, mettant ainsi en danger la sécurité des utilisateurs.

1.7.2 Risques pour les entreprises

Contournement des politiques de sécurité : l'utilisation de réseaux anonymes par des employés empêche les systèmes de surveillance de détecter leur activité, ce qui facilite l'accès à des sites illicites ou la fuite d'informations sensibles de l'entreprise.

Téléchargement de contenus infectés : un employé utilisant un réseau anonyme peut télécharger des fichiers malveillants, introduisant des malwares dans le réseau de l'entreprise. Cela peut compromettre l'ensemble du système informatique.

Attaque par Déni de Service Distribué (DDoS) : lorsqu'un employé commence à utiliser Tor, les serveurs de l'organisation peuvent se retrouver à relayer une grande quantité de trafic Tor, cela expose en permanence l'organisation à une attaque DDoS, visant à rendre un service en ligne indisponible en le submergeant de trafic provenant de multiples sources. [24]

1.7.3 Risques juridiques et éthiques

Si un employé utilise des réseaux anonymes pour accéder à du contenu illégal via le réseau de l'entreprise, celle-ci peut être tenue partiellement responsable, surtout si aucune politique n'est mise en place pour bloquer ce type d'usage.

1.7.4 Risques lié à l'utilisation du Dark Web

En 2021, un groupe de cybercriminels russes, **REvil**, également connu sous le nom de **Sodinokibi**, a été arrêté après avoir mené de nombreuses attaques par ransomware contre des entreprises à l'échelle mondiale, notamment **Quanta**, un fournisseur d'Apple, et **Kesaya**, une entreprise spécialisé dans la gestion informatique. Les membres du

groupe menaçaient de publier ou de vendre les données sensibles qu'ils avaient volées sur des plateformes du Dark Web, en utilisant des services cachés accessible via le réseau Tor. [25]

1.8 Conclusion

Dans ce premier chapitre, nous avons présenté les différentes parties de l'Internet, expliqué les notions d'anonymat. Nous avons aussi vu comment ces réseaux fonctionnent, quels sont les risques liés à leur utilisation et comment ils peuvent être utilisés.

Ce chapitre nous a permis de mieux comprendre le rôle et le fonctionnement des réseaux anonymes, ainsi que les dangers qu'ils peuvent représenter.

Le prochain chapitre sera consacré à l'analyse de ces réseaux afin d'extraire leurs empreintes caractéristiques permettant de les détecter.

Chapitre 2

Analyse des Réseaux Anonymes et Extraction des Signatures

2.1 Introduction

Dans le cadre d'une entreprise, l'usage de réseaux anonymes comme Tor, JonDonym et Freenet offre à certains utilisateurs la possibilité de contourner les règles et restrictions existantes. Ces réseaux peuvent également être utilisés pour faire sortir des données sensibles sans être détectés. Cela représente un risque de sécurité important pour l'entreprise. C'est pour cette raison que beaucoup d'entreprises cherchent à bloquer ce type de trafic pour éviter toute fuite d'information ou activité non autorisée.

Cependant, détecter l'utilisation de Tor, JonDonym ou Freenet sur les réseaux d'entreprises n'est pas facile. Car leur trafic crypté ressemble à celui du web classique. Afin de les identifier, il est nécessaire de comparer les deux types de trafic pour extraire des signatures réseau spécifiques qui peuvent être utilisées pour les détecter.

Dans ce chapitre, nous allons présenter notre architecture de travail dont le but est de simuler un réseau d'entreprise. Par la suite nous effectuerons une capture et une analyse du trafic afin d'identifier l'utilisation de Tor, JonDonym et Freenet au sein de notre réseau. Cette analyse reposera principalement sur l'utilisation de l'outil Wireshark pour extraire les paquets contenant des signatures caractéristiques de ces réseaux anonymes.

2.2 Présentation de notre Projet

Ce projet vise à étudier et analyser le comportement des réseaux anonymes lorsqu'ils sont utilisés pour contourner des restrictions d'accès dans une entreprise. L'objectif principal est de comprendre comment ces outils fonctionnent sur le plan réseau, et surtout

comment détecter leur utilisation via l'analyse des signatures réseau qu'ils laissent.

Une approche structurée en quatre phases a été définie pour la réalisation de ce travail :

1. **Étude Théorique** : cette phase a consisté à comprendre le fonctionnement en détail des réseaux anonymes.
2. **Phase de Capture** : durant cette phase, l'objectif principal est de capturer le trafic réseau à l'aide de l'outil **Wireshark** afin de collecter des données nécessaire à l'analyse.
3. **Phase d'Analyse** : cette phase permet d'analyser les paquets capturés afin d'identifier les caractéristiques spécifiques de chaque réseau anonyme :
   — Protocoles utilisés
   — Ports utilisés
   — Type de paquets échangés
   — Schéma de chiffrement TLS/SSL
4. **Phase de Détection** : enfin, cette phase consiste à implémenter des règles basées sur les signatures extraites dans le système de détection d'intrusion **Suricata**, afin de détecter l'utilisation des réseaux anonymes.

2.3 Environnement de Travail

Pour simuler un réseau d'entreprise et étudier l'utilisation de réseaux anonymes, une architecture client-serveur a été mise en place. Cette infrastructure permet de capturer, d'analyser et de détecter le trafic réseau généré par l'utilisation de Tor, JonDonym et Freenet.

2.3.1 Architecture client-serveur

Le schéma suivant illustre l'architecture mise en place :

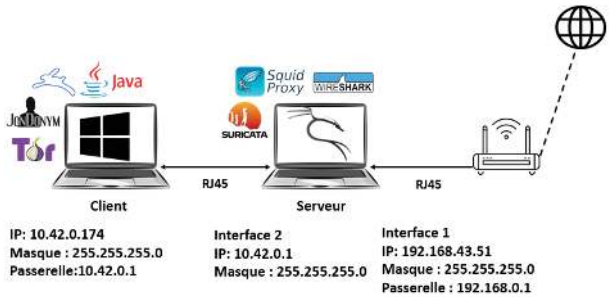


Figure 2.1 – Architecture client-serveur.

L'infrastructure se compose de deux machines principales connectées via un câble RJ45 :

2.3.1.1 Client

— Il s'agit d'un ordinateur Windows 10, configuré avec l'adresse IP « 10.42.0.174 » et utilisant « 10.42.0.1 » comme Passerelle.
— Sur cette machine, plusieurs logiciels de navigation anonyme sont installés, notamment Tor, JonDonym et Freenet, ce dernier fonctionnant sous Java.

2.3.1.2 Serveur

— La machine serveur fonctionne sous **Kali Linux**, elle possède deux interfaces réseaux : une connecté au client avec l'adresse IP « 10.42.0.1 » et l'autre connecté au modem avec l'adresse IP « 192.168.43.51 ».
— Le serveur agit comme un point de contrôle entre le client et Internet.
— Il joue le rôle de proxy grâce à l'outil **Squid**, utilisé pour filtrer les communications.
— Des outils de capture et de détection, comme **Wireshark** et **Suricata** sont également installés sur cette machine.

2.3.2 Serveur Proxy

2.3.2.1 Définition

Un serveur proxy est un système ou un routeur qui fonctionne comme un relais entre le client et le serveur. Il aide à empêcher un attaquant d'envahir un réseau privé et constitue l'un des outils utilisés pour construire un pare-feu. Le mot "proxy" signifie "agir au nom d'un autre", et un serveur proxy agit au nom de l'utilisateur. [26]

2.3.2.2 Fonctionnement

Un proxy fonctionne comme un intermédiaire entre un client et l'internet. Lorsqu'une requête d'accès à un site web est envoyée, celle-ci est d'abord dirigée vers le serveur proxy. Le serveur traite la demande, récupère le contenu souhaité et le renvoie à l'utilisateur. Ce processus masque effectivement l'adresse IP réelle de l'utilisateur.

En outre, le serveur proxy peut filtrer et bloquer le contenu indésirable, ainsi que mettre en cache des données pour accélérer les requêtes ultérieures. Les serveurs proxy sont donc précieux pour la sécurité et l'efficacité de l'utilisation de l'internet. [27]

2.3.3 Le Proxy Squid

2.3.3.1 Définition

Squid est un proxy de mise en cache pour le Web qui prend en charge le protocole de transfert hypertext (HTTP), le protocole de transfert hypertext sécurisé (HTTPS), le protocole de transfert de fichier (FTP), et bien plus encore. Il réduit la bande passante et améliore les temps de réponse en mettant en cache et en réutilisant les pages web fréquemment demandées. Squid dispose de contrôles d'accès étendus et constitue un excellent accélérateur de serveur. [28]

2.3.3.2 Configuration de Squid

Avant de présenter la configuration de notre serveur Squid, il est nécessaire d'introduire un concept fondamental : **les ACL (Access Control Lists)** .

Dans Squid, une ACL est un élément qui permet de spécifier certaines caractéristiques d'une requête, telles que l'adresse IP du client, le port de destination, le nom de domaine, l'URL et plus d'autres. Ces éléments servent à créer des règles de contrôle d'accès. Une fois définies, les ACL sont combinées dans des règles qui permettent d'autoriser ou de bloquer l'accès à des ressources selon des critères précis. Elle constituent ainsi la base du mécanisme de filtrage et de sécurisation du trafic dans Squid. [29]

Maintenant que nous avons défini les ACL et expliqué leur fonctionnement, nous allons présenter en détail la configuration que nous avons mise en place sur notre serveur Squid.

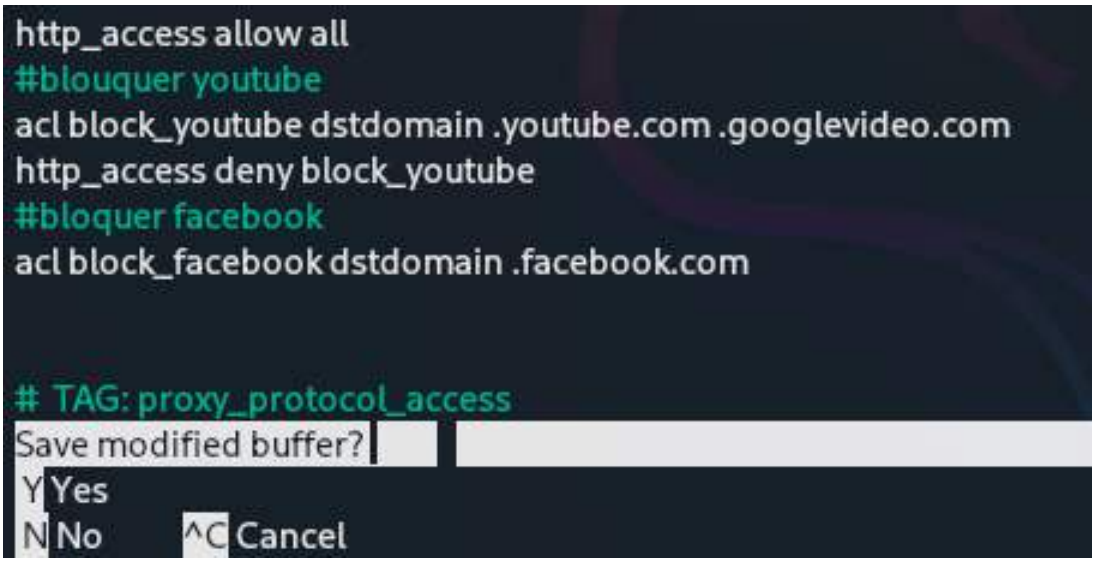


Figure 2.2 – Configuration du proxy Squid avec règles de filtrage.

Comme le montre la **figure 2.2**, nous avons tout d'abord autorisé globalement l'accès

à Internet pour tous les utilisateurs.

Par la suite, nous avons mis en place des restrictions visant à bloquer l'accès aux sites Facebook et YouTube, à l'aide des ACL.

2.3.4 Contournement des restrictions proxy par les réseaux anonymes

2.3.4.1 Accès via Tor Browser

À cette étape, nous avons configuré manuellement les paramètres de proxy sur le poste client afin que tout le trafic HTTP/HTTPS passe par le serveur Squid. La configuration est illustrée dans la **figure 2.3**.

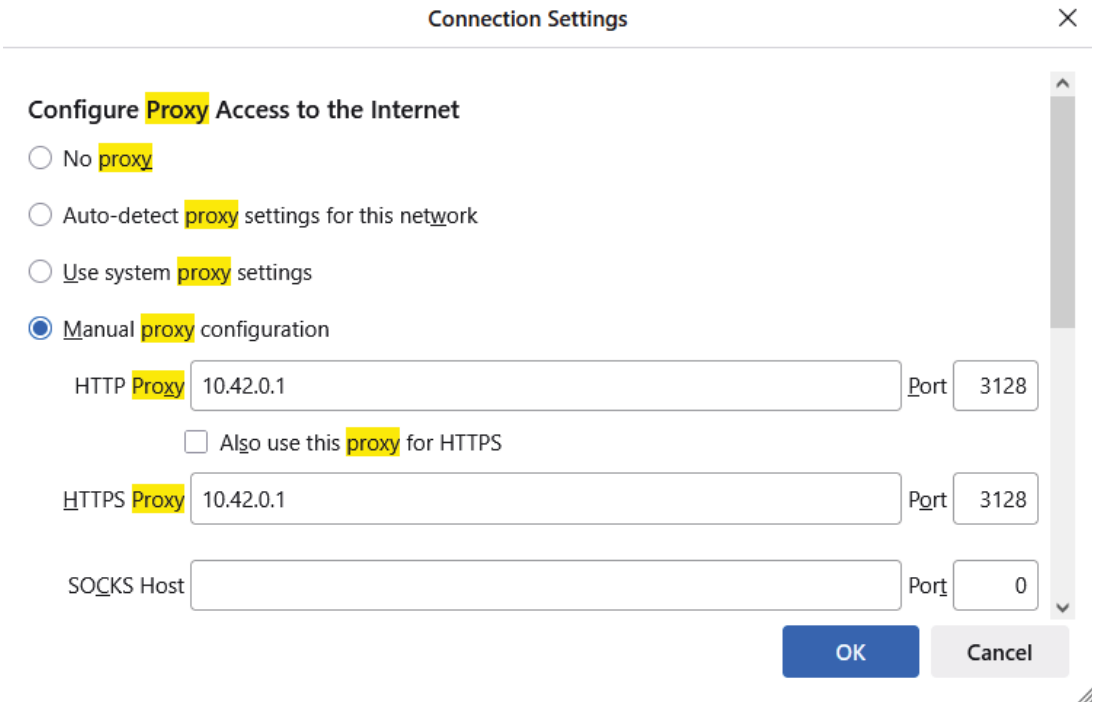


FIGURE 2.3 – Configuration du proxy Squid sur le poste client.

Construction du circuit Tor

Une fois Tor browser lancé, un circuit Tor est automatiquement construit. Il est composé de trois nœuds, comme précisé dans la partie théorique. Ce circuit est illustré dans la **figure 2.4**.

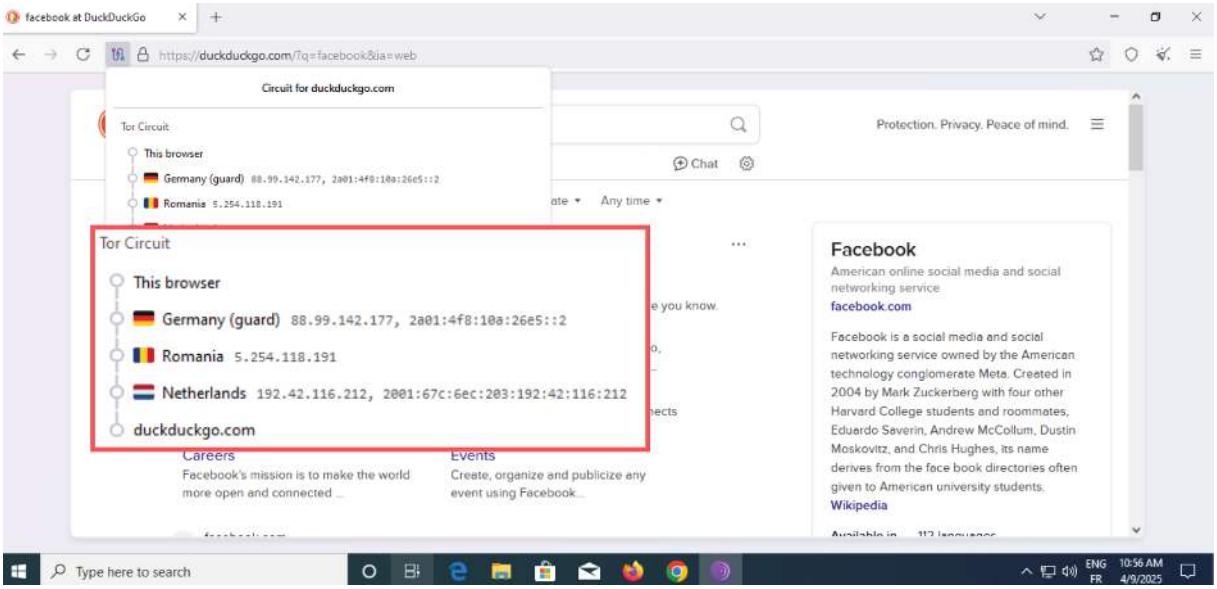


FIGURE 2.4 – Circuit Tor

Ensuite, nous avons tenté d'accéder à Facebook via des navigateurs différents. Comme prévu, l'accès via Firefox a été bloqué par Squid. En revanche, l'utilisation de Tor Browser a permis de contourner les restrictions et d'accéder au site bloqué.

Les **figures 2.5** et **2.6** illustrent cette expérience :

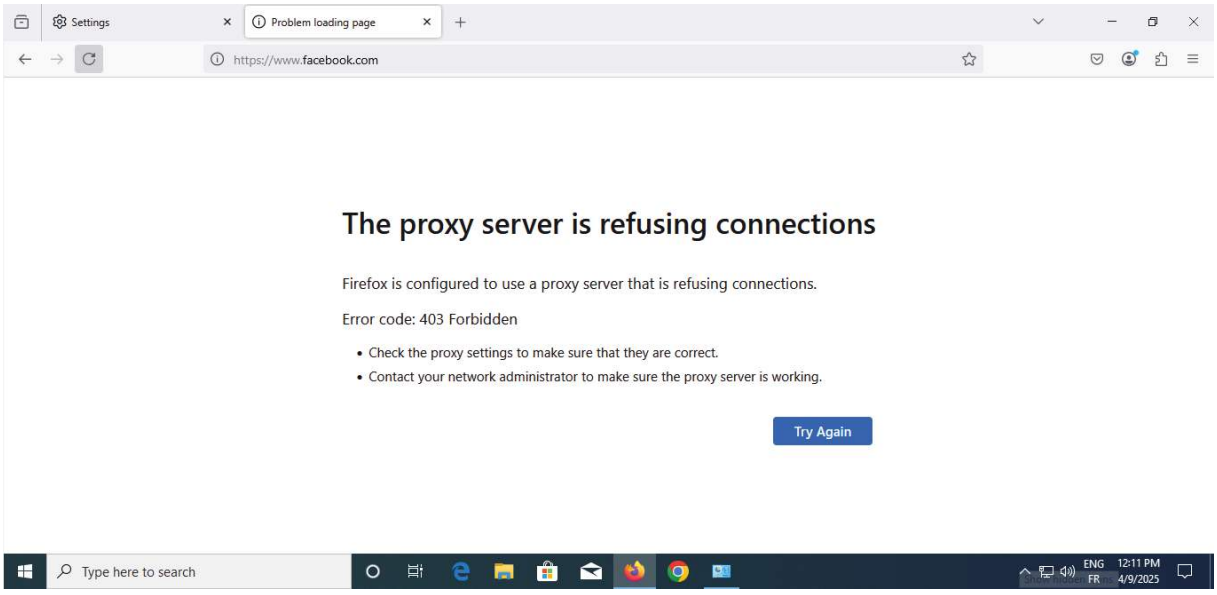


FIGURE 2.5 – L'accès à Facebook via Firefox.

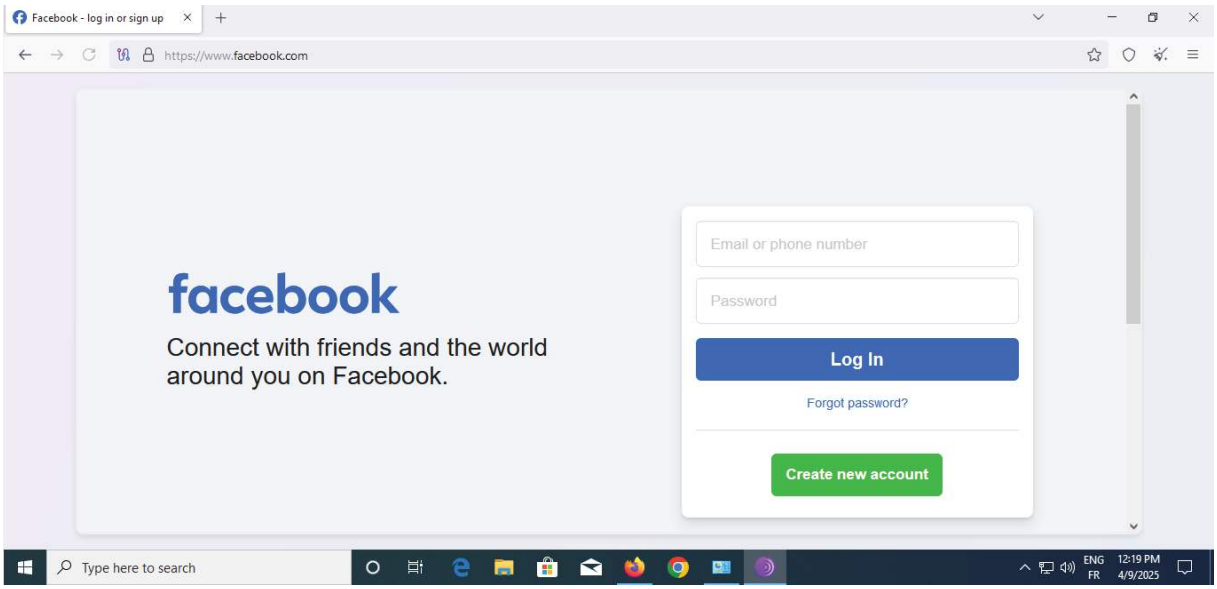


FIGURE 2.6 – L'accès à Facebook via Tor Browser.

Comme expliqué dans la **section 2.3.2.2**, un proxy comme Squid intercepte les requêtes HTTP/HTTPS et peut filtrer ou bloquer l'accès à certains sites comme dans notre cas Facebook en se basant sur le nom de domaines demandés. Dans le cas d'une tentative d'accès via un navigateur classique tel que Firefox, la requête transite directement par le proxy, qui peut alors analyser le domaine de destination Facebook.com et appliquer les règles de filtrage définies. Comme illustré dans la **figure 2.7.**

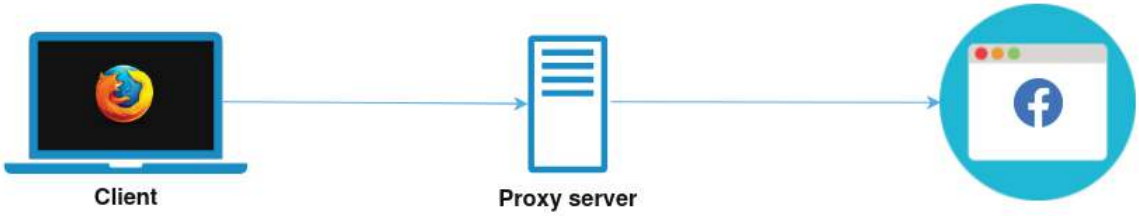


FIGURE 2.7 – Filtrage d'une requête Facebook via proxy .

En revanche Tor fonctionne différemment. Lorsqu'un utilisateur tente d'accéder à Facebook via Tor, la requête n'est pas directement transmise au proxy avec le nom de domaine en clair. Grâce au mécanisme de routage en oignon, Tor encapsule la requête en plusieurs couches de chiffrement, qui sont ensuite acheminées à travers une série de relais, comme illustré dans la **figure 2.8**.

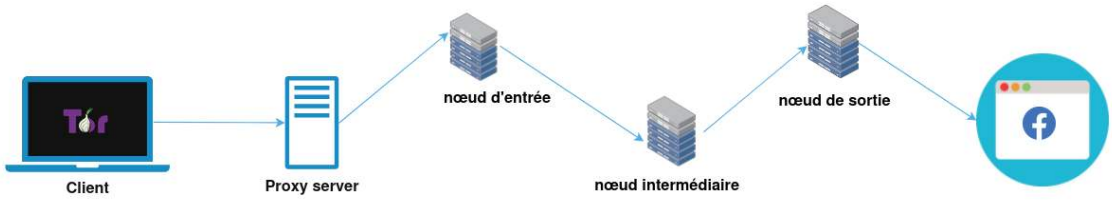


FIGURE 2.8 – Contournement du proxy par Tor via un circuit chiffré en oignon.

2.3.5 Accès via JonDoBrowser

De manière similaire, nous avons reproduit la même expérience en utilisant JonDoBrowser.

Lancement de JAP et formation d'une cascade

Avant d'ouvrir JonDoBrowser, il faut d'abord lancer le programme JAP (Java Anon Proxy), également appelé JonDo, car c'est lui qui établit une connexion à une cascade de relais. Une fois la cascade active, ce qui est signalé par l'indicateur vert dans l'interface, comme illustré dans la **figure 2.9**, il est alors possible d'ouvrir JonDoBrowser et de commencer la navigation anonyme.

FIGURE 2.9 – Interface de JAP montrant une cascade active.

Comme précédemment, l'accès à Facebook via Firefox a été bloqué. En revanche, JonDoBrowser a également réussi à contourner les restrictions et accéder au site bloqué. La **figure 2.10** montre bien le résultat.

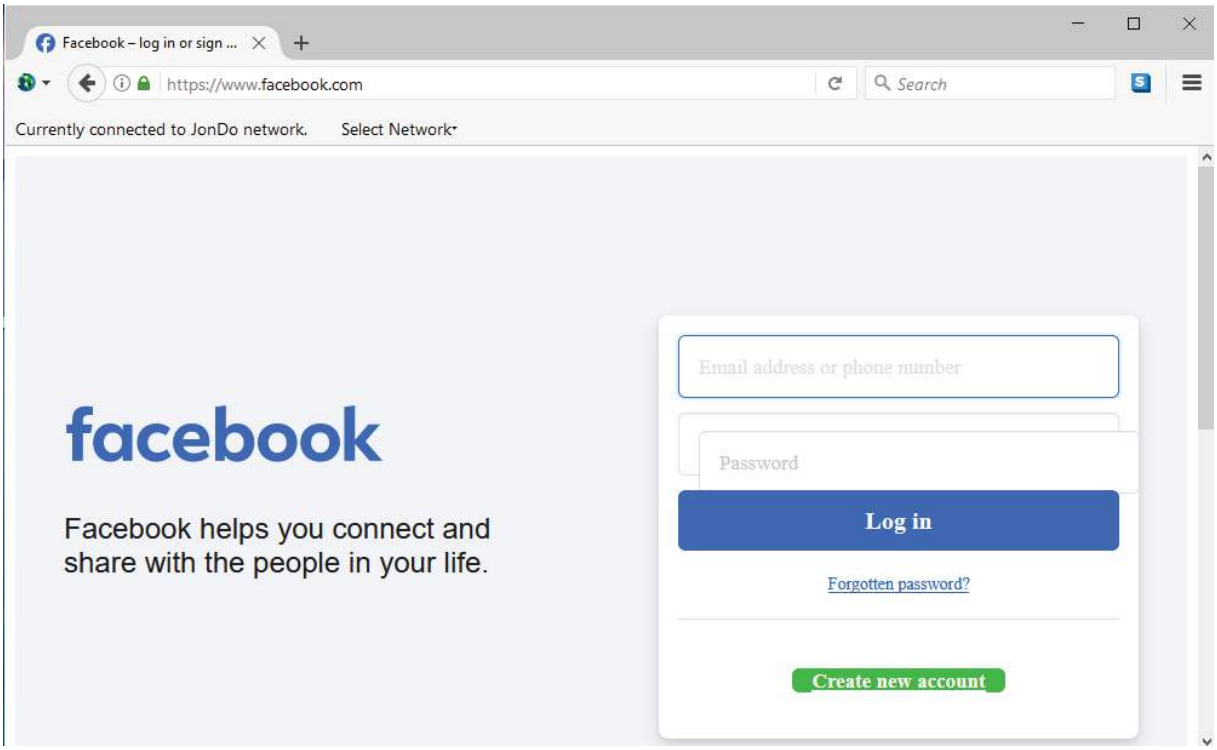


FIGURE 2.10 – L'accès à Facebook via JonDoBrowser.

Ce contournement s'explique par le mécanisme de fonctionnement de JonDonym. Comme expliqué précédemment, le trafic est d'abord pris en charge par le client JonDo (JAP), qui établit une connexion chiffrée vers la cascade de relais. Donc le proxy Squid ne voit pas la destination finale, mais uniquement le trafic à destination du premier relais de la cascade.

2.3.6 Accès via Freenet

À cette étape, nous avons tenté d'accéder à Facebook via le réseau Freenet. Cependant, cette accès est impossible, non pas parce que Freenet ne peut pas contourner les restrictions imposées par le proxy Squid, mais tout simplement parce qu'il ne permet pas d'accéder au Web classique.

En effet, Freenet ne génère pas de requêtes HTTP/HTTPS, il utilise son propre protocole, appelé FNP. Il permet uniquement la consultation de contenus héberges seulement au sein de son propre réseau, appelés Freesites.

Après avoir constaté l'impossibilité d'accéder au Web classique via Freenet, nous avons exploré son interface locale, qui donne accès aux contenus internes du réseau, comme illustré dans la **figure 2.11** ci-dessous.

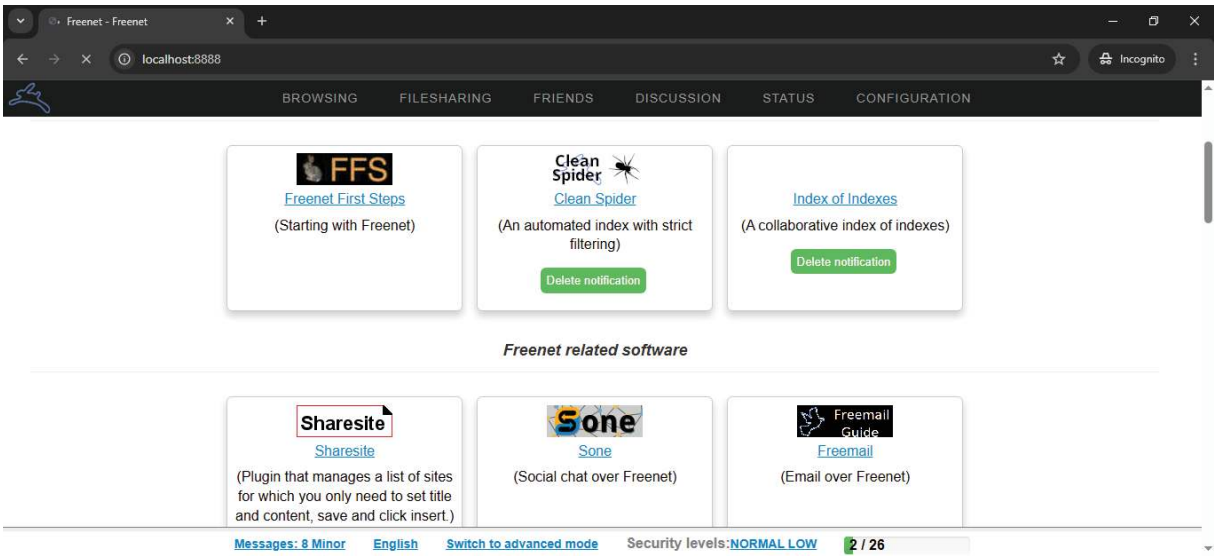


FIGURE 2.11 – Interface locale de Freenet.

Cette interface nous a permis de visualiser les différents services intégrés à Freenet, tels que :

— Freemail : un service de messagerie anonyme.
— Sharelists : un plugin pour créer et gérer des freesites.
— FFS (Freenet First Steps) : un guide de démarrage.

Comme le montre la **figure 2.12**, nous avons également remarqué un champ de saisie de clé Freenet, pour accéder a un freesite ou télécharger un fichier.

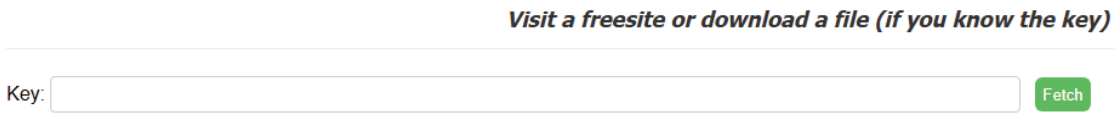


FIGURE 2.12 – Champ de saisie d'une clé Freenet.

Bien que Freenet ne permette pas de contourner les restrictions de proxy, il reste intéressant à analyser dans le cadre de notre étude.

2.3.7 Analyse et extraction du réseau Tor

Avant de procéder à l'analyse du trafic réseau, il est essentiel de définir ce que l'on entend par **signature réseau.** Une signature réseau est un ensemble unique de caractéristiques identifiables dans le trafic réseau, utilisé pour détecter des comportements spécifiques tels que les attaques ou l'utilisation de réseaux anonymes.

2.3.7.1 Wireshark

Wireshark est un analyseur de protocoles réseau, ou une application qui capture des paquets à partir d'une connexion réseau. Cet outil est considéré comme le renifleur (sniffer) de paquets le plus utilisé au monde en raison de sa facilité d'utilisation. Wireshark offre trois fonctionnalités principales : [30]

1. **Capture de paquets** : Wireshark écoute une connexion réseau en temps réel, en collectant d'une manière continue les flux de données, ce qui peut représenter des dizaines de milliers de paquets à la fois.
2. **Filtrage des données** : Wireshark est capable de trier et d'isoler les paquets à l'aide de filtres définis par l'utilisateur, afin de faciliter l'analyse.
3. **Visualisation** : l'interface de Wireshark permet d'examiner en profondeur le contenu d'un paquet réseau. Elle permet également de visualiser des conversations et des flux réseaux entiers.

La fenêtre principale de l'interface de Wireshark est structurée en trois sections illustrées dans la **figure 2.13** :

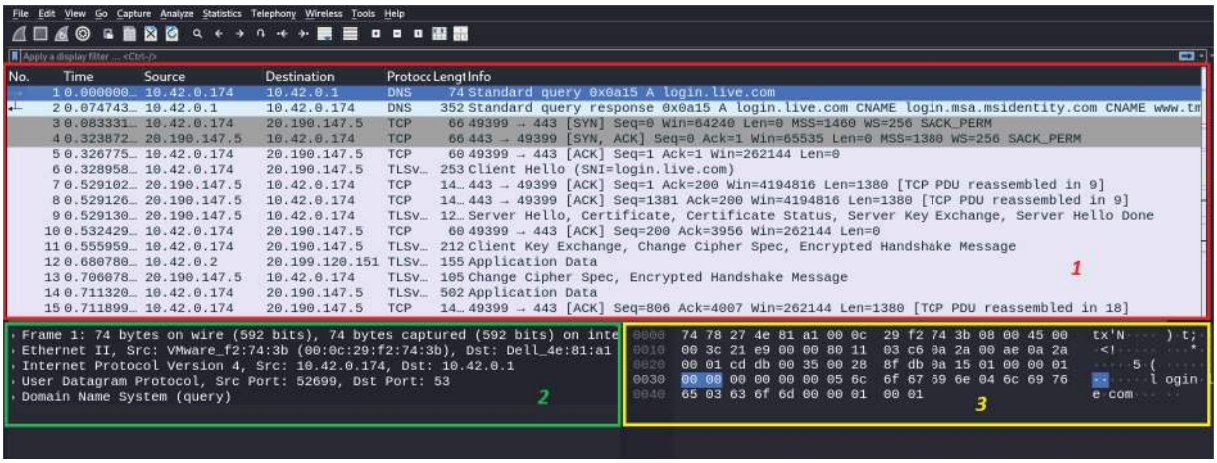


FIGURE 2.13 – Interface de Wireshark.

— **Section 1** : elle présente une liste des différents paquets qui circulent dans le réseau.
— **Section 2** : elle présente l'interprétation de Wireshark, divisée en plusieurs catégories. Une section affiche les informations générales sur la trame, telles que la taille, tandis que les autres sections représentent les différentes parties du modèle TCP/IP.
— **Section 3** : elle représente le contenu du paquet en hexadécimal et ASCII.

2.3.7.2 Méthodologie de l'analyse

Comme mentionné précédemment, le trafic Tor est entièrement chiffré à l'aide du protocole TLS. Ce chiffrement repose sur une connexion fiable, assurée par le protocole TCP.

Afin d'identifier le trafic Tor, il est essentiel de le comparer avec celui généré par Firefox, dans le but d'observer les différences entre les deux types de communication.

Nous allons utiliser Wireshark pour capturer et analyser deux processus fondamentaux dans le fonctionnement de Tor.

Le premier est le processus d'établissement d'une connexion TCP, appelé « TCP Three-Way-Handshake », qui correspond à une phase initiale d'échange entre le client et le nœud d'entrée Tor.

Le second processus concerne établissement d'une session sécurisé via le protocole TLS, également appelé «TLS Handshake», qui suit immédiatement la connexion TCP et permet le chiffrement du trafic.

2.3.7.3 Analyse du trafic réseau généré par Tor et Firefox

A. TCP Three-Way-Handshake

Le Handshake TCP en trois étapes est un processus fondamental qui permet d'établir une connexion fiable entre deux dispositifs sur un réseau TCP/IP. Il comprend trois étapes : [31]

1. **SYN (Synchronize)** : dans la première étape, le client souhaite établir une connexion avec un serveur. Il envoie donc un segment avec le drapeau SYN, qui informe le serveur que le client souhaite initier une communication et précise le numéro de séquence avec lequel il commence à envoyer les segments.
2. **SYN-ACK (Synchronize-Acknowledgement** : le serveur répond à la requête du client en envoyant un segment contenant les bits SYN et ACK activés.

3. **ACK (Acknowledgement)** : dans la dernière étape, le client confirme la réception de la réponse du serveur réel de données peut commencer.

Ce processus est illustré dans la **figure 2.14** ci-dessous.

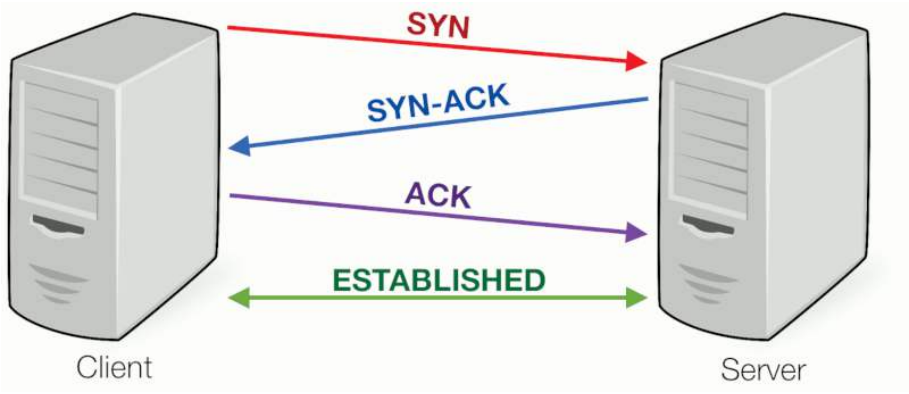


Figure 2.14 – TCP Three-Way-Handshake. [32]

Après avoir étudié le fonctionnement du TCP Three-Way-Handshake, nous allons analyser les trois segments échangés : SYN, SYN-ACK et ACK, pour les navigateurs Firefox et Tor Browser, afin d'extraire les informations suivantes :

— **Total Length :** c'est la longueur du datagramme, mesurée en octets, incluant l'en-tête Internet et les données.
— **Identification :** une valeur d'identification attribuée par l'émetteur pour faciliter le réassemblage des fragments d'un datagramme.
— **Time to Live (TTL) :** ce champ indique le temps maximal pendant lequel le datagramme est autorisé à rester dans le système Internet. Si ce champ contient la valeur zéro, alors le datagramme doit être détruit. [33]
— **Sequence number :** le numéro de séquence correspond au premier octet de données dan le segment.
— **Flags :** les flags également appelés **bits de contrôle**, servent à gérer l'établissement et le contrôle des connexions TCP.
— **Window :** ce champ correspond à la taille de fenêtre TCP, qui permet de contrôler le flux de données entre les deux hôtes.
— **Checksum :** le checksum est un champ utilisé pour assurer l'intégrité des données transmises dans un segment TCP.

— **Stream Index :** identifiant unique attribué à chaque flux TCP. [34]

— **RTT to ACK :** le temps de trajet aller-retour (RTT, pour Round-Trip Time), est défini comme une mesure, en millisecondes, du temps nécessaire pour qu'un paquet de données soit envoyé, **ajouté** au temps nécessaire pour recevoir l'accusé de réception de ce paquet. [35]

On va commencer l'analyse par le navigateur Firefox, pour faciliter la visualisation des paquets ciblés dans Wireshark, le filtre suivant est appliqué : ip.addr==157.240.243.35, cette adresse IP correspond à celle du site Facebook. La **figure 2.15** illustre bien le résultat de cette analyse.

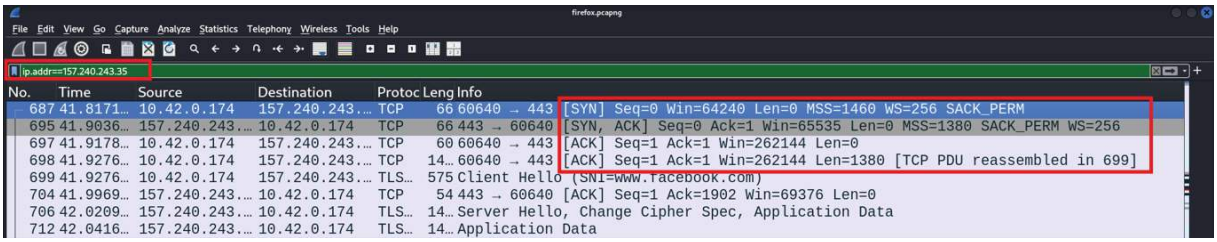


FIGURE 2.15 – Analyse de trafic Firefox via Wireshark.

Concernant Tor browser, nous allons d'abord vérifier que l'adresse 88.99.142.177, qui apparaît comme adresse du nœud d'entrée dans le circuit Tor illustré précédemment dans la **figure 2.4**, correspond bien à un nœud Tor. Pour cela, nous allons accéder à Tor Metrics, un site spécialisé dans l'affichage des informations des relais du réseau Tor.

FIGURE 2.16 – Nœud d'entrée Tor

La **figure 2.16** ci-dessus confirme que l'adresse IP 88.99.142.177, utilisée dans notre analyse correspond bien à un nœud d'entrée Tor actif.

L'analyse portera donc sur les paquets échangés entre le navigateur et le nœud d'entrée. Pour cela, nous utiliserons dans Wireshark le filtre suivant : ip.addr=88.99.142.177.

La **figure 2.17** illustre ces échanges.

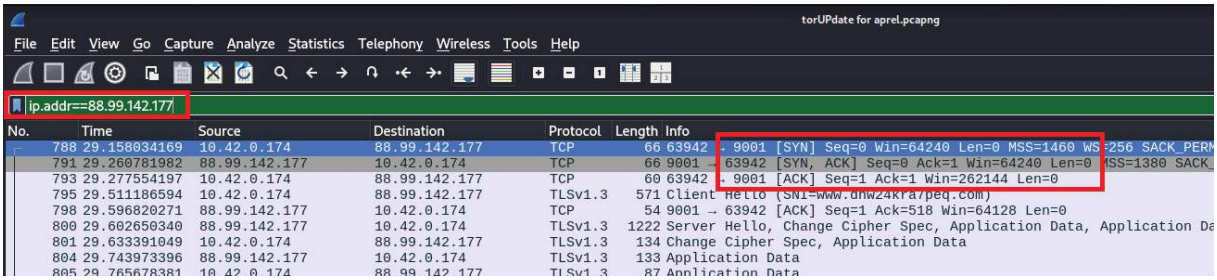


FIGURE 2.17 – Analyse de trafic Tor via Wireshark.

Les informations extraites lors de l'analyse des paquets SYN pour les deux navigateurs (Tor et Firefox) seront classées dans le **tableau 2.1**. La même méthodologie s'appliquera aux paquets SYN-ACK et ACK, comme illustré dans les **tableaux 2.2** et **2.3**.

1. **Paquet TCP (SYN) :**
   Le **tableau 2.1** présente les champs des paquets SYN envoyés respectivement par Tor Browser et Firefox, lors de l'initiation de la connexion TCP.

Champ	Tor Browser	Firefox
Port source	63942	60640
Port destination	9001	443
Sequence number	0	0
Flags	0x002 (SYN)	0x002 (SYN)
Window size	64240	64240
Checksum	0xde8c	0x6fe1
Total Length	52	52
Identification	0x593d(22845)	0xe645(58949)
TTL	128	128
Stream Index	12	13

TABLEAU 2.1 – Champs des paquets SYN du TCP Three-Way-Handshake.

2. **Paquet TCP (SYN-ACK) :**

Le **tableau 2.2** montre les réponses SYN-ACK reçues en retour.

Champ	Tor Browser	Firefox
Port source	9001	443
Port destination	63942	60640
Sequence number	0	0
Flags	0x012 (SYN-ACK)	0x012 (SYN-ACK)
Window size	64240	65535
Checksum	0x5cf9	0x5409
Total Length	52	52
Identification	0x0000(0)	0x0000(0)
TTL	49	51
Stream Index	12	13
RTT to ACK	0.10274747813	0.086528036

Tableau 2.2 – Champs des paquets SYN-ACK du TCP Three-Way-Handshake.

3. **Paquet TCP (ACK) :**

Le **tableau 2.3** présente la dernière étape du Three-Way Handshake, correspondant à l'envoi du paquet ACK par le client.

Champ	Tor Browser	Firefox
Port source	63942	60640
Port destination	9001	443
Sequence number	1	1
Flags	0x010 (ACK)	0x010
Window size	1024	1024
Checksum	0x946c	0x908c
Total Length	40	40
Identification	0x593e(22846)	0xe646(58950)
TTL	128	128
Stream Index	12	13
RTT to ACK	0.016772215	0.014179093

Tableau 2.3 – Champs des paquets ACK du TCP Three-Way-Handshake.

D'après les **tableaux 2.1**, **2.2** et **2.3** précédents, on constate les points suivants :

— Contrairement aux connexions HTTPS classiques qui utilisent le port de destination 443, comme le cas pour Firefox, Tor Browser utilise le port 9001, correspondant au nœud d'entrée du réseau Tor.
— Les ports sources sont dans les deux cas choisis aléatoirement.
— La valeur TTL observée dans les paquets SYN-ACK de Tor est plus faible (49), implique plus de sauts intermédiaires, ce qui est caractéristique de son routage en oignon.
— Le RTT observé lors du SYN-ACK est plus élevé pour Tor Browser (0.1027 s) que pour Firefox (0.0865 s), qui illustre bien l'impact de routage en oignon.

Cependant, ces éléments ne permettent pas de conclure de manière définitive qu'il s'agit d'une connexion Tor. Pour cela nous allons nous concentrer sur le processus TLS Handshake.

B. TLS Handshake

Le protocole TLS (Transport Layer Security) est conçu pour ajouter une couche de sécurité aux communications réseau. C'est lui qui fait la différence entre HTTP et HTTPS lors de la navigation sur Internet. [36]

Lors de l'établissement de la connexion, le client initie l'échange avec un message « ClientHello », contenant la version de TLS, un nombre aléatoire, une liste restreinte de suites de chiffrement sécurisées, ainsi que des paramètres pour l'échange de clés. Grâce à la simplification des options disponibles, le client peut anticiper la méthode de chiffrement que le serveur utilisera, ce qui évite des allers-retours supplémentaires.

Le serveur, à réception, génère la clé de session, puis répond avec un message « ServerHello » accompagné de son certificat, d'une signature et des paramètres choisis. Il termine en envoyant son message « Finished ».

Le client vérifie alors la validité du certificat, calcule à son tour la clé de session, et clôt l'échange en envoyant son propre message « Finished ». Une fois cette étape achevée, le chiffrement symétrique est activé pour sécuriser les communications, comme illustré dans la **figure 2.18** [37]

FIGURE 2.18 – TLS Handshake. [38]

Nous allons maintenant procéder à l'analyse des paquets TLS Handshake, en comparant le comportement du Tor Browser et de Firefox.

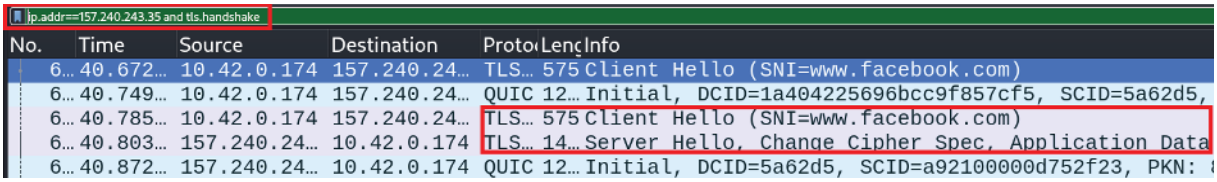


FIGURE 2.19 – Trrafic TLS Handshake de Firefox.

La **figure 2.19** illustre les échanges TLS capturés lors de connexions effectuées avec Firefox, tandis que la **figure 2.20** illustre ceux observés avec Tor Browser. Ces figures permettent d'observer les différences dans les messages ClientHello et ServerHello émis par les deux navigateurs.

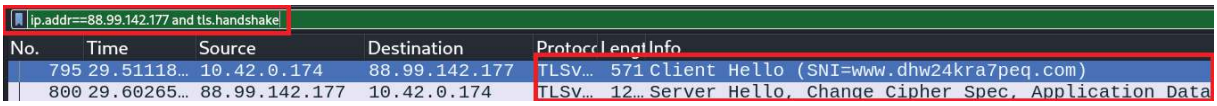


FIGURE 2.20 – Trafic TLS Handshake de Tor Browser.

Le processus débute par le message ClientHello, envoyé par le client. Le client fournit des informations incluant les éléments suivants : [39]

- Des données aléatoires générées par le client.
- Une liste de suites de chiffrement que le client prend en charge.
- Une liste de clés publiques que le serveur pourrait juger appropriées pour l'échange de clés.
- Les versions du protocole que le client peut supporter.

Suites de chiffrement

Nous commençons par comparer les suites de chiffrement proposées par les deux navigateurs, Firefox et Tor Browser.

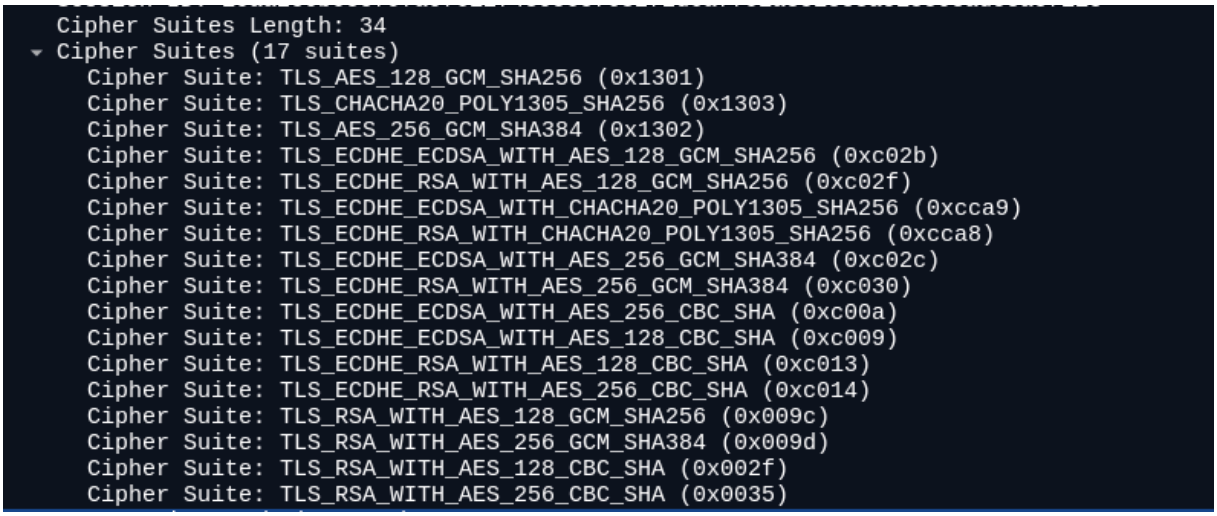


Figure 2.21 – Les suites de chiffrement proposées par Firefox.

La **figure 2.21** montre que Firefox propose 17 suites de chiffrement, tandis que Tor propose 18 suites de chiffrement présentées dans la **figure 2.22**.

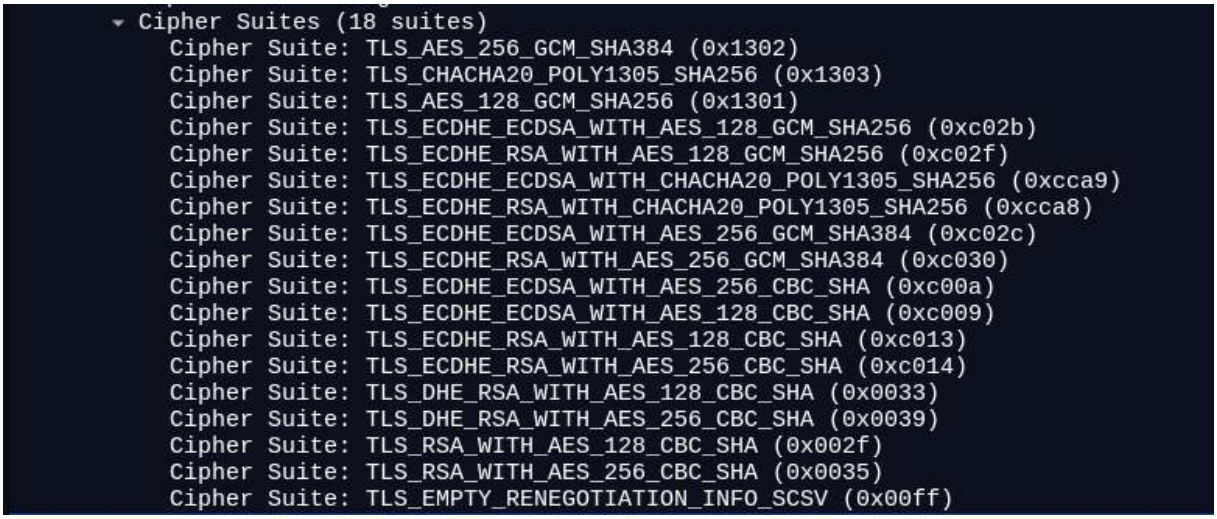


Figure 2.22 – Les suites de chiffrement proposées par Tor Browser.

Tor Browser contient trois suites de chiffrement différentes de celles proposées par le navigateur Firefox, qui seront illustrées dans le **tableau 2.4**.

Code	Suite de chiffrement
(0x0033)	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
(0x0039)	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
(0x00ff)	TLS_EMPTY_RENEGOTIATION_INFO_SCSV

Tableau 2.4 – Suites de chiffrement spécifiques proposées par le Tor Browser.

Extensions TLS Dans la suite, nous analyserons les extensions présentent dans les deux navigateurs.

Tor Browser supporte 11, tandis que Firefox supporte 17. L'étude portera sur celles illustrées dans les **figures 2.23** et **2.24.**.

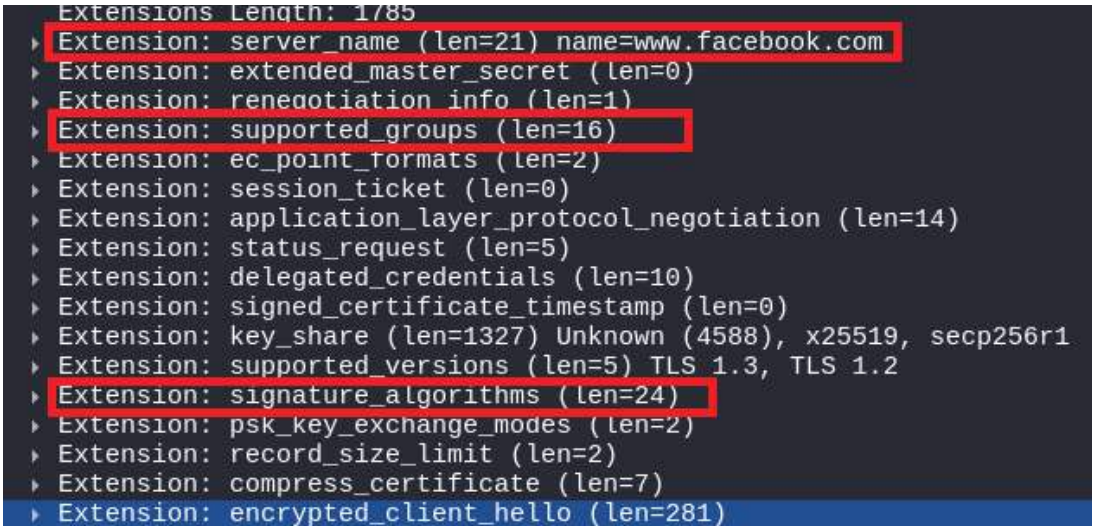


Figure 2.23 – Les extensions de Firefox.

Les extensions illustrées dans la **figure 2.23** sont présentes dans les deux navigateurs, Tor et Firefox, ce qui justifie notre intérêt pour leur analyse.

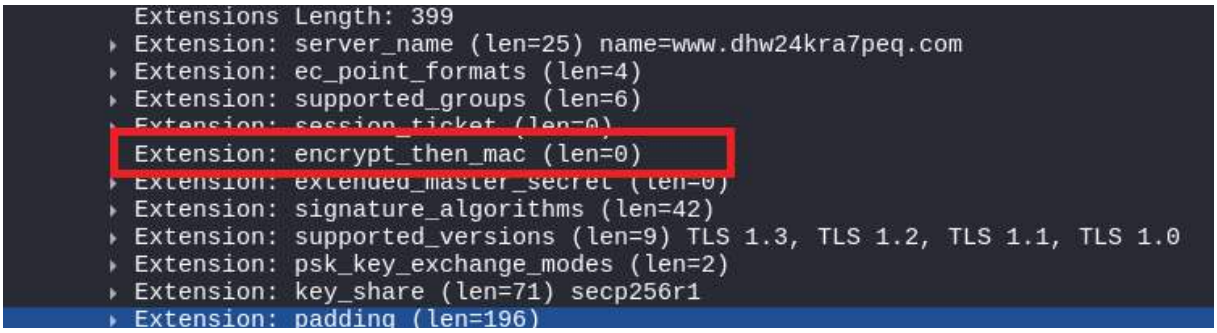


Figure 2.24 – Les extensions de Tor Browser.

L'extension illustrée dans la **figure 2.24** est propre à Tor et n'est pas présente dans les extensions supportées par Firefox.

1. **Extension Server_name :** l'extension server_name permet au client de spécifier le nom de domaine du serveur auquel il souhaite se connecter**.** [40]

FIGURE 2.25 – Extension server_name de Firefox.

La **figure 2.25** illustre l'extension **server_name** utilisée par le navigateur Firefox. On remarque que le nom de domaine est visible et correspond clairement au site visité.

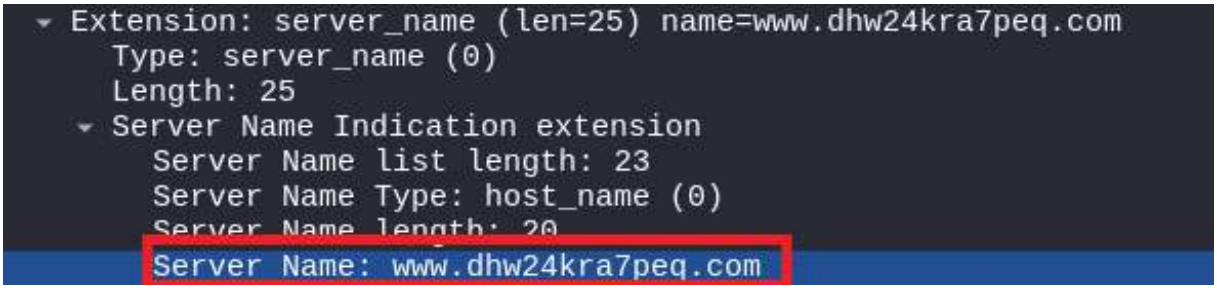


FIGURE 2.26 – Extensions Server_name de Tor.

En revanche, la **figure 2.26** illustre l'extension **server_name** du Tor Browser. Le nom de domaine utilisé prend la forme d'une chaîne de caractères aléatoires, qui ne correspond à aucun domaine existant.

2. **Extension Supported_groups :** cette extension signale les groupes supportés et négociés pour réaliser l'échange de clé lors du handshake. [40]

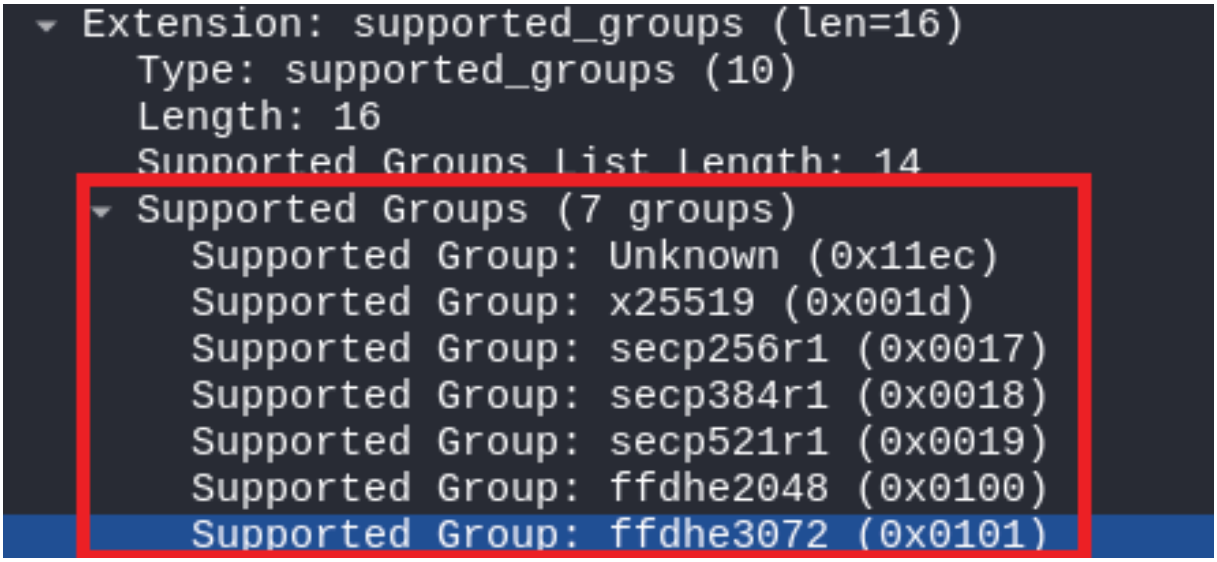


FIGURE 2.27 – Extension Supported_groups envoyées par Firefox.

Le navigateur Firefox supporte 7 groupes, comme illustré dans la **figure 2.27**.

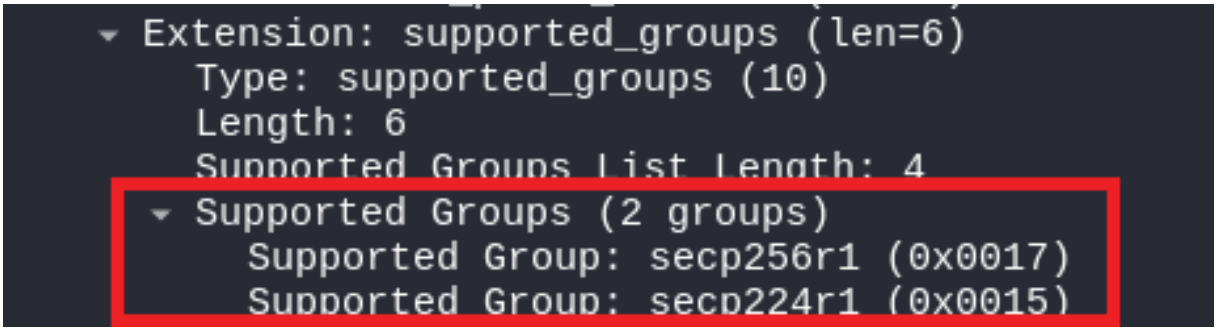


FIGURE 2.28 – Extension Supported_groups envoyées par Tor Browser.

En revanche, la **figure 2.28** illustre que Tor Browser ne supporte que 2 groupes.

L'extension **supported_group** inclut, dans Tor Browser, un groupe non présent chez Firefox, comme le montre le **tableau 2.5**. :

Code	Groupe
(0x0015)	secp224r1

TABLEAU 2.5 – Extension `supported_group` utilisée par Tor Browser.

3. **Extension Signature_algorithms :** cette extension signale les algorithmes

de hachage et de signature pris en charge pour vérifier l'authenticité des futurs messages du handshake. [40]

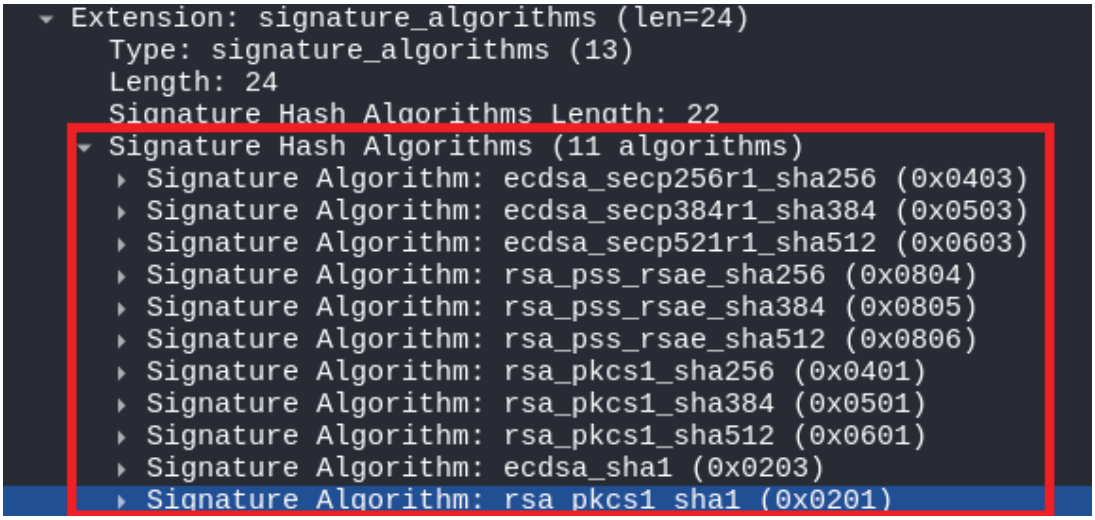


FIGURE 2.29 – Extension signature_algorithms envoyées par Firefox.

La **figure 2.29** illustre l'extension **signature_algorithms** envoyée par le navigateur Firefox. Ce dernier utilise 11 algorithmes de hachage. En comparaison, la **figure 2.30** illustre la même extension envoyée par Tor Browser, qui en utilise 23.

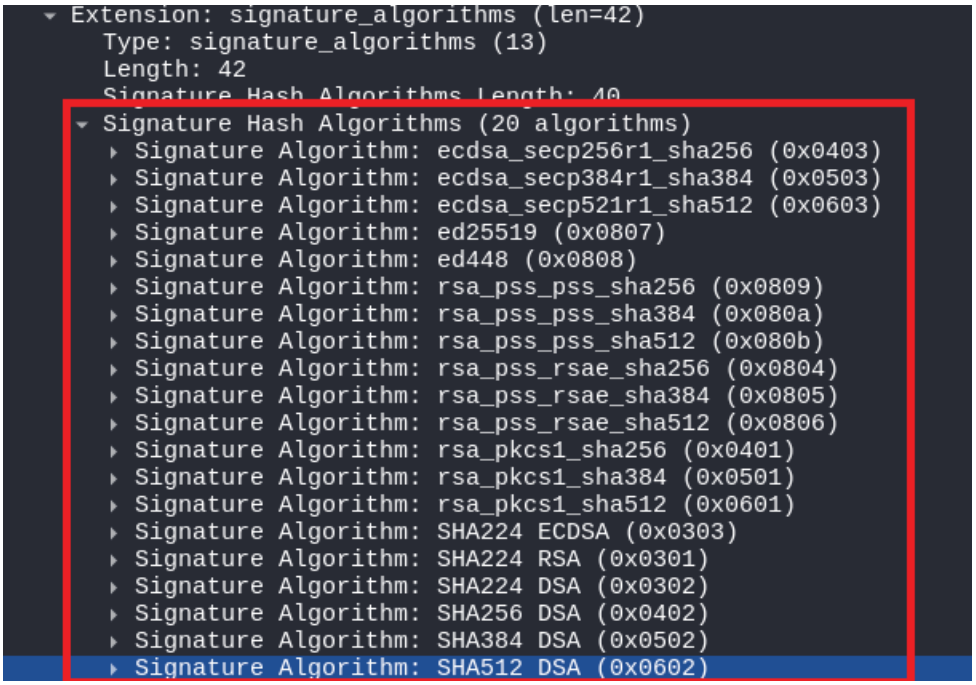


FIGURE 2.30 – Extension signatures_algorithms envoyées par Tor Browser.

Les algorithmes de hachage utilisés par Tor Browser seront présentés dans le **tableau 2.6.**

Code	Signature
(0x0807)	ed25519
(0x0808)	ed448
(0x0809)	rsa_pss_pss_sha256
(0x080a)	rsa_pss_pss_sha384
(0x080b)	rsa_pss_pss_sha512
(0x0303)	SHA224_ECDSA
(0x0301)	SHA224_RSA
(0x0302)	SHA224_DSA
(0x0402)	SHA256_DSA
(0x0502)	SHA384_DSA
(0x0602)	SHA512_DSA

TABLEAU 2.6 – Algorithmes de hachage envoyés par Tor Browser.

4. **Extension encrypt_then_mac :** Cette extension signale la prise en charge de la construction cryptographique encrypt-then-mac, en remplacement de la construction mac-then-encrypt historique. [40]

FIGURE 2.31 – L'empreinte numérique de l'extension encrypt_then_mac

La **figure 2.31** présente l'empreinte numérique de l'extension **encrypt_then_mac** observée dans le trafic TLS du Tor Browser. Cette extension n'est observée que dans le cas de Tor. Elle peut donc être considérée comme une signature caractéristique de ce réseau.

Après avoir reçu le message **ClientHello** du client, le serveur répond à son tour avec un message **ServerHello**. Celui-ci contient les éléments suivants : [39]

— Des données aléatoires générées par le serveur.
— Une suite de chiffrement sélectionnée parmi celles proposées par le client.
— Une clé publique pour l'échange de clés.
— La version du protocole TLS négociée pour la session.

La **figure 2.32** illustre le message **ServerHello** généré lors d'une connexion TLS à partir du navigateur Firefox. On observe que la version TLS négociée est **TLS 1.3**, comme indiqué dans l'extension **supported_versions**. La suite de chiffrement selectionée par le serveur est : **TLS_AES_128_GCM_SHA_256**

FIGURE 2.32 – Le message ServerHello de Firefox.

La **figure 2.33** illustre le message **ServerHello** généré lors d'une connexion TLS à partir du Tor Browser. Ici la version TLS négociée est **TLS 1.3**. En revanche la suite de chiffrement choisie par le navigateur Tor diffère de celle utilisée par Firefox**.** Il s'agit de : **TLS_AES_256_GCM_SHA_384**

FIGURE 2.33 – Le message ServerHello de Tor Browser.

Le **tableau 2.7** résume les suites de chiffrement négociées par les deux navigateurs.

Suite de chiffrement	Code	Navigateur
TLS_AES_256_GCM_SHA_384	(0x1302)	Tor
TLS_AES_128_GCM_SHA_256	(0x1301)	Firefox

TABLEAU 2.7 – Suites de chiffrement choisies par les deux serveurs.

D'après l'analyse des processus TCP Handshake et TLS Handshake, nous avons extrait des identifiants spécifiques à Tor. Ces identifiants seront utilisés par la suite comme signatures réseau pour détecter le trafic Tor à l'aide du système de détection d'intrusion Suricata.

2.3.8 Analyse et extraction du réseau JonDonym

2.3.8.1 Méthodologie de l'analyse

Pour analyser le trafic de JonDonym, nous allons suivre la même méthodologie que celle appliquée pour Tor. Nous étudierons ainsi le processus de négociation TCP Three-Way Handshake, ainsi que le TLS Handshake. Ces étapes seront comparées à celles observées lors d'une navigation via Firefox, afin d'identifier les signatures spécifiques associées à JonDonym.

2.3.8.2 Analyse du trafic réseau généré par JonDoBowser et Firefox

A. TCP Three-Way-Handshake

Étant donné que la capture du trafic généré par Firefox a déjà été réalisée lors de l'analyse comparative avec Tor, et que les conditions de test sont restées identiques, cette capture servira également de référence dans la présente analyse. Nous nous appuierons sur la capture Firefox déjà présentée en **section 2.3.7.3** (voir **figure 2.15**)
Lors de l'analyse du trafic généré par JonDonym, nous avons constaté l'apparition récurrente d'une même adresse IP : 141.76.42.133, comme illustré dans la **figure 2.34**.

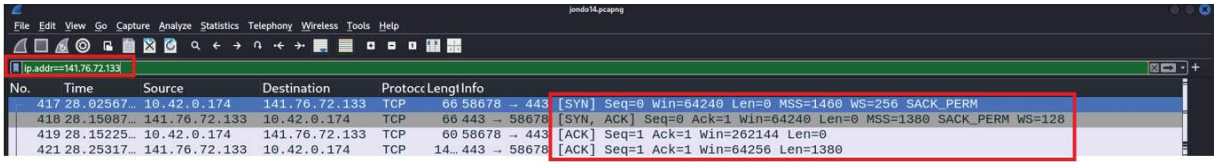


FIGURE 2.34 – Analyse de trafic JonDonym via Wireshark.

Nous avons alors soumis cette adresse au service en ligne ipinfo.io, ce qui nous a permis d'obtenir des informations sur son origine. Il s'agit d'un serveur mix faisant partie de la cascade de Dresden, comme illustré dans la **figure 2.35**.

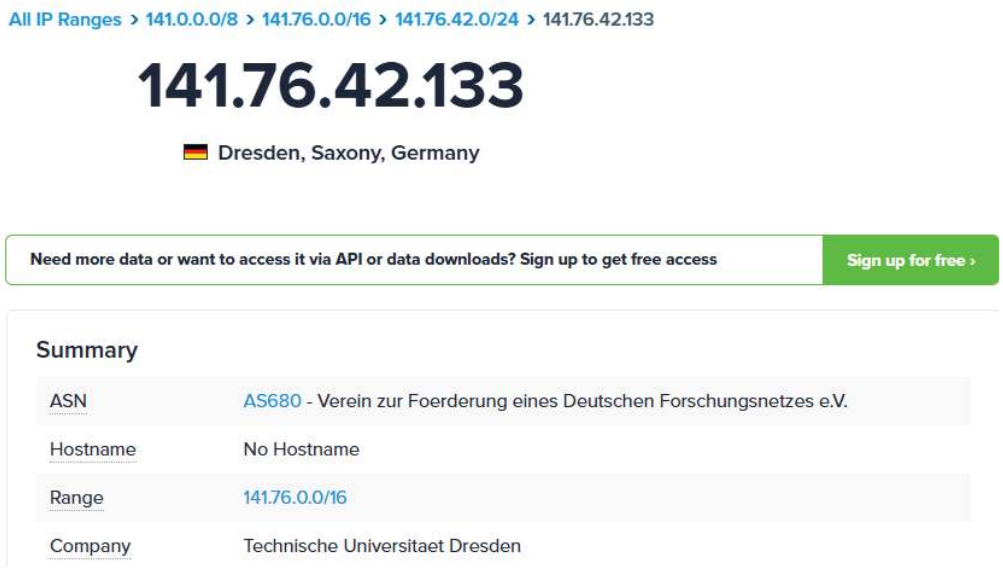


FIGURE 2.35 – Adresse du serveur mix de JonDonym.

À cette étape, nous allons comparer les éléments des paquets SYN, SYN-ACK et ACK observés lors du TCP Three-Way Handshake, cette fois-ci entre les deux navigateurs : Firefox et JonDoBrowser.

1. **Paquet TCP (SYN) :** Le **tableau 2.8** présente les champs des paquets SYN envoyés respectivement par JonDoBrowser et Firefox, lors de l'initiation de la connexion TCP.

Champ	JonDoBrowser	Firefox
Port source	58678	60640
Port destination	443	443
Sequence number	0	0
Flags	0x002 (SYN)	0x002 (SYN)
Window size	64240	64240
Checksum	0xb2e8	0x6fe1
Total Length	52	52
Identification	0x3d5d (15709)	0xe645 (58949)
TTL	128	128
Stream Index	14	13

TABLEAU 2.8 – Champs des paquets SYN du TCP Three-Way-Handshake.

2. **Paquet TCP (SYN-ACK) :**

Le **tableau 2.9** montre les réponses SYN-ACK reçues en retour.

Champ	JonDoBrowser	Firefox
Port source	443	443
Port destination	58679	60640
Sequence number	0	0
Flags	0x012 (SYN-ACK)	0x012 (SYN-ACK)
Window size	64240	65535
Checksum	0x7587	0x5409
Total Length	52	52
Identification	0x0000 (0)	0x0000 (0)
TTL	41	51
Stream Index	14	13
RTT to ACK	0.12520066	0.086528036

Tableau 2.9 – Champs des paquets SYN-ACK du TCP Three-Way-Handshake.

3. **Paquet TCP (ACK) :**

Le **tableau 2.10** présente la dernière étape du Three-Way Handshake, correspondant à l'envoi du paquet ACK par le client.

Champ	JonDoBrowser	Firefox
Port source	58678	60640
Port destination	443	443
Sequence number	1	1
Flags	0x010 (ACK)	0x010 (ACK)
Window size	1024	1024
Checksum	0xacfa	0x908c
Total Length	40	40
Identification	0x3d5e (15710)	0xe646 (58950)
TTL	128	128
Stream Index	14	13
RTT to ACK	0.001381789	0.014179093

Tableau 2.10 – Champs des paquets ACK du TCP Three-Way-Handshake.

À partir des tableaux précédents, plusieurs observations peuvent être faites :

— JonDoBrowser utilise le port 443 pour les connexions HTTPS, ce qui correspond au comportement standard d'un trafic web classique.

— La valeur TTL observée dans les paquets SYN-ACK de JonDoBrowser est plus faible (41), ce qui s'explique par l'architecture en cascade utilisée par le réseau JonDonym.

— Le RTT mesuré lors de l'envoi du paquet SYN-ACK est plus élevé pour JonDoBrowser (0,12520096 s) que pour Firefox (0,0865 s), traduisant une latence plus importante liée au fonctionnement du réseau anonyme.

Pour approfondir cette comparaison, nous allons maintenant procéder à l'analyse de TLS Handshake

B. TLS/SSL Handshake

Nous allons maintenant procéder à l'analyse des paquets TLS Handshake du réseau JonDonym, en comparant son comportement à celui de Firefox, comme cela a été fait pour le réseau Tor.

Pour cela, nous avons capturé le trafic réseau généré par le navigateur JonDoBrowser. Lors de l'analyse, nous avons constaté que le processus de Handshake SSL/TLS n'est pas visible dans le trafic observé. Contrairement à Firefox qui affiche clairement les étapes du TLS Handshake (comme illustré précédemment dans la **figure 2.19**), JonDoBrowser ne laisse apparaître que des paquets de type **Encrypted Data**, comme montré dans la **figure 2.36.**

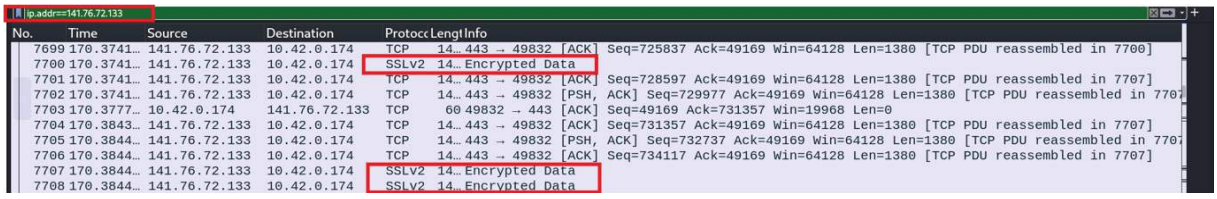


FIGURE 2.36 – Les paquets Encrypted Data de JonDoBrowser.

Cette absence rend l'analyse du processus de négociation TLS difficile dans le cas de JonDonym. Nous allons donc comparer les paquets **Encrypted Data** échangés par JonDoBrowser à ceux de Firefox, afin de mettre les différences entre les deux.

Lors de cette comparaison, une différence notable a été observée au niveau de la **taille totale des paquets (Total length)** et de la **valeur de la fenêtre TCP (Window Size)**.

Dans le cas de JonDoBrowser, la majorité des paquets présentent une taille constante, souvent égale à 1420 octets ou 1038 octets et une valeur de fenêtre TCP fixe, généralement

de 501 ou 1024, comme le montre la **figure 2.37**.

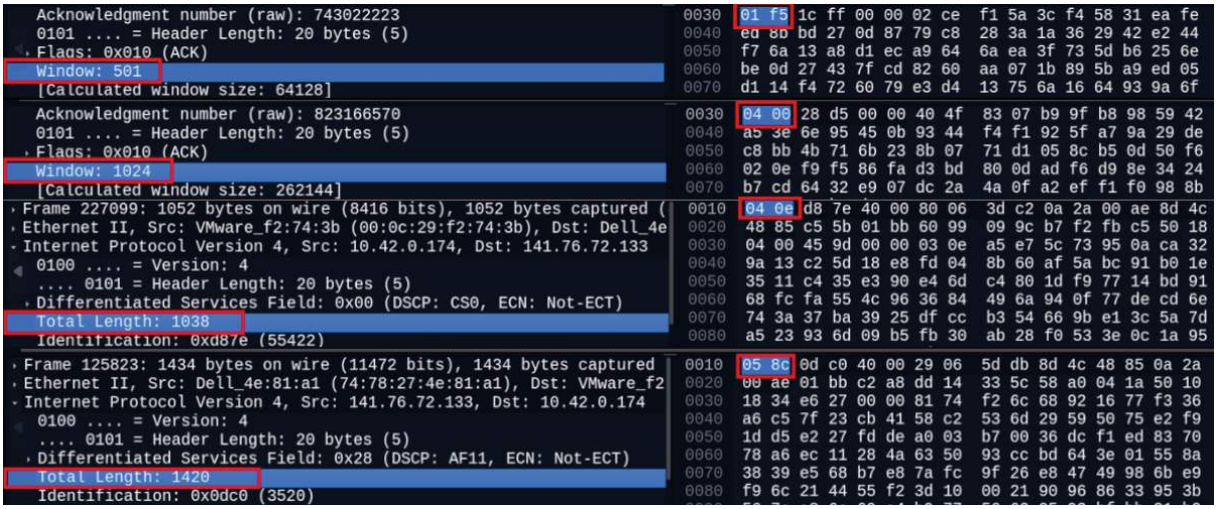


Figure 2.37 – Paquets Encrypted Data de JonDoBrowser présentant une taille constante et une fenêtre TCP fixe.

En revanche, dans le cas de Firefox, les paquets présentent au contraire des tailles variables, ainsi que des valeurs de fenêtre TCP non constantes, comme illustré dans la **figure 2.38**.

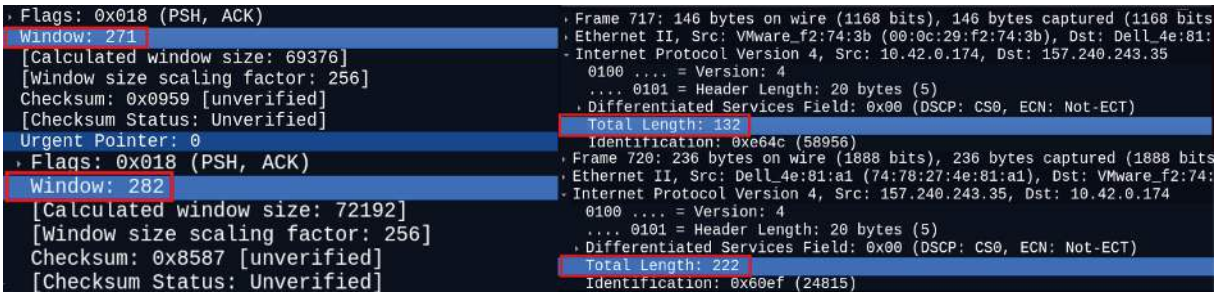


Figure 2.38 – Paquets Encrypted Data de Firefox présentant des tailles et des fenêtres TCP variables.

Les empreintes extraites lors de l'analyse du trafic généré par JonDoBrowser seront utilisées comme signatures, afin de détecter le trafic JonDonym à l'aide du système de détection d'intrusion Suricata.

2.3.9 Analyse et extraction du réseau Freenet

2.3.9.1 Méthodologie de l'analyse

Pour ce réseau anonyme, l'analyse se distinguera de celles des réseaux précédemment étudiés, car Freenet ne repose pas sur les protocoles HTTP ou HTTPS pour ses communications. L'analyse portera donc principalement sur le protocole UDP, comme mentionné précédemment.

Dans ce cadre, nous allons observer les ports utilisés en mode Opennet, accéder à la liste des nœuds connectés, et consulter le fichier de configuration afin d'extraire des paramètres réseau tels que la taille maximale des paquets.

2.3.9.2 Le protocole UDP (User Datagram Protocol)

Le User Datagram Protocol, est un protocole de communication utilisé sur Internet pour des transmissions nécessitant rapidité et faible latence, telles que la lecture de vidéos ou les recherches DNS.

Contrairement à TCP, qui établit une connexion fiable via le processus Three-Way-Handshake, UDP envoie des paquets directement à un ordinateur cible, sans établir de connexion préalable, sans indiquer l'ordre de ces paquets ni vérifier s'ils sont arrivés comme prévu. [41]

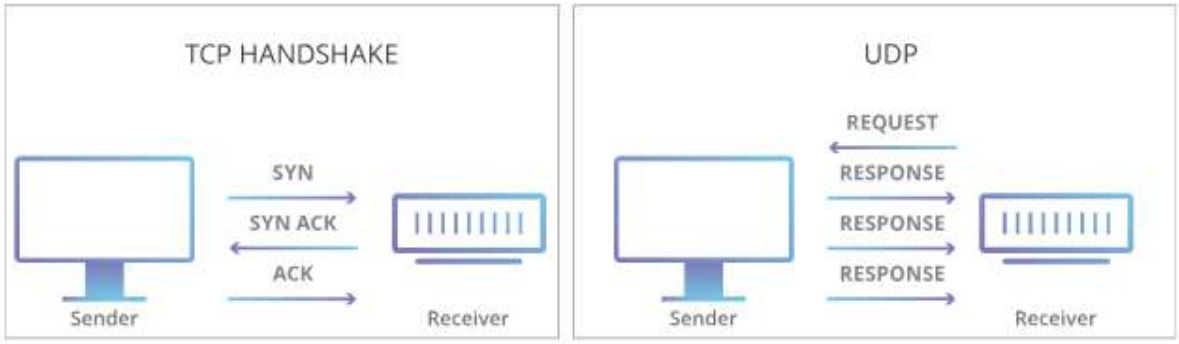


FIGURE 2.39 – Différence entre communication TCP et UDP. [41]

Ce fonctionnement d'UDP correspond bien aux objectifs du réseau Freenet. L'absence de mécanisme de connexion handshake, permet aux nœuds d'échanger des paquets plus rapidement. De plus, cette absence de handshake renforce la résistance à la censure, car elle rend le trafic plus difficile à identifier

2.3.9.3 Analyse du trafic réseau généré par Freenet

Nous allons analyser le trafic généré par Freenet en mode Opennet. L'objectif est d'identifier les adresses IP des nœuds connectés, les ports utilisés, ainsi que certains paramètres internes liés aux échanges UDP.

2.3.9.3.1 Accès à la liste des nœuds connectés

Une fois connecté en mode Opennet, Freenet maintient une liste active des pairs accessibles, que l'on peut consulter via l'adresse `http://localhost:8888/strangers/`. Cette liste contient toutes les adresses IP publiques et les ports d'écoute UDP des nœuds.

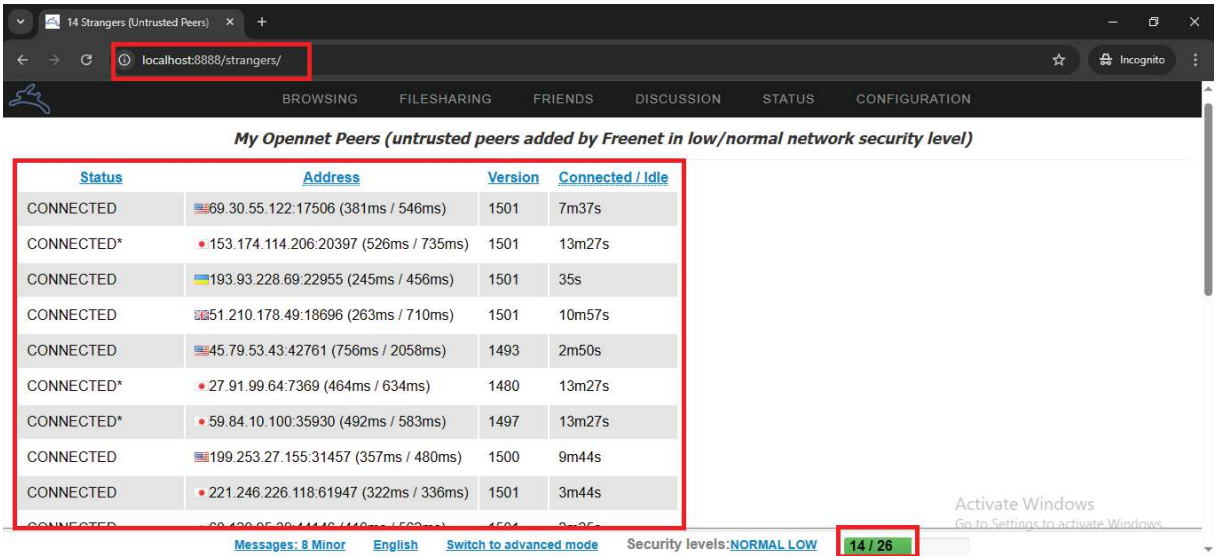


FIGURE 2.40 – Liste des pairs connectés en mode *Opennet* sur Freenet.

La **figure 2.40** illustre clairement les adresses IP des nœuds connectés, accompagnées de leurs ports dynamiques, ainsi que le statut de chaque nœud et la durée de la session en cours.

En bas à droite, on peut également observer le nombre total de connexions actives (**14 / 26**). Cela signifie que 14 pairs sont actuellement connectés parmi les 26 nœuds disponibles. Ce nombre peut évoluer en fonction de la disponibilité des nœuds dans le réseau Freenet.

2.3.9.3.2 Identification du port UDP utilisé

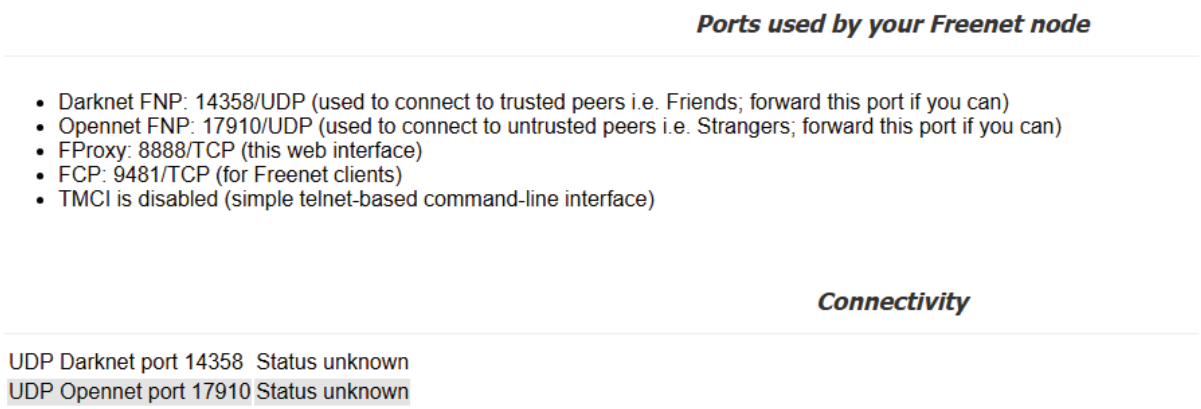


FIGURE 2.41 – Les ports utilisés par Freenet.

La **figure 2.41** présente les ports utilisés par un nœud Freenet. On remarque que le port UDP 17910 est réservé au mode Opennet, utilisé pour les communications entre les nœuds. Le port UDP 14358 est dédié au mode Darknet. D'autres ports sont également visibles, notamment pour l'interface web (8888/TCP) et pour les clients Freenet (9481/TCP).
À cette étape, nous avons procédé à une analyse du trafic via Wireshark. Lorsque l'analyse est lancée, elle intercepte immédiatement une série de communications UDP, utilisant notamment le port 17910 mentionné précédemment, comme illustré dans la **figure 2.42**.

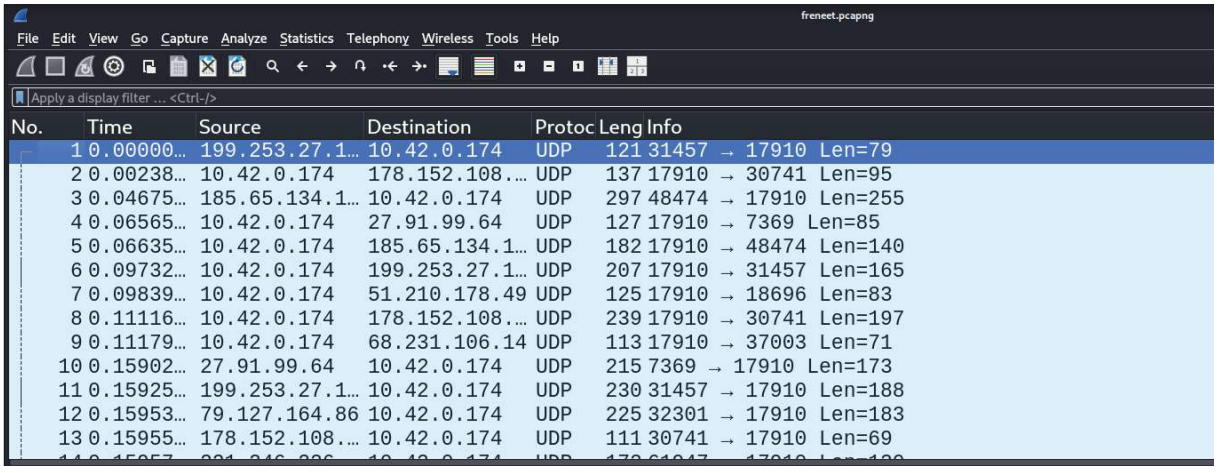


FIGURE 2.42 – Analyse du trafic Freenet via Wireshark.

Cela confirme le fonctionnement du protocole Freenet en mode Opennet, basé sur des échanges chiffrés en UDP entre les nœuds.

2.3.9.3.3 Consultation du fichier de configuration

Afin de mieux comprendre le comportement du trafic réseau généré par Freenet, nous avons consulté le fichier de configuration du nœud illustré dans la **figure 2.43**. Ce fichier contient plusieurs paramètres techniques, dont la taille maximale des paquets UDP utilisés pour les communications entre pairs.

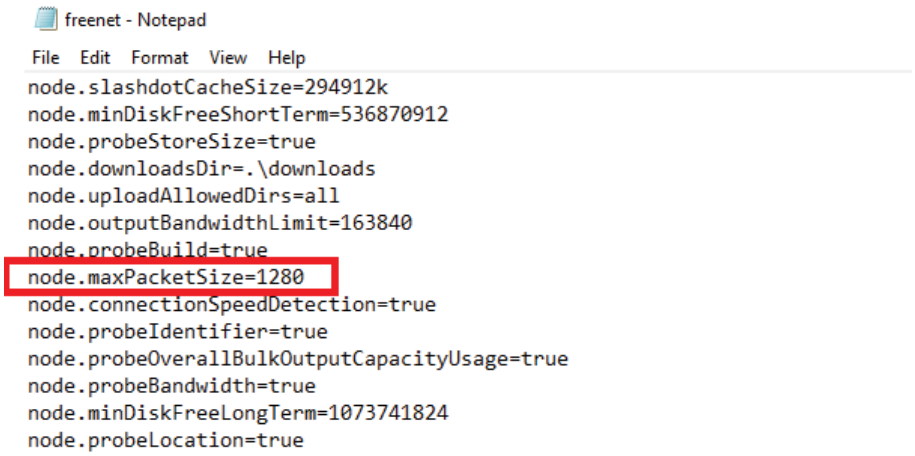


FIGURE 2.43 – Fichier de configuration de Freenet.

L'extraction de cette valeur nous a permis d'affiner notre analyse réseau. En effet, en la combinant avec le port UDP 17910, nous avons pu définir un filtre spécifique :

```
ip.len <= 1280 and udp.port == 17910
```

dans Wireshark, ciblant à la fois le port utilisé par Freenet en mode Opennet et la taille des paquets échangés, comme le montre la **figure 2.44**.

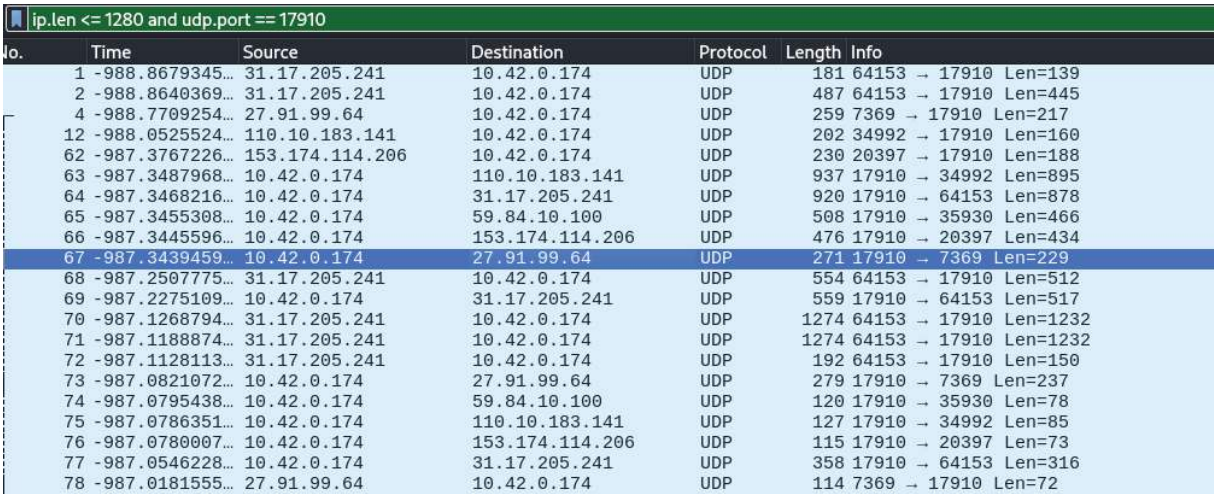


FIGURE 2.44 – Application de filtre pour cibler les connexions Freenet

Grâce à ce filtre, nous avons pu identifier plusieurs paquets échangés entre notre nœud

et d'autres pairs du réseau. Les adresses IP observées dans la capture réseau correspondent bien à celles listées précédemment, ce qui confirme que les échanges Freenet transitent effectivement par le port UDP 17910 avec des paquets, inférieur à 1280 octets.

Pour rendre la lecture plus facile et comparer rapidement les différentes signatures, On a résumé les signatures caractéristiques de chaque réseau anonyme dans le **tableau 2.11**.

Champ	Réseau	Signatures
1	Tor	Port TCP utilisé (habituellement 9001).
2	Tor	Suites de chiffrement spécifiques dans le message `ClientHello`.
3	Tor	Algorithmes de signature utilisés dans `ClientHello`.
4	Tor	L'extension Encypted then mac
5	Tor	Les suites de chiffrement dans le message ServerHello.
8	JonDonym	Adresse IP du relais dans la cascade de mix.
9	JonDonym	Longueur totale du paquet Encrypted Data échangé.
10	JonDonym	Taille de la fenêtre TCP (Window Size).
11	Freenet	Numéro de port UDP utilisé.
12	Freenet	Taille maximale des paquets UDP échangés dans le réseau.

Tableau 2.11 – Signatures spécifiques aux réseaux anonymes Tor, JonDonym et Freenet.

2.4 Conclusion

Dans ce chapitre, nous avons simulé un environnement réseau représentatif d'une entreprise afin d'observer concrètement le comportement des réseaux anonymes Tor, JonDonym et Freenet lorsqu'ils sont utilisés pour contourner des restrictions d'accès.

Ce chapitre nous a permis de capturer et analyser le trafic réseau généré par ces outils à l'aide de Wireshark. L'analyse du trafic réseau capturé a permis d'identifier des signatures caractéristiques propres à chaque réseau anonyme. Ces signatures représentent : les ports utilisés, les types de paquets échangés, les extensions TLS spécifiques, les longueurs de paquets, ainsi que les suites de chiffrement employées. Elles permettent de différencier clairement le trafic anonyme du trafic classique pour une détection efficace.

Le prochain chapitre sera consacré à la détection de ces réseaux à partir des signatures réseau extraites, en les implémentant sous forme de règles dans le système de détection d'intrusion Suricata.

Chapitre 3

Implémentation des signatures et détection

3.1 Introduction

Aucun système informatique n'est entièrement protégé contre les attaques. Pour une entreprise connectée à internet, la question n'est plus de savoir si elle sera attaquée mais plutôt quand cela se produira. Face à cette réalité l'une des réponses possibles consiste à mettre en place divers moyens permettent de renforcer la sécurité. Parmi ces moyens, les systèmes de détection et d'intrusion (IDS-Intrusion Detection Systems) sont de plus en plus adoptés par les entreprises pour faire face aux menaces. Un IDS agit en surveillant discrètement le trafic réseau afin de repérer toute activité suspecte et de générer des alertes en cas de détection d'intrusion. Dans ce chapitre, nous commencerons par élaborer des règles à partir des signatures recueillies précédemment , que nous intégrerons dans le système de détection d'intrusion « Suricata ». Nous procéderons ensuite à l'évaluation de la fiabilité de ces règles pour identifier l'utilisation de réseaux anonymes tels que Tor, JonDonym et Freenet.

3.2 Système de détection et d'intrusion

3.2.1 définition

IDS est un système ou un logiciel conçu pour surveiller l'activité d'un réseau ou un système informatique afin d'y détecter des comportements suspects ou malveillants pouvant indiquer une tentative d'intrusion, une attaque ou une utilisation non autorisée des ressources. Certains termes techniques sont fréquemment utilisés dans le contexte des

systèmes de détection d'intrusion (IDS), notamment :
- **Faux positif :** Correspond à une alerte générée par l'IDS alors qu'aucune attaque réelle n'a eu lieu.
- **Faux négatif :** Désigne une attaque véritable qui n'a pas été détectée par le système IDS. [42]

3.2.2 Fonctions

Un IDS remplit généralement quatre fonctions principales : **journalisation**, **l'analyse**, **l'action** et **la gestion**.
a. Journalisation : Consiste à consigner les événements détectés dans des fichiers de log afin d'assurer la traçabilité et l'analyse ultérieure.
b. Analyse : Permet d'examiner les journaux système afin d'identifier des intentions malveillants à partir de l'ensemble des données collectées par l'IDS. Généralement, deux approches sont utilisées : l'analyse par signature, reposant sur la détection de motifs connus d'attaque, et l'analyse par détection d'anomalies, qui identifie les écarts par rapport à un comportement normal.
c. Action : Lorsque le système identifie une activité potentiellement dangereuse, il déclenche une alerte destinée à prévenir l'administrateur de sécurité afin qu'une réponse appropriée puisse être apportée.
d. Gestion : Un IDS doit être surveillé et géré en continu pour bien fonctionner, on peut le comparer à une caméra de surveillance qui observe sans arrêt le réseau pour repérer des activités suspectes. [43]
Il existe trois types d'IDS :
Les NIDS : qui surveille le trafic réseau.
Les HIDS : qui analysent l'activité des hôtes.
Les IDS hybridques : qui combinent les deux pour une détection plus complète.
les HIDS sont efficaces pour identifier des compromissions locales. Tandis les NIDS offrent une vue global du réseau, La **figure 3.1** vient illustrer les trois étapes fondamentales du fonctionnement des NIDS : **capture**, **signature**, **alertes**. [44]

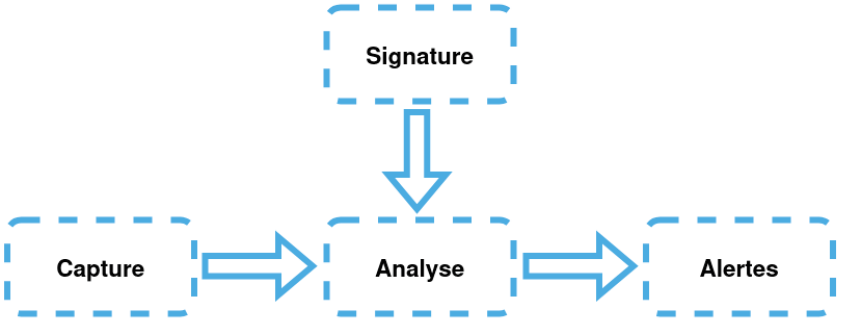


FIGURE 3.1 – Fonctionnement du NIDS.

Capture : Le NIDS surveille le réseau en temps réel et capture les paquets à l'aide d'outils comme PCap"Packet capture".
Analyse : Il compare le trafic capturé à des signatures d'attaques connues. cela permet de détecter rapidement les menaces identifiées, mais nécessite des mises à jour régulières.
Alertes : Lorsqu'une anomalie est détectée, le système envoie une alerte à l'administrateur et enregistrer l'événement pour une analyse ultérieure.
Signatures : Permet de reconnaître des modèles de trafic malveillant connus en comparant les paquets réseau à une base de règles prédéfinies.

3.2.3 Modes de détection

Les IDS utilisent plusieurs méthodes pour identifier les comportements suspects ou malveillants dans un système ou un réseau. Les principaux modes de détection sont les suivants :
a. Détection par signature : Cette méthode repose sur la comparaison du trafic réseau ou des événements système avec une base de données de signatures d'attaques connues. Elle ne permet de détecter que les attaques pour lesquelles une signature a été préalablement enregistrée.
b. Détection par anomalie : Elle consiste à établir un profil de comportement normal (trafic habituel), puis à signaler toute activité s'en écartant de manière significative. Cette méthode est particulièrement efficace pour détecter des attaques inconnues ou des variantes nouvelles d'attaques existantes.
c. Détection hybride : Elle combine les approches par signature et par anomalie, permettant d'obtenir un meilleur équilibre entre détection d'attaques connues et identification de comportements nouveaux ou suspects. [45]

3.3 Outils de détection utilisés

3.3.1 Présentation générale de Suricata

Suricata est un moteur open source de détection (IDS) et de prévention (IPS) d'intrusions réseau, développé par l'Open Information Security Foundation (OISF). Il permet d'analyser en temps réel le trafic réseau afin de détecter des menaces et, dans certains cas, de prévenir les attaques.
En mode IDS, Suricata surveille passivement le trafic à la recherche d'activités malveillantes ou de violations des politiques de sécurité.
En mode IPS, il agit de manière proactive en bloquant le trafic malveillant pour protéger les ressources du réseau contre toute tentative de compromission. [44]

3.3.2 L'architecture de Suricata

L'architecture de Suricata est conçue de manière modulaire afin d'assurer un traitement complet du trafic réseau, depuis la capture des paquets jusqu'à la génération d'alertes, comme illustré dans la **figure 3.2**.

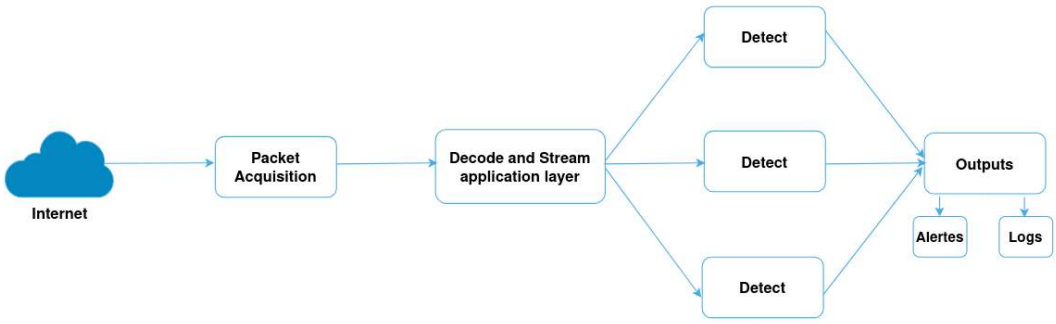


Figure 3.2 – Suricata multi-threaded architecture. [46]

a. Packet Acquisition : L'IDS fonctionne en mode active, un ou plusieurs interfaces réseaux en mode promiscuous, ce qui offre la capacité d'intercepter toute les paquets circulant sur le réseau.
b. Decode and Stream application layer : Le module "décodage et reconstruction de flux" joue un rôle crucial en transformant les paquets bruts en flux complet et en analysant les protocoles applicatifs.
c. Detect : Le module de détection consiste à utilise plusieurs threads de détection parallèles pour analyser le trafic réseau en le comparant à un ensemble des régles

prédéfinie.
d. Outputs : Une fois qu'une règle est déclenchée, Suricata exécute l'action définie (génération des alertes, l'enregistrement des logs). Ensuite les événements détectées sont journalisées. [47]

3.3.3 Format des règles de détection

Les signatures sont essentielles dans le fonctionnement de Suricata. En général on utilise des ensembles de règles déjà disponible. Chaque règle ou signature est composée de trois éléments principaux :
L'action : Indique ce que Suricata doit faire si la signature correspond.
L'en-tête : Définit le protocole, les adresses IP, les ports et le sens de trafic.
Les options : Précisent les conditions supplémentaires à respecter pour déclencher la règle.
On prend un exemple de règle :

```
alert tcp any any -> any 9001 (msg:"[GPL ATTACK RESPONSE id check
returned root"; content:"uid=|28|root|29|";classtype:bad-unknown;
sid:2100498; rev:1;)
```

-L'action c'est **alert** , **ip any any ->** **any any** c'est l'en-tête, et le reste de la signature qui commençant par **(msg :GPL ATTACK RESPONSE...** contient les options de la règle.

3.3.3.1 Action :

Chaque signature possède des propriétés spécifiques, dont l'une des plus importantes est l'action, qui définit le comportement du système lorsqu'une signature est détectée.
Quatre types d'actions sont possibles :
a. Drop : Réservée une mode IPS, cette action bloque immédiatement le paquet détecté comme malveillant.
b. Reject : Elle rejette activement lr paquet en informant à la fios l'expéditeur et le destinataire via un message de refus.
c. Alert : Le paquet est autorisé à circuler, mais une alerte est générée et enregistrée, visible uniquement par l'administrateur du système.
d. Pass : Lorsque cette action est utilisée, Suricata ignore le paquet concerné et arrête son analyse pour ce paquet uniquement.

3.3.3.2 L'en-tête :

Une signature Suricata contient plusieurs éléments essentiels :
a. Protocole : Ce champ indique à Suricata le protocole réseau concerné (exemple : TCP, UDP, ICMP...).
b. Adresse IP source et destination : Elles précisent les hôtes impliquées dans le trafic analysé.
c. Ports source et destination : Utilisées pour identifier les services réseau ciblés.
d. Direction : Elle définit le sens du flux réseau auquel la signature doit s'appliquer.
-la flèche -> signifie que la règle s'applique uniquement dans le sens source vers destination.
-la notation <> permet une correspondance dans deux directions.
La **figure 3.3** illustre ces éléments de l'en-tête avec un exemple coloré de signature.

FIGURE 3.3 – Composante de l'en-tête.

3.3.3.3 Option de règle :

Le reste de la règle Suricata est constitué d'options, qui sont placées entre parenthèses et séparées par des points-virgules. certaines d'entre elle utilisent des paramètres définis après un mot-clé, suivi de deux-points, Voici quelque exemple courants :
msg : Fournit un message descriptif associé à l'alerte générée.
sid : Identifiant unique attribué à chaque signature.
id : Permet de cibler un identifiant IP précis dans les paquets.
seq : Recherche un numéro de séquence TCP spécifique.
dsize : Cible les paquets selon la taille de leur charge utile.
content : Élément central des signatures, utilisé pour spécifier un motif à détecter dans le contenu du paquet
rev : Indique la révision ou la version de la signature. Ce nombre est incrémenté lors de chaque mise à jour.
offset : Sert à décaler le point de départ de la recherche de motif dans la charge outille

de paquet. [48]
La **figure 3.4** illustre ces éléments de l'option avec un exemple coloré de signature.

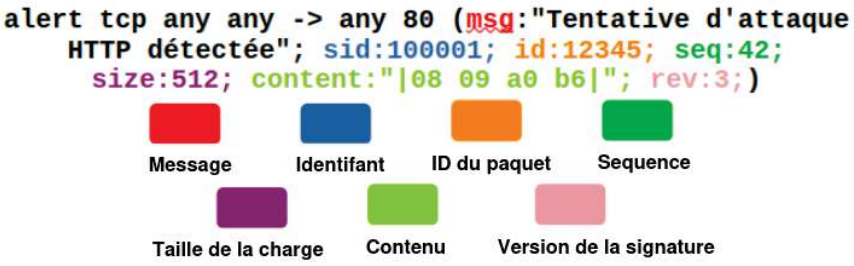


FIGURE 3.4 – Composant de l'option.

3.4 Génération des règles Suricata à partir des signatures identifiées

Au cours de notre étude, nous avons identifié plusieurs caractéristiques spécifiques aux réseaux Tor, JonDonym et Freenet.
Certaines de ces caractéristiques sont relativement simples à intégrer et à tester dans notre IDS, tandis que d'autres se révèlent beaucoup plus complexes, voire difficiles à exploiter directement.
Dans cette section, nous allons implémenter certains signatures capables de détecter une activité potentielle liée à l'utilisation de ces réseaux anonymes. Pour cela, nous avons opté Suricata. reconnu pour sa capacité à analyser le trafic réseau en s'appuyant sur des règles de détection personnalisées.
La première étape c'est de concevoir des signature adaptées à Tor, JonDonym et Freenet. Celles-ci définissent la direction du trafic ainsi que les motifs binaires (patterns hexadécimaux) permettent de reconnaître ces communications.
Les règles suricata sont organisées dans un dossier qui s'appelle "rules", il contient des règles se forme de texte standard, que Suricata utilise pour surveiller le réseau et exécute lors de sont démarrage.

3.4.1 Création des règles du navigateur Tor

Pour détecter le trafic généré par le navigateur Tor, nous avons procédé à une analyse fine des paquets TLS échangés, en particulier les messages **ClientHello** et **ServerHello**, qui contiennent des éléments caractéristiques tels que les **suites de chiffrement**, les **algorithmes de hachage**, ou encore les **extensions TLS** spécifiques.

En nous basant sur ces éléments, nous avons élaboré 12 règles de détection Suricata, capables d'identifier avec précision du protocole Tor sur le réseau.

Règle 1 :

La première règle vise à détecter une tentative via le port 9001, utilisé par le relais de garde.

```
alert tcp any any -> any 9001 (msg:"[ALERT] Tor connexion detected on
9001"; sid:1000001; rev:1;)
```

Règle 2 :

Cette règle vise à détecter l'usage spécifique de l'algorithme cryptographique RSA_PSS_PSS_SHA256, reconnu par la séquence hexadécimale |**08 09**|.

```
alert tls any any -> any 9001 (msg:"TLS Algorithm Detected :
RSA_PSS_PSS_SHA256 ";content:"|08 09|"; sid : 1000002; rev : 1;)
```

Règle 3 :

Cette règle cible l'identification de l'algorithme RSA_PSS_PSS_SHA384, présent lors de l'échange de paramètres cryptographiques. reconnu par le motif binaire spécifique |**08 0a**|.

```
alert tls any any -> any 9001 (msg:"TLS Algorithm Detected :
RSA_PSS_PSS8SHA384 ";content:"|08 0A|"; sid : 1000003; rev : 1;)
```

Règle 4 :

Cette règle permet d'identifier la présence de l'extension TLS "Encrypted_then_MAC", souvent utilisée par Tor. Sa détection constitue un indicateur fort d'une communication anonyme, identifier par la séquence hexadécimale |**00 16 00 00**|.

```
alert tls any any -> any 9001 (msg:"TLS Extension Encrypted then mac
detected ";content:"|00 16 00 00 |"; sid : 1000004; rev : 1;)
```

Règle 5 :

Cette règle détecte la suite de chiffrement |**00 ff**| dans le message clientHello, cette dernièr peut indiquer une tentative de négociation de chiffrement non standard au protocole Tor.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Cipher suite 00ff
detected";content:"|00 ff|";sid:1000005;rev:1;)
```

Règle 6 :

Cette règle permet d'identifier la présence de la suite de chiffrement |**00 33**| dans le message clientHello, utilisée lors de l'établissement de la connexion.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Cipher suite 00 33
detected";content:"|00 33|";sid:1000006;rev:1;)
```

Règle 7 :

Cette règle détecte la présence de la suite de chiffrement |**00 39**| dans le message clientHello. La présence de cette suite provoque un niveau de sécurité très élevé.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Cipher suite 00 39
detected";content:"|00 39|";sid:1000007;rev:1;)
```

Règle 8 :

Cette règle permet d'identifier l'algorithme de hachage référence par l'identifiant |**08 07**|, présent dans le message ClientHello.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm
08 07 detected";content:"|08 07|";sid:1000008;rev:1;)
```

Règle 9 :

Cette alerte vise à détecter l'usage de l'algorithme de hachage identifié par |**08 08**| dans le message ClientHello.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm
08 08 detected";content:"|08 08|";sid:1000009;rev:1;)
```

Règle 10 :

Cette règle détecte l'algorithme de hachage |**08 0a**| utilisé dans le message ClientHello.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm
08 0a detected";content:"|08 0a|";sid:1000010;rev:1;)
```

Règle 11 :

Cette alerte signale la présence de la version TLS 1.2 |**03 03**| dans le message ClientHello.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm
03 03 detected";content:"|03 03|";sid:1000011;rev:1;)
```

Règle 12 :

Cette alerte détecte l'utilisation de l'algorithme de hachage identifié par l'ID |**05 02**| dans le message ClientHello, ce qui peut refléter une préférence cryptographique propre aux configurations du navigateur Tor.

```
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm
05 02 detected";content:"|05 02|";sid:1000012;rev:1;)
```

3.4.2 Création des règles du navigateur JonDonym

Pour détecter le trafic généré par le réseau JonDonym, nous avons réalisé une observation ciblée des échanges TCP établies entre le client et le serveur Jondonym. Contrairement à Tor, ce réseau repose sur des relais intermédiaires fixes, avec des connexions souvent orientées vers des adresses IP connues.
En nous basant sur ces éléments, nous avons conçu cinq règles qui exploitent l'adresse IP de la cascade, des motifs hexadécimaux dans la charge utile, ainsi que certains champs des en-têtes TCP comme la taille ou la fenêtre. Ces signatures permettent d'identifier efficacement les tentatives de connexion vers le réseau JonDonym.

Règle 1 :
La première alerte permet de détecter une tentative de connexion vers le réseau JonDonym en surveillant l'adresse IP **141.76.72.133**, utilisée par l'un des relais connus.

```
alert ip any any -> any [141.76.72.133] (msg:"JonDonym connexion
detected";sid : 1000013;rev : 1;)
```

Règle 2 :
La deuxième alerte se déclenche lorsque la longueur totale |**05 8c**| de paquet correspond à celle fréquemment observée dans les échanges avec le serveur JonDonym.

```
alert tcp any any -> [141.76.72.133] any ( msg:"Total length detected
1420"; content:"|05 8c|";sid: 1000014;rev:1;)
```

Règle 3 :
La troisième alerte permet d'identifier la longueur totale |**04 0e**| de paquet correspond à celle fréquemment observée dans les échanges avec le serveur JonDonym.

```
alert tcp any any -> [141.76.72.133] any ( msg:"Total length detected
1038 "; content:"|04 0e|";sid: 1000015;rev:1;)
```

Règle 4 :
La quatrième alerte permet d'identifier la taille de la fenêtre particulière |**04 00**| utilisée dans les échanges avec le serveur JonDonym, ce qui est caractéristique de ce type de trafic.

```
alert tcp any any -> [141.76.72.133] any ( msg:"Window size detected
1024 "; content:"|04 00|";sid: 1000016;rev:1;)
```

Règle 5 :
La cinquième alerte permet d'identifier la taille de la fenêtre particulière |**01 f5**| utilisée dans les échanges avec le serveur JonDonym, ce qui est caractéristique ce type de trafic.

```
alert tcp any any -> [141.76.72.133] any ( msg:"Window size detected
501"; content:"|01 f5|";sid: 1000017;rev:1;)
```

3.4.3 Création des règles du navigateur Freenet

Pour détecter le trafic généré par le réseau Freenet, nous avons ciblé l'analyse des paquets UDP et IP échangés entre les nœuds du réseau. Ce dernier utilise un modèle de réseau décentralisé basé sur le partage de données P2P.
Nous avons élaboré deux règles de détection : l'une ciblant le port de communication en mode Opennet, et l'autre portant sur la taille des paquets échangés.
Règle 1 :
La première règle, **"Freenet connexion detected"**, détecte les connexions entrantes sur le port 17910, à fin d'établir la communication entre les nœuds.

```
alert udp any any -> any 17910 (msg:"Freenet connexion detected";sid :
1000017;rev : 1;)
```

Règle 2 :
La deuxième,**"Packet size exceeds 1280 bytes"**, cible les paquets qui ne dépassent pas 1280 octets qui sont souvent observés dans les communications .

```
alert ip any any -> any 17910 (msg:"[ALERT] Packet size exceeds 1280
bytes"; dsize:<1280 ;sid: 1000018 ;rev:1;)
```

3.5 Mise en place du test

3.5.1 Architecture de détection finale

L'architecture de notre plateforme de test repose sur deux machines principales connectées en réseau local. le poste client (adresse IP : 10.42.0.174), simule l'utilisateur final et intègre les outils d'anonymat Tor, JonDonym et Freenet. Le trafic de ce client transite par un proxy Squid installé sur le poste serveur (adresse IP : 10.42.0.1).
Ce serveur héberge également Suricata, qui agit en tant qu'un IDS et capture le flux réseau en temps réel, ainsi que Splunk pour la journalisation centralisée des événements.

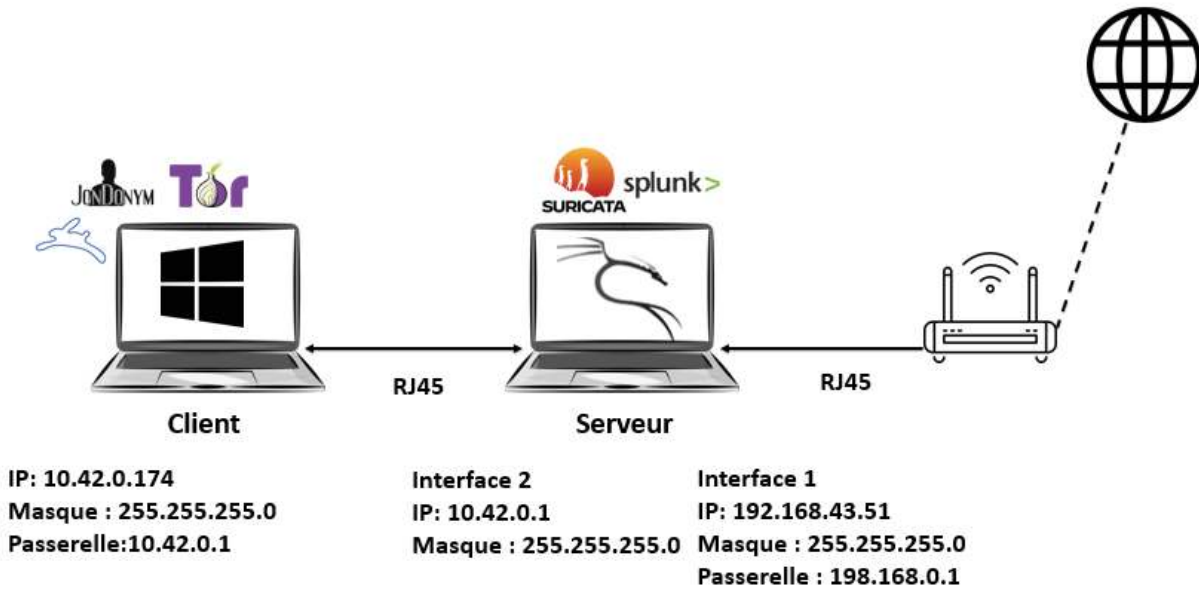


FIGURE 3.5 – Architecture de test.

3.5.2 Implémentation des empreintes

Pour implémenter nos règles dans Suricta, nous avons ajouté nos signatures personnalisées dans le fichier <**suricata.rules**>. Nous avons ensuite configuré le fichier <**suricata.yaml**> pour prendre en compte ce fichier de règles.
La **figure 3.6** illustre les règles utilisées.

```
GNU nano 8.3                    /etc/suricata/rules/suricata.rules *
#Tor-rules
alert tcp any any  -> any 9001 (msg:"[ALERT ] Tor connexion detected on 9001 "; sid : 1000001; rev : 1;)
alert tls any any   -> any 9001 (msg:"TLS Algorithm Detected :RSA_PSS_PSS_SHA256 ";content:"|08 09|"; sid : 1000002; rev : 1;)
alert tls any any  -> any 9001 (msg:"TLS Algorithm Detected :RSA_PSS_PSS8SHA384 ";content:"|08 0A|"; sid : 1000003; rev : 1;)
alert tls any any   -> any 9001  (msg:"TLS Extension Encrypted then mac detected ";content:"|00 16 00 00 |"; sid : 1000004; rev : 1;)
alert tls any any -> any 9001 (msg:"TLS client Hello-Cipher suite 00ff detected";content:"|00 ff|";sid:1000005;rev:1;)
alert tls any any -> any 9001 (msg:"TLS client Hello-Cipher suite 00 33 detected";content:"|00 33|";sid:1000006;rev:1;)
alert tls any any -> any 9001 (msg:"TLS client Hello-Cipher suite 00 39 detected";content:"|00 39|";sid:1000007;rev:1;)
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm 08 07 detected";content:"|08 07|";sid:1000008;rev:1;)
alert tls any any -> any 9001(msg:"TLS client Hello-Hashing algorithm 08 08 detected";content:"|08 08|";sid:1000009;rev:1;)
alert tls any any -> any 9001(msg:"TLS client Hello-Hashing algorithm 08 0a detected";content:"|08 0a|";sid:1000010;rev:1;)
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm 03 03 detected";content:"|03 03|";sid:1000011;rev:1;)
alert tls any any -> any 9001 (msg:"TLS client Hello-Hashing algorithm 05 02 detected";content:"|05 02|";sid:1000012;rev:1;)
#JonDonym-rules
alert ip  any any -> any [141.76.72.133] (msg:"JonDonym connexion detected";sid : 1000013;rev : 1;)
alert tcp any any -> [141.76.72.133] any ( msg:"Total length detected 1420"; content:"|05 8c|";sid: 1000014;rev:1;)
alert tcp any any -> [141.76.72.133] any ( msg:"Total length detected 1038 "; content:"|04 0e|";sid: 1000015;rev:1;)
alert tcp any any -> [141.76.72.133] any ( msg:"Window size detected 1024 "; content:"|04 00|";sid: 1000016;rev:1;)
alert tcp any any -> [141.76.72.133] any ( msg:"Window size detected 501"; content:"|01 f5|";sid: 1000017;rev:1;)
#Freenet-rules
alert udp any any -> any 17910 (msg:"Freenet connexion detected";sid : 1000018;rev : 1;)
alert ip any any -> any 17910 (msg:"[ALERT] Packet size exceeds 1280 bytes"; dsize:<1280 ;sid: 1000019 ;rev:1;)
```

FIGURE 3.6 – Les règles implémentées dans le fichier **suricata.rules**.

Une fois les règles ajoutées et le fichier **suricata.yaml** correctement configuré, nous avons lancé Suricata en mode détection à l'aide de la commande :

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

3.5.3 Splunk

3.5.3.1 Définition

Splunk est un logiciel utilisé pour la gestion des informations et des événements de sécurité (SIEM). il permet de collecter, d'analyser et de visualiser les données générées par divers systèmes, y compris les alertes générées par Suricata. Grace à son interface web intuitive, Splunk offre des outils avancés permettant de rechercher, examiner et analyser les alertes en temps réel, offrant ainsi une compréhension approfondie des événements de sécurité et des informations essentielles pour la prise de décision. [49]

3.5.3.2 Configuration Splunk

Pour faire l'intégration du Splunk avec Suricata, nous avons d'abord créé un compte gratuit sur le site Splunk pour pouvoir télécharger logiciel. Une fois l'installation terminée, nous avons lancé le serveur Splunk, en utilisant la commande :

```
sudo /opt/splunk/bin/splunk start
```

Accessible via l'adresse **https ://10.42.0.1 :8000/**.
La **figure 3.7** illustre le profil de Splunk.

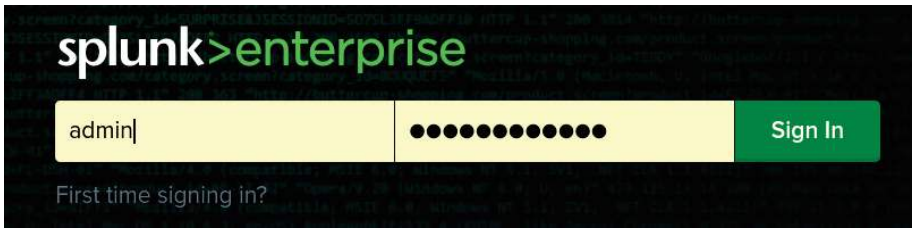


FIGURE 3.7 – Consultation des journaux dans Splunk.

Nous avons ensuite ajouté le module complémentaire **Suricata JSON Alerts** pour permettre à Splunk de reconnaître et analyser les journaux générés par Suricata. Enfin, pour afficher les alertes de manière clair et structurée, nous avons utilisé le filtre suivant dans l'interface de recherche :

```
index=suricata sourcetype=_json event_type=alert | table _time src_ip
src_port dest_ip dest_port alert.signature.
```

Ce filtre permet de visualiser l'heure d'alerte, les adresse IP, les ports utilisés, ainsi que la signature déclenchée.

3.5.4 Scénarios de test

Afin de vérifier l'efficacité de nos signatures et leurs visibilités dans splunk, nous avons élaboré plusieurs scénarios de test simulant l'usage réel de réseaux anonymes. Deux types de tests ont été réalisés :

a. Navigateur classique : nous avons utilisé le navigateur FireFox sur le poste client pour nous assurer que Suricata ne génère pas de fausses alertes lors d'un trafic web habituel (faux positifs).

b. Utilisation de réseaux anonymes : nous avons ensuite testé la navigation via le navigateur Tor, la connexion au service JonDonym, ainsi que le lancement du réseau Freenet, afin d'observer si Suricata parvient à détecter correctement les signatures associées à chacun de ces réseaux.

3.5.5 Résultats obtenus par l'IDS

Après l'exécution de nos scénarios de test, nous avons observé les alertes générées par Suricata dans l'interface Splunk.

a. Résultats 1 : Après avoir lancé Suricata et navigué au différant sites web avec le navigateur FireFox, nous avons constaté qu'aucune fausse alerte n'a été générée.

b. Résultats 2 : Lors de l'utilisation des réseaux anonymes , Suricata a correctement généré des alertes correspondant au signatures.

3.5.5.1 Résultats pour Tor

Dès le lancement du navigateur Tor, nous avons observé que Suricata génère simultanément douze alertes, correspondant aux règles spécifiques.

La **figure 3.8** met en évidence les résultats issus du processus de détection.

```
05/18/2025-12:04:42.868244  [**] [1:1000001:1] [ALERT ] Tor connexion detected on 9001  [**] [Classification: (null)] [Priority: 3] {TC
P} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000002:1] TLS Algorithm Detected :RSA_PSS_PSS_SHA256  [**] [Classification: (null)] [Priority: 3]
{TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000003:1] TLS Algorithm Detected :RSA_PSS_PSS8SHA384  [**] [Classification: (null)] [Priority: 3]
{TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000004:1] TLS Extension Encrypted then mac detected  [**] [Classification: (null)] [Priority: 3] {
TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000006:1] TLS client Hello-Cipher suite 00 33 detected [**] [Classification: (null)] [Priority: 3]
 {TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000007:1] TLS client Hello-Cipher suite 00 39 detected [**] [Classification: (null)] [Priority: 3]
 {TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000008:1] TLS client Hello-Hashing algorithm 08 07 detected [**] [Classification: (null)] [Priorit
y: 3] {TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000009:1] TLS client Hello-Hashing algorithm 08 08 detected [**] [Classification: (null)] [Priorit
y: 3] {TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000010:1] TLS client Hello-Hashing algorithm 08 0a detected [**] [Classification: (null)] [Priorit
y: 3] {TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000011:1] TLS client Hello-Hashing algorithm 03 03 detected [**] [Classification: (null)] [Priorit
y: 3] {TCP} 10.42.0.174:51325 -> 88.99.142.177:9001
05/18/2025-12:04:43.209775  [**] [1:1000012:1] TLS client Hello-Hashing algorithm 05 02 detected [**] [Classification: (null)] [Priorit
```

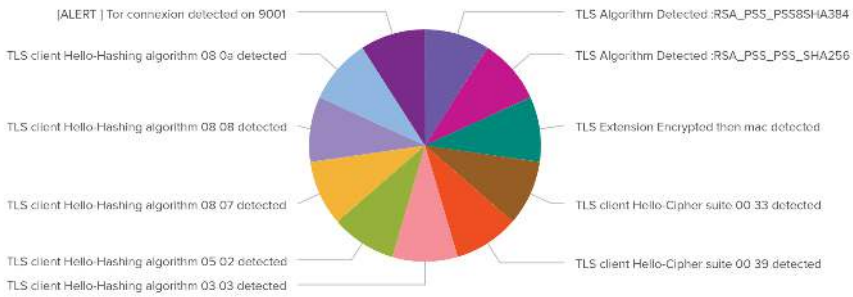
FIGURE 3.8 – Détection de l'utilisation du navigateur Tor sur Suricata.

nous observons que Splunk affiche uniquement les alertes correspondant aux règles de détection configurées dans Suricata.
La **figure 3.9** illustre la manière dont les alertes sont présentées dans l'interface de Splunk.

_time	src_ip	src_port	dest_ip	dest_port	alert.signature
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS client Hello-Hashing algorithm 05 02 detected
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS client Hello-Hashing algorithm 03 03 detected
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS client Hello-Hashing algorithm 08 0a detected
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS client Hello-Hashing algorithm 08 08 detected
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS client Hello-Hashing algorithm 08 07 detected
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS client Hello-Cipher suite 00 39 detected
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS Extension Encrypted then mac detected
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS Algorithm Detected :RSA_PSS_PSS8SHA384
2025-05-18 12:19:37.806	10.42.0.174	51377	88.99.142.177	9001	TLS Algorithm Detected :RSA_PSS_PSS_SHA256
2025-05-18 12:19:37.521	10.42.0.174	51377	88.99.142.177	9001	[ALERT] Tor connexion detected on 9001

FIGURE 3.9 – Surveillance de Tor via Splunk.

Comme le montre la **figure 3.10**, chaque type d'alerte représente **9,09%** du total, ce qui indique une répartition uniforme des alertes enregistrées.

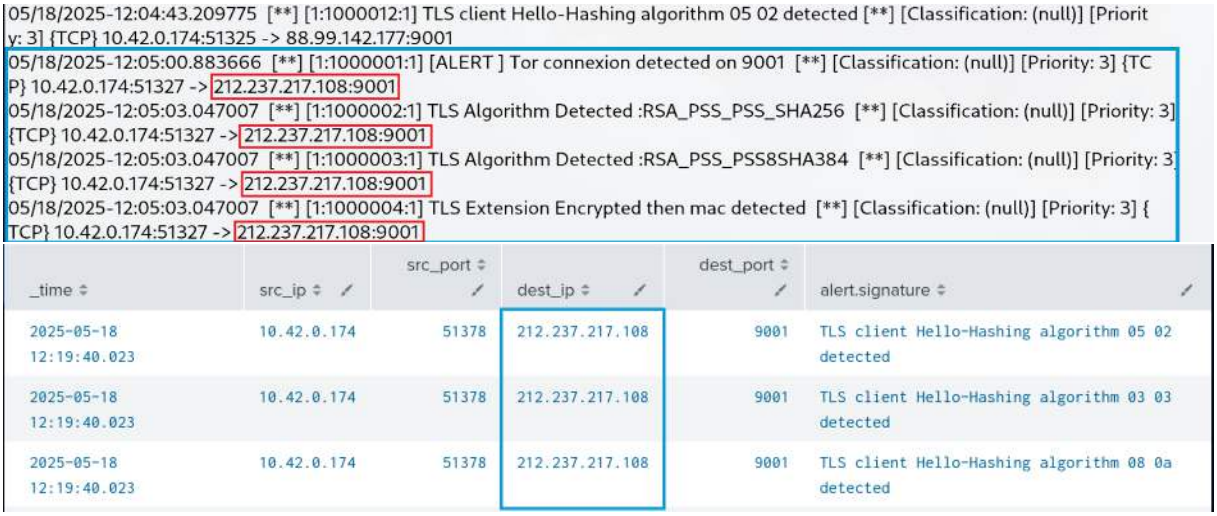


alert.signature	count	percentage	total
TLS Algorithm Detected :RSA_PSS_PSS8SHA384	11	9.09	121
TLS Algorithm Detected :RSA_PSS_PSS_SHA256	11	9.09	121
TLS Extension Encrypted then mac detected	11	9.09	121
TLS client Hello-Cipher suite 00 33 detected	11	9.09	121
TLS client Hello-Cipher suite 00 39 detected	11	9.09	121
TLS client Hello-Hashing algorithm 03 03 detected	11	9.09	121
TLS client Hello-Hashing algorithm 05 02 detected	11	9.09	121
TLS client Hello-Hashing algorithm 08 07 detected	11	9.09	121
TLS client Hello-Hashing algorithm 08 08 detected	11	9.09	121
TLS client Hello-Hashing algorithm 08 0a detected	11	9.09	121
[ALERT] Tor connexion detected on 9001	11	9.09	121

FIGURE 3.10 – Analyse des pourcentages d'alertes détectées.

Remarque :

Nous avons observé un changement d'adresse du circuit Tor, comme le montre la **figure 3.11**, avec des connexions fréquentes à l'adresse **IP 212.237.217.108** .



_time	src_ip	src_port	dest_ip	dest_port	alert.signature
2025-05-18 12:19:40.023	10.42.0.174	51378	212.237.217.108	9001	TLS client Hello-Hashing algorithm 05 02 detected
2025-05-18 12:19:40.023	10.42.0.174	51378	212.237.217.108	9001	TLS client Hello-Hashing algorithm 03 03 detected
2025-05-18 12:19:40.023	10.42.0.174	51378	212.237.217.108	9001	TLS client Hello-Hashing algorithm 08 0a detected

FIGURE 3.11 – Changement de circuit Tor.

3.5.5.2 Test de fiabilité des règles TLS dans la détection du trafic Tor

À l'issue de la première phase de test, la **figure 3.10** a présenté une répartition équilibrée des alertes déclenchées par les règles Suricata configurées pour le navigateur

Tor.
Une observation importante a été faite lors de l'analyse du trafic comme le montre la **figure 3.11**, le navigateur Tor a établi des connexions vers une nouvelle adresse **IP 212.237.217.108**, différente de celles rencontrées lors des premiers tests.
L'adresse **IP 212.237.217.108** , observée dans le trafic réseau, correspond bien à une adresse utilisée par le réseau Tor, comme illustré dans la **figure 3.12**.

FIGURE 3.12 – Vérification de l'adresse de destination dans « metrics.torproject.org ».

Pour cela, un ensemble de nouvelles règles Suricata centrées sur les caractéristiques du protocole TLS a été élaboré. Ces règles ne s'appuient plus sur le port 9001, mais uniquement sur des motifs binaires identifiables dans les messages **Client-Hello**, les suites de chiffrement, les extensions et algorithmes de hachage couramment utilisés par le navigateur Tor.
L'ensemble des règles testées est présenté ci-dessous :

```
alert tls any any -> any any (msg:"TLS Algorithm Detected
:RSA_PSS_PSS_SHA256 ";content:"|08 09|"; sid : 1000002; rev : 1;)
alert tls any any -> any any (msg:"TLS Algorithm Detected
:RSA_PSS_PSS8SHA384 ";content:"|08 0A|"; sid : 1000003; rev : 1;)
alert tls any any -> any any (msg:"TLS Extension Encrypted then mac
detected ";content:"|00 16 00 00 |"; sid : 1000004; rev : 1;)
alert tls any any -> any any (msg:"TLS client Hello-Cipher suite 00ff
detected";content:"|00 ff|";sid:1000005;rev:1;)
alert tls any any -> any any (msg:"TLS client Hello-Cipher suite 00 33
detected";content:"|00 33|";sid:1000006;rev:1;)
alert tls any any -> any any (msg:"TLS client Hello-Cipher suite 00 39
detected";content:"|00 39|";sid:1000007;rev:1;)
```

```
alert tls any any -> any any (msg:"TLS client Hello-Hashing algorithm 08
07 detected";content:"|08 07|";sid:1000008;rev:1;)
alert tls any any -> any any (msg:"TLS client Hello-Hashing algorithm 08
08 detected";content:"|08 08|";sid:1000009;rev:1;)
alert tls any any -> any any (msg:"TLS client Hello-Hashing algorithm 08
0a detected";content:"|08 0a|";sid:1000010;rev:1;)
alert tls any any -> any any (msg:"TLS client Hello-Hashing algorithm 03
03 detected";content:"|03 03|";sid:1000011;rev:1;)
alert tls any any -> any any (msg:"TLS client Hello-Hashing algorithm 05
02 detected";content:"|05 02|";sid:1000012;rev:1;)
```

Lors du test, l'ensemble des onze alertes ont bien été détectées. Cependant, on remarque que la règle « **TLS client Hello-Hashing algorithm 03 03 detected** » se déclenche plusieurs fois, comme le montre la **figure 3.13**.

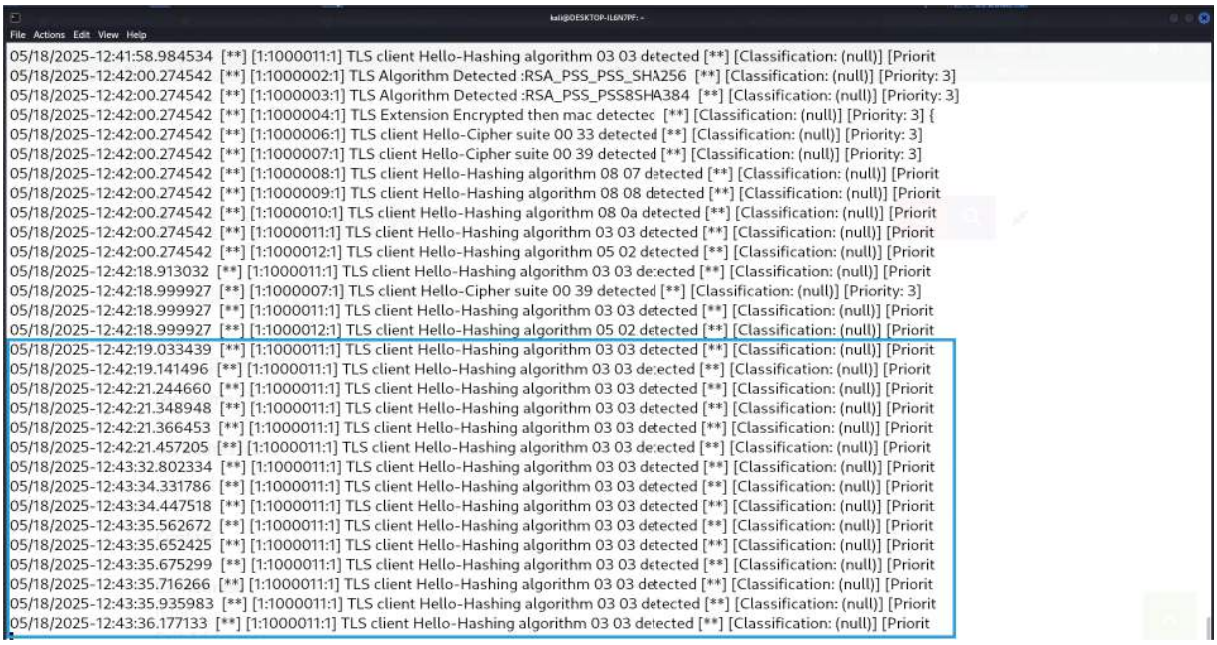


FIGURE 3.13 – Alertes générées lors du deuxième test de détection du Tor.

Pour une visualisation plus détaillée des alertes générées, la **figure 3.14** présente l'affichage correspondant dans l'interface de Splunk.

_time	src_ip	src_port	dest_ip	dest_port	alert.signature
2025-05-18 12:39:34.646	10.42.0.174	51483	52.182.143.210	443	TLS client Hello-Hashing algorithm 03 03 detected
2025-05-18 12:39:33.126	10.42.0.174	51482	23.62.180.198	443	TLS client Hello-Hashing algorithm 03 03 detected
2025-05-18 12:39:33.126	10.42.0.174	51482	23.62.180.198	443	TLS client Hello-Cipher suite 00 33 detected
2025-05-18 12:39:31.779	10.42.0.174	51481	20.42.65.89	443	TLS client Hello-Hashing algorithm 03 03 detected
2025-05-18 12:39:31.779	10.42.0.174	51481	20.42.65.89	443	TLS client Hello-Cipher suite 00 33 detected
2025-05-18 12:39:31.772	20.42.65.89	443	10.42.0.174	51481	TLS client Hello-Hashing algorithm 03 03 detected
2025-05-18 12:39:20.521	10.42.0.174	51471	142.202.51.68	9001	TLS client Hello-Hashing algorithm 08 0a detected
2025-05-18 12:39:20.521	10.42.0.174	51471	142.202.51.68	9001	TLS client Hello-Hashing algorithm 08 08 detected
2025-05-18 12:39:20.521	10.42.0.174	51471	142.202.51.68	9001	TLS client Hello-Hashing algorithm 08 07 detected

FIGURE 3.14 – Affichage sur Splunk.

Afin de confirmer que les adresses IP de destination observées proviennent réellement du réseau Tor (et qu'il ne s'agit pas de faux positif), une vérification a été effectuée à l'aide du site Tor Metrics, comme illustré dans la **figure 3.15**.

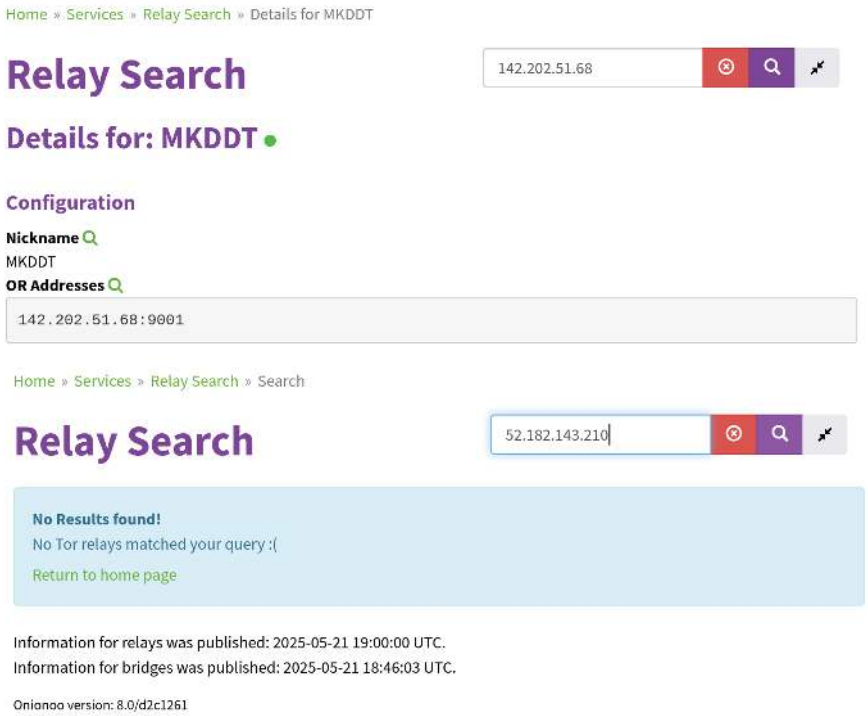


FIGURE 3.15 – Vérification des adresses IP via « metrics.torproject.org »

Une vérification a également été effectuée pour les adresses **IP 23.62.180.198** et **20.42.65.89**, et il a été constaté qu'elles ne sont pas associées au réseau Tor.

3.5.5.3 Résultats pour JonDonym

Lors du test avec le réseau JonDonym, nous avons pu détecter quatre alertes sur les cinq attendus.
La **figure 3.16** illustre les observations tirées de la phase de détection.

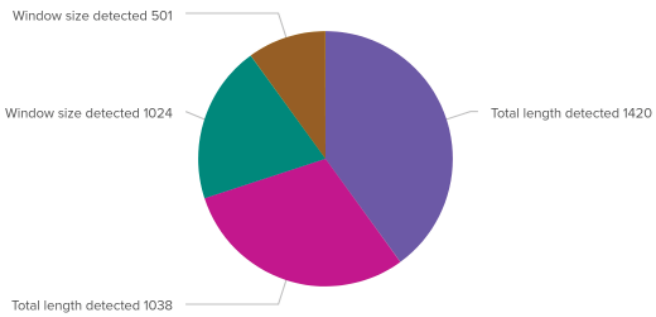
Figure 3.16 – Détection de l'utilisation du service JonDonym.

Dans le cas de JonDonym, cinq règles ont été configurées dans Suricata, mais une seule n'a généré aucune alerte. La **figure 3.17** présente l'affichage des alertes correspondantes dans l'interface de Splunk.

_time	src_ip	src_port	dest_ip	dest_port	alert.signature
2025-05-18 11:48:49.695	10.42.0.174	50672	141.76.72.133	443	Window size detected 1024
2025-05-18 11:45:49.623	10.42.0.174	50672	141.76.72.133	443	Window size detected 1024
2025-05-18 11:42:49.038	10.42.0.174	50672	141.76.72.133	443	Window size detected 501
2025-05-18 11:31:34.704	10.42.0.174	50672	141.76.72.133	443	Window size detected 1024
2025-05-18 11:29:05.403	10.42.0.174	50672	141.76.72.133	443	Total length detected 1038
2025-05-18 11:17:33.622	10.42.0.174	50672	141.76.72.133	443	Total length detected 1420
2025-05-18 11:14:34.137	10.42.0.174	50672	141.76.72.133	443	Total length detected 1420
2025-05-18 11:11:34.014	10.42.0.174	50672	141.76.72.133	443	Total length detected 1038
2025-05-18 11:08:33.873	10.42.0.174	50672	141.76.72.133	443	Total length detected 1038

FIGURE 3.17 – Surveillance de JonDonym via Splunk.

Comme le montre la **figure 3.18**, les alertes pour Jondonym sont réparties de manière inégale, avec une prédominance des signatures « Total length detected 1420 » (40 %), suivies par « Total length detected 1038 » (30 %). Les alertes restantes, « Window size detected 1024 » et « Window size detected 501 », représentent respectivement 20 % et 10 %, ce qui reflète une diversité modérée des types d'alertes enregistrées.



alert.signature	count	percentage	total
Total length detected 1420	4	40.00	10
Total length detected 1038	3	30.00	10
Window size detected 1024	2	20.00	10
Window size detected 501	1	10.00	10

FIGURE 3.18 – Répartition des Alertes pour Jondonym.

3.5.5.4 Résultats pour Freenet

Lors de l'exécution de freenet, Suricata a généré de nombreuses alertes liées à des paquets UDP dont la taille ne dépasse pas 1280 octets, ce qui correspond à la signature mise en place.
La **figure 3.19** présente l'affichage des alertes générées.

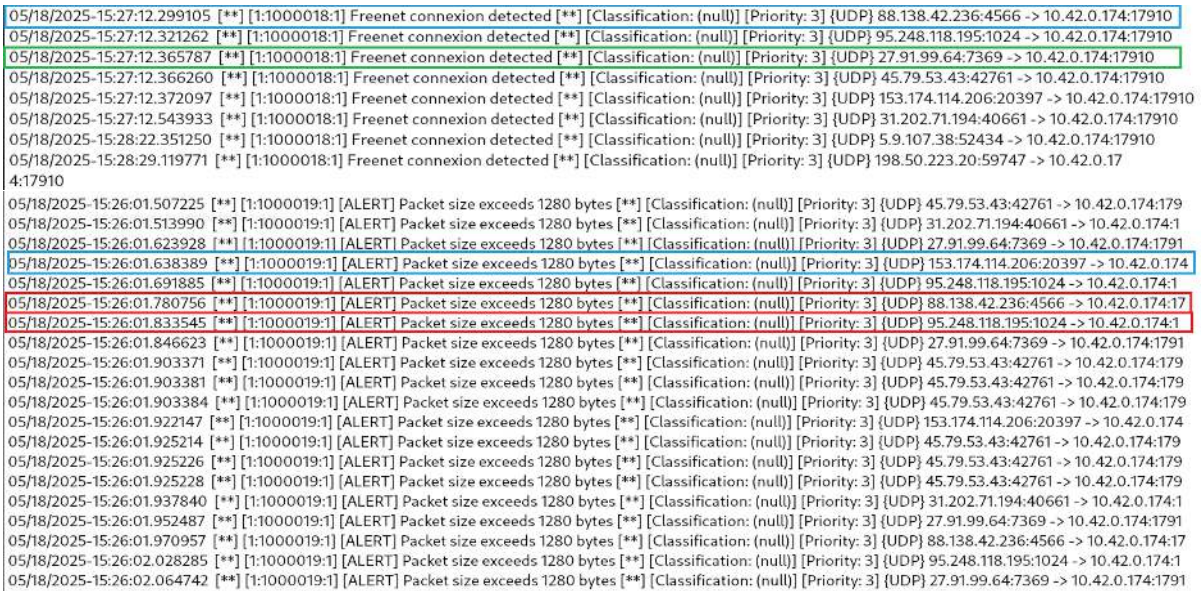


FIGURE 3.19 – Détection de l'utilisation du réseau Freenet.

L'analyse du trafic Freenet a généré des alertes visibles dans Splunk. Comme le montre la **figure 3.20**, ces alertes correspondent aux signatures liées à ce protocole. Le système de détection a donc bien identifié l'activité Freenet sur le réseau.

_time	src_ip	src_port	dest_ip	dest_port	alert.signature
2025-05-18 15:27:12.365	27.91.99.64	7369	10.42.0.174	17910	Freenet connexion detected
2025-05-18 15:27:12.321	95.248.118.195	1024	10.42.0.174	17910	Freenet connexion detected
2025-05-18 15:27:12.299	88.138.42.236	4566	10.42.0.174	17910	Freenet connexion detected
2025-05-18 15:27:09.325	153.174.114.206	20397	10.42.0.174	17910	[ALERT] Packet size exceeds 1280 bytes
2025-05-18 15:27:09.325	153.174.114.206	20397	10.42.0.174	17910	[ALERT] Packet size exceeds 1280 bytes
2025-05-18 15:27:09.325	95.248.118.195	1024	10.42.0.174	17910	[ALERT] Packet size exceeds 1280 bytes
2025-05-18 15:27:09.310	153.174.114.206	20397	10.42.0.174	17910	[ALERT] Packet size exceeds 1280 bytes

FIGURE 3.20 – Surveillance de Freenet via Splunk.

Comme le montre la **figure 3.21**, les alertes liées à Freenet sont majoritairement dues à des paquets qui ne depassent pas 1280 octets, représentant **99,79%** des cas (3785 occurrences). Les connexions Freenet détectées ne constituent que **0,21%** des alertes (8 occurrences).

alert.signature	count	percentage	total
[ALERT] Packet size exceeds 1280 bytes	3785	99.79	3793
Freenet connexion detected	8	0.21	3793

Figure 3.21 – Répartition des Alertes pour Freenet.

3.6 Discussion des résultats obtenus par l'IDS

-Les règles basées sur le port 9001, telles que **[ALERT] Tor connexion detected on 9001**, ont permis une détection correcte des connexions Tor, avec une répartition équilibrée des alertes à hauteur de **9,09% par signature**, soit un total de 11 règles actives. Cependant, certaines signatures spécifiques, notamment **TLS client Hello-Hashing algorithm 03 03** et **TLS client Hello-Cipher suite 00 33 detected**, ont été déclenchées à plusieurs reprises en dehors du trafic Tor identifié. Ces résultats suggèrent une certaine sensibilité excessive de ces règles, qui peut conduire à des **faux positifs** dans un environnement où d'autres applications utilisent également TLS.
-Les règles appliquées au trafic JonDonym, basées sur l'adresse IP **141.76.72.133** ainsi que sur des tailles de paquets typiques, ont généré quatre alertes sur cinq prévues. La répartition est relativement bien structurée : **40%** des alertes correspondent à une longueur de 1420 octets, **30%** à 1038 octets, **20%** à une taille de fenêtre de 1024, et **10%** à 501 octets. Aucun faux positif n'a été observé pendant les tests. Toutefois, cette détection repose fortement sur des paramètres statiques (adresse IP fixe, tailles précises), ce qui limite sa robustesse à long terme, notamment si le réseau JonDonym évolue.
-Pour Freenet, la détection s'est principalement appuyée sur l'analyse de la taille des paquets UDP. La règle **Packet size exceeds 1280 bytes** a généré la quasi-totalité des alertes, représentant **99,79%** des cas (soit 3785 occurrences), tandis que la détection explicite des connexions sur le port **17910** n'a concerné que **0,21%** des alertes (8 occurrences). Aucune fausse alerte n'a été relevée, ce qui montre une détection fiable dans ce contexte. Cependant, la concentration sur un seul critère (la taille du paquet) pourrait, dans un environnement plus varié, générer des faux positifs.

3.7 Conclusion

Dans ce chapitre, nous avons expliqué comment configurer et utiliser Suricata comme système de détection d'intrusion réseau (NIDS) sous Kali Linux.
Nous avons montré qu'il est possible de détecter en temps réel l'usage des réseaux anonymes tels que Tor, JonDonym et Freenet, en s'appuyant sur les règles que nous avons élaborées.
L'intégration avec Splunk nous a permis de visualiser et d'analyser les alertes générées par Suricata. Ces alertes peuvent inclure des faux positifs, ce qui rend nécessaire l'analyse du contexte de chaque détection afin de confirmer une véritable tentative d'accès à ces réseaux.
En conclusion, les tests réalisés ont permis de valider la pertinence et la robustesse de nos règles de détection.

Conclusion Générale

L'essor des réseaux anonymes tels que Tor, JonDonym et Freenet pose aujourd'hui un véritable défi pour la sécurité des systèmes d'information, notamment dans les environnements professionnels. Bien qu'utilisés à des fins légitimes comme la protection de la vie privée, ces outils peuvent aussi permettre à des utilisateurs malintentionnés de contourner les mécanismes de sécurité, d'accéder à des contenus illégaux ou de faire circuler des données sensibles à l'insu des entreprises. La problématique centrale de ce travail était donc de savoir s'il est possible de détecter l'utilisation de ces réseaux dans un réseau d'entreprise.

Pour répondre à cette question, nous avons proposé une solution basée sur l'analyse du trafic réseau et la détection des signatures caractéristiques laissées par ces outils lors de leur utilisation. En identifiant les empreintes spécifiques de Tor, JonDonym et Freenet, il devient possible de configurer un système de détection d'intrusion capable d'émettre des alertes en temps réel.

La méthodologie adoptée s'est articulée en plusieurs phases. Nous avons d'abord étudié le fonctionnement des réseaux anonymes et leurs particularités. Ensuite, un environnement de test a été mis en place pour capturer le trafic généré par ces outils. À l'aide de **Wireshark (analyseur de paquets)**, nous avons analysé ce trafic pour extraire des éléments distinctifs (comme les ports, les suites de chiffrement TLS ou les tailles de paquets). Ces signatures ont ensuite servi à créer des règles personnalisées dans **Suricata (système de détection d'intrusion)**.

Les résultats obtenus indiquent que le système a permis d'identifier certaines tentatives de connexion aux réseaux anonymes, y compris en présence de techniques de dissimulation telles que l'utilisation de proxies. Néanmoins, cette détection reste dépendante de la qualité et de l'actualité des signatures mises en place. Par ailleurs, l'intégration de Splunk a facilité la consultation et l'analyse des alertes, en offrant un

support de visualisation centralisé et structuré.

Cependant, certaines limites ont été identifiées. Les signatures peuvent évoluer dans le temps avec les mises à jour des outils anonymes, ce qui rend nécessaire une mise à jour régulière des règles. Par ailleurs, il est parfois difficile de distinguer certains flux chiffrés légitimes de ceux issus des réseaux anonymes, ce qui peut entraîner des faux positifs.

Enfin, ce travail ouvre plusieurs perspectives. Il pourrait être enrichi par l'intégration d'algorithmes de classification, par l'automatisation de l'extraction des signatures ou par l'étude de nouveaux outils émergents. L'amélioration continue de ces méthodes est essentielle pour garantir une cybersécurité efficace face à des techniques d'anonymat en constante évolution.

Bibliographie

[1] L. Rédaction. (2023, Oct.) Confidentialité, anonymat & pseudonymat sur internet : c'est quoi ? Datackathon. [Online]. Available : https://www.datackathon.com/confidentialite-anonymat-pseudonymat-sur-internet/

[2] C. d'avocats Mathias. (2023) Deep web et dark web : Comprendre les différences et les enjeux. [Online]. Available : https://www.avocats-mathias.com/technologies-avancees/deep-web-dark-web

[3] Cybernod. (2024, november) Exploring the hidden layers of the internet : Surface web, deep web, and dark web. [Online]. Available : https://blog.cybernod.com/2024/11/exploring-the-hidden-layers-of-the-internet-surface-web-deep-web-and-dark-web/

[4] D. Moore and T. Rid, "Cryptopolitik and the darknet," *Survival*, vol. 58, no. 1, pp. 7–38, 2016. [Online]. Available : https://doi.org/10.1080/00396338.2016.1142085

[5] A. Ajdini, "Étude et conception d'un service assurant l'anonymat," Travail de Bachelor, Haute École de Gestion de Genève (HEG-GE), Filière informatique de gestion, Sep. 30 2019, 30 septembre 2019.

[6] Le Big Data. (2025) Tor : tout savoir sur le réseau anonyme. [Online]. Available : https://www.lebigdata.fr/tor-tout-savoir

[7] Ciberseguridad.com. (n.d.) ¿qué es i2p ? [Online]. Available : https://ciberseguridad.com/guias/recursos/i2p/#%C2%BFQue_es_I2P

[8] Framalibre. (2025) Freenet : Un logiciel libre pour une communication anonyme et sans censure. [Online]. Available : https://framalibre.org/notices/freenet.html

[9] Hyphanet Project. (2023) Freenet renamed to hyphanet. [Online]. Available : https://www.hyphanet.org/freenet-renamed-to-hyphanet.html

[10] Technische Universität Dresden. (n.d.) Jondonym - anonymity on the internet. [Online]. Available : https://anon.inf.tu-dresden.de/help/jap_help/en/help/jondonym.html

[11] Privacy Guides. (2025) Aperçu de tor : Comprendre les nœuds d'entrée. [Online]. Available : https://www.privacyguides.org/fr/advanced/tor-overview/#le-nud-dentree

[12] WebsiteRating. (2025) Tor vs vpn : Quelle est la meilleure solution pour votre confidentialité en ligne ? [Online]. Available : https://www.websiterating.com/fr/blog/vpn/tor-vs-vpn/

[13] H. Wei, L. Li, and Y. Yang, "Research on key technologies of anonymous network tor and i2p," *Computer Science and Application*, vol. 9, no. 7, pp. 1296–1308, 2019. [Online]. Available : https://doi.org/10.12677/csa.2019.97146

[14] ResearchGate. (2025) The concept of onion routing. [Online]. Available : https://www.researchgate.net/figure/The-concept-of-onion-routing_fig1_306357723

[15] W3R One. (2025) Aes, rsa et au-delà : un aperçu des algorithmes de chiffrement dans la blockchain. [Online]. Available : https://w3r.one/fr/blog/blockchain-web3/cryptographie/algorithmes-chiffrement/aes-rsa-et-au-dela-un-apercu-des-algorithmes-de-chiffrement-dans-la-blockchain

[16] J. Mueller, "Analysis of the i2p network," 2016. [Online]. Available : https://www.benoist.ch/research/thesis/FS16/Analysis_of_the_I2P_Network-Bachelorarbeit_Jens_Mueller.pdf

[17] OnlineSim. Dark web, partie 3 : qu'est-ce que freenet ? [Online]. Available : https://onlinesim.io/fr/instructions/dark-web-partie-3-quest-ce-que-freenet

[18] LinuxFr.org. Freenet. [Online]. Available : https://linuxfr.org/wiki/freenet

[19] K. C. N. Halvemaan, "Freenet darknet mapping," Master's thesis, University of Amsterdam, 2017.

[20] VPN Unlimited. Anonymat et cybersécurité : Pourquoi l'anonymat est-il important ? [Online]. Available : https://www.vpnunlimited.com/fr/help/cybersecurity/anonymity

[21] Use Your Law. La place de l'anonymat dans la lutte contre la cybercriminalité. [Online]. Available : https://www.useyourlaw.com/la-place-de-lanonymat-dans-la-lutte-contre-la-cybercriminalite/

[22] Kaspersky. Que sont le deep web et le dark web ? [Online]. Available : https://www.kaspersky.fr/resource-center/threats/deep-web

[23] K. Bedouet. (2025, feb) Qu'est-ce qu'une attaque sybil et comment s'en protéger ? Blog BIM Finance. [Online]. Available : https://www.bim-finance.com/blog/attaque-sybil

[24] CyberProof. (2025) An analysis of the security risks posed by tor browser. [Online]. Available : https://www.cyberproof.com/blog/an-analysis-of-the-security-risks-posed-by-tor-browser/

[25] The Verge. (2021, nov) An alleged member of the revil ransomware gang was arrested in poland. [Online]. Available : https://www.theverge.com/2021/11/8/22770701/revil-ransomware-arrest-kaseya-crypto-europol-cybersecurity

[26] PCMag. Definition of proxy server. Page consultée le 11 mai 2025. [Online]. Available : https://www.pcmag.com/encyclopedia/term/proxy-server

[27] Netskope. Qu'est-ce qu'un serveur proxy ? types et cas d'utilisation. [Online]. Available : https://www.netskope.com/fr/security-defined/what-is-a-proxy-server

[28] Squid-cache.org. Squid : Optimising web delivery. [Online]. Available : https://www.squid-cache.org/

[29] D. Wessels, *Squid : The Definitive Guide*. O'Reilly Media, 2004. [Online]. Available : https://books.google.dz/books?id=3xHKOYLKwlsC&pg=PR7

[30] CompTIA. What is wireshark and how to use it. [Online]. Available : https://www.comptia.org/content/articles/what-is-wireshark-and-how-to-use-it

[31] GeeksforGeeks. Tcp 3-way handshake process. [Online]. Available : https://www.geeksforgeeks.org/tcp-3-way-handshake-process/

[32] C. Goedegebure. Tcp 3-way handshake and port scanning. [Online]. Available : https://www.coengoedegebure.com/tcp-3-way-handshake-port-scanning/

[33] J. Postel, "Rfc 791 : Internet protocol," https://datatracker.ietf.org/doc/html/rfc791#section-3.2, 1981.

[34] J. Touch, "Rfc 9293 : Transmission control protocol (tcp)," https://www.rfc-editor.org/rfc/rfc9293.html, 2022.

[35] StormIT. What is rtt (round-trip time) and how to reduce it ? [Online]. Available : https://www.stormit.cloud/blog/what-is-round-trip-time-rtt-meaning-calculation/

[36] Keyfactor. What is tls handshake and how does it work ? [Online]. Available : https://www.keyfactor.com/blog/what-is-tls-handshake-how-does-it-work/

[37] Cloudflare. Négociation tls | le transport layer security handshake. [Online]. Available : https://www.cloudflare.com/fr-fr/learning/ssl/what-happens-in-a-tls-handshake/

[38] The SSL Store. The tls handshake : Taking a closer look. [Online]. Available : https://www.thesslstore.com/blog/explaining-ssl-handshake/#the-tls-13-handshake-step-by-step

[39] Xargs. The illustrated tls 1.3 connection : Every byte explained. [Online]. Available : https://tls13.xargs.org/#client-hello

[40] Agence nationale de la sécurité des systèmes d'information (ANSSI). (2017) Recommandations de sécurité relatives à tls. [Online]. Available : https://cyber.gouv.fr/publications/recommandations-de-securite-relatives-tls

[41] Cloudflare. User datagram protocol (udp). [Online]. Available : https://www.cloudflare.com/fr-fr/learning/ddos/glossary/user-datagram-protocol-udp/

[42] K. Scarfone and P. Mell. (2007) Guide to intrusion detection and prevention systems (idps). Disponible à l'adresse. [Online]. Available : https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf

[43] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, disponible à l'adresse. [Online]. Available : https://www.researchgate.net/publication/2597023_Intrusion_Detection_Systems_A_Survey_and_Taxonomy

[44] H. A. Jalil, A. Jaber, I. T. Almomani *et al.*, "Intrusion detection systems : A cross-domain overview," *Security and Privacy*, vol. 2022, 2022, disponible à l'adresse. [Online]. Available : https://onlinelibrary.wiley.com/doi/full/10.1155/2022/9663052

[45] E. Albin and N. C. Rowe. A realistic experimental comparison of the suricata and snort intrusion-detection systems. Disponible à l'adresse. [Online]. Available : https://faculty.nps.edu/ncrowe/suricataeval_fina12.htm

[46] S. A. R. Shah and B. Issac. Performance comparison of intrusion detection systems and application of machine learning to snort system. Disponible à l'adresse. [Online]. Available : https://www.researchgate.net/publication/320413444_Performance_Comparison_of_Intrusion_Detection_Systems_and_Application_of_Machine_Learning_to_Snort_System

[47] R. Fekolkin. Intrusion detection and prevention systems overview of snort and suricata. Disponible à l'adresse. [Online]. Available : https://www.researchgate.net/profile/Roman-Fekolkin-2/publication/297171228_Intrusion_Detection_and_Prevention_Systems_Overview_of_Snort_and_Suricata

[48] J. Camisso. Understanding suricata signatures. Disponible à l'adresse. [Online]. Available : https://www.digitalocean.com/community/tutorials/understanding-suricata-signatures

[49] Splunk siem : Comprendre la gestion des événements de sécurité. Disponible à l'adresse. [Online]. Available : https://docs.splunk.com/Documentation