الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

En Télécommunication

Spécialité : Réseaux & Télécommunications

Présenté par

AMALOU Hayet

&

NAIMI Yasmine

Identification de l'utilisation des réseaux P2P «BitTorrent Vs IPFS»

Proposé par : Mr. Mehdi Merouane & Mlle. Benachour Lina

Année Universitaire 2022-2023

Remerciements

En préambule à ce mémoire nous remercions ALLAH qui nous a aidé et nous a donné la patience et le courage durant ces longues années d'étude.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Nous remercions sincèrement Mr MEHDI Merouane qui a accepté de diriger ce travail de Master, ainsi que pour sa disponibilité et pour ses précieux conseils et précieuses orientations.

Nous remercions madame Benachour Lina pour ces conseils judicieux ses recommandation, son aide pratique et pour la motivation qu'elle nous a apportée.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils porteront à notre travail en acceptant d'examiner ce mémoire et de l'enrichir par leurs propositions.

A nos parents de nous avoir encouragés, supportés, épaulés et avoir cru en nous tout au long de ces années. Sans eux, nous ne serons pas là.

A nos frangins, frangines et amis de la promo de 2023. Nous remercions enfin tous ceux qui, d'une manière ou d'une autre, ont contribué à la réussite de ce travail et qui n'ont pas pu être cités ici.

Merci à tous.

ملخص:

يعد البي تورنت وابي يافاس أشهر شبكات اند لند المستخدمة على نطاق واسع لتخزين الملفات ومشاركتها. ومع ذلك، فإن لهذه الشعبية أيضًا عواقبها، لا سيما فيما يتعلق بالثغرات الأمنية وتشبع النطاق الترددي. يهدف مشروعنا إلى اكتشاف ومنع استخدام شبكة نظير إلى نظير داخل شبكة شركة. لهذا الغرض، تم إجراء تحليل متعمق لحزم مرور الشبكة باستخدام برنامج تحليل الحزم وايرشارك لاستخراج البصمات الرقمية الخاصة بـ البي تورنت وابي يافاس يتم بعد ذلك تنفيذ البصمات الرقمية هذه من خلال قواعد مخصصة في نظام اكتشاف ومنع التسلل سوري كاتا. النتائج التي تم الحصول عليها خلال الاختبارات التي أجريت مشجعة، مما يؤكد فعالية نهجنا

كلمات المفاتيح : شبكة الند لند ، بي تورنت ، ايبي يافاس ، وايرشارك ، البصمات الرقمية ،سوري كاتا

Résumé :

BitTorrent et IPFS sont les réseaux Peer to Peer les plus populaires, largement utilisés pour le stockage et le partage des fichiers. Cependant, cette popularité a également ses conséquences, notamment en termes de vulnérabilités en matière de sécurité et de saturation de la bande passante. Notre projet vise à détecter et à bloquer l'utilisation du réseau Peer to Peer au sein d'un réseau d'entreprise. Pour cela, une analyse approfondie des paquets de trafic réseau a été réalisée à l'aide du logiciel d'analyse de paquets Wireshark afin d'extraire les empreintes numériques spécifiques à BitTorrent et IPFS. Ces empreintes sont ensuite implémentées à travers des règles personnalisées dans un système de détection et de prévention d'intrusion IDS/IPS Suricata. Les résultats obtenus lors des tests effectués sont encourageants, confirmant ainsi l'efficacité de notre approche.

Mots clés : Réseau pair à pair; BitTorrent; IPFS; Wireshark; empreintes numériques; IDS; IPS; suricata.

Abstract :

BitTorrent and IPFS are the most popular peer-to-peer networks widely used for storing and sharing files. However, this popularity also has its consequences, especially in terms of security vulnerabilities and bandwidth saturation. Our project aims to detect and block the use of the peer-to-peer network within a corporate network. For this, an in-depth analysis of the network traffic packets was carried out using the packet analysis software Wireshark in order to extract the digital prints specific to BitTorrent and IPFS. These digital prints are then implemented through personalized rules in an IDS/IPS Suricata intrusion detection and prevention system. The results obtained during the tests carried out are encouraging, thus confirming the effectiveness of our approach.

Keywords : Peer-to-Peer Network; BitTorrent; IPFS; Wireshark; digital prints; IDS; IPS; Suricata.

Listes des acronymes et abréviations

BT-DHT : BitTorrent Distributed Hash Table.
CID : Content Identifier.
DAG : Directed Acyclic Graph.
DDOS : Distributed Denial of Service.
DHT : Distributed Hash Table.
DNS : Domain Name System.
DOS : Denial of Service.
HIDS : Host Intrusion Detection Systems.
HTTP : Hypertext Transfer Protocol.
ICMP : Internet Control Message Protocol.
ID : Identifier.
IDS : Intrusion Detection System.
IP : Internet Protocol.
IPFS : Interplanetary File System.
IPLD : Interplanetary Linked Data.
IPNS : interplanetary Name System.
MDNS : Multicast Domain Name System.
MSG : Message.
NIDS : Network intrusion detection systems.
P2P : Peer to Peer.
***Rev* :** revision.
RST : Reset.
SEIM : Security Information & Event Management.
Seti@Home : Search for Extra Terrestrial Intelligence.
SHA-256 : Secure Hash Algorithm.
SID : Signature Identifier.
TCP : Transmission Control Protocol.
TTL : Time To Live.
UDP : User Datagram Protocol.
URL : Uniform Resource Locator.
VOIP : Voice Over Internet Protocol.
VoP2P *:* Voice over Peer-to-Peer.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

L'avènement des réseaux Pair-à-Pair (P2P) a eu un impact majeur sur le développement d'Internet ces dernières années. Il a révolutionné la façon dont les utilisateurs interagissent et partagent des informations en proposant une approche décentralisée. Cette technologie favorise la collaboration, le partage des connaissances et l'accès à une variété de ressources à l'échelle mondiale. Les flux P2P sont de plus en plus populaires et représentent environ 80 % des flux de communication numérique.

Ces réseaux ont connu un immense succès dans le partage mondial de fichiers grâce à leur remarquable capacité à s'adapter à des environnements avec un grand nombre d'utilisateurs. Par exemple, ils permettent de distribuer rapidement et efficacement des fichiers d'une taille de 1 Go entre 100 000 utilisateurs en une seule journée. Cette statistique met en évidence les performances exceptionnelles de ces réseaux et souligne leur potentiel.

BitTorrent et IPFS sont parmi les protocoles Peer-to-Peer (P2P) les plus utilisés pour le partage et le stockage de fichiers. Ils sont accessibles à tous et offrent aux utilisateurs une méthode conviviale et simple pour le partage de fichiers.

À côté de ces évolutions et progrès incontestables, l'utilisation des réseaux Peer-to-Peer (P2P) peut engendrer certains problèmes, notamment une utilisation intensive de la bande passante et une facilitation du partage de logiciels piratés. Ces facteurs peuvent exposer les utilisateurs à des risques potentiels tels que les logiciels malveillants et le vol de données. De plus, le P2P est souvent associé à des activités illégales telles que le téléchargement de fichiers protégés par des droits d'auteur.

Un exemple de ce type d’attaque est celui qui s’est produit le 1er juin 2022, dans lequel un client de Google Cloud Armor a été la cible d’une série d’attaques DDoS HTTPS, qui ont culminé à 46 millions de requêtes par seconde. Il s’agit de la plus grande DDoS

signalée à ce jour. Environ 22 % (1169) des adresses IP sources correspondaient à des nœuds de sortie BitTorrent [1].

L'utilisation de ce dernier dans un réseau d'entreprise peut l'exposer à divers risques de sécurité et à des problèmes judiciaires. Donc il est recommandé aux entreprises et aux administrateurs de détecter voire bloquer la communication qui pourrait être établie vers les nœuds P2P.

À cet effet, on a examiné le phénomène du Peer to Peer d'un point de vue technique, en se concentrant sur les protocoles BitTorrent et IPFS, ainsi que leurs principes de fonctionnement et les différents risques qu'ils peuvent causer. Par la suite, On a mis en œuvre une solution qui consiste à analyser le trafic d'un réseau local pour détecter et bloquer l'utilisation du réseau Peer-to-Peer au sein de ce dernier en passant par les étapes suivantes :

1- Analyse profonde du trafic de réseau à l'aide de l'analyseur de paquets "Wireshark".
2- Extraction des empreintes P2P (BitTorrent et IPFS).
3- Implémentation des empreintes à travers des règles personnalisées dans le système IDS/IPS Suricata.

Pour mener à bien notre travail, on a structuré notre mémoire en quatre chapitres répartis comme suit :

- Dans le premier chapitre, on présente de manière générale les réseaux Peer-to-Peer (P2P).
- Dans le deuxième chapitre, on expose en détail le réseau IPFS et le protocole BitTorrent.
- Dans le troisième chapitre, on effectue une analyse minutieuse à l'aide de Wireshark afin d'extraire les empreintes numériques spécifiques à BitTorrent et IPFS.
- Enfin, dans le dernier chapitre, on détecte et bloque l'utilisation de BitTorrent et IPFS en mettant en œuvre ces empreintes dans le système de détection et de prévention d'intrusion Suricata.

Chapitre 1

Les Réseaux Peer to Peer

Chapitre 1 Les réseaux Peer to Peer

1.1 Introduction

Au cours des dernières années, les réseaux pairs à pairs (P2P) ont connu une popularité croissante grâce à leurs caractéristiques avantageuses pour les utilisateurs. Actuellement, l'industrie et la recherche considèrent ce modèle comme une alternative crédible au modèle client-serveur traditionnel et travaillent activement dans ce domaine.

Dans ce chapitre, on va mettre l'accent sur les généralités des réseaux Peer to Peer en abordant leurs architectures, leur fonctionnement, ainsi que leurs caractéristiques et applications.

1.2 Réseau client/serveur

Un réseau client/serveur spécifie un mode de communication entre plusieurs ordinateurs sur un réseau. Chaque entité est considérée comme un client ou un serveur. Chaque logiciel client peut envoyer des requêtes au serveur. Un serveur pouvant être dédié aux applications, aux fichiers, aux terminaux ou à la messagerie électronique.

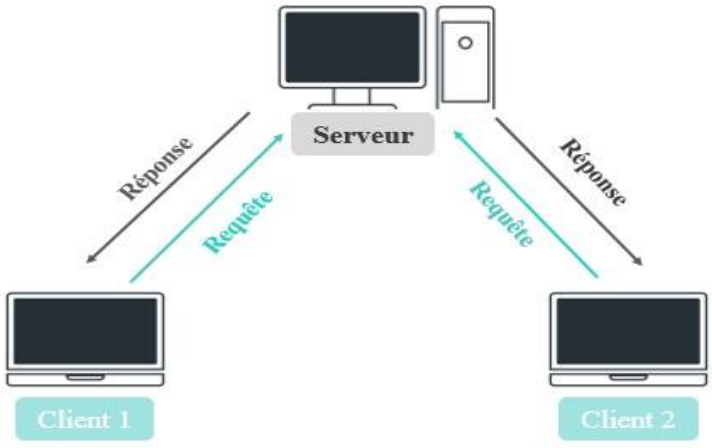


Figure 1.1 : Réseau client/serveur.

- **Le client**

  Le client est caractérisé comme suit :

  - Il envoie des requêtes au serveur.
  - Il attend et reçoit les réponses du serveur.

- **Le serveur**

  Le serveur est caractérisé comme suit :

  - Il attend, il est à l'écoute.
  - Prêt à répondre aux demandes envoyées par des clients. Une fois que la requête lui parvient, il la traite et envoie une réponse.

1.2.1 Les avantages du réseau client/serveur

- Une meilleure sécurité : Toutes les données sont centralisées sur un seul serveur.
- Une administration au niveau serveur : vu que les clients ont moins d'importance dans ce modèle, leur administration n'est pas une nécessité.

1.2.2 Les inconvénients du réseau client/serveur

- Un coût élevé dû à la complexité de la technologie du serveur.
- Le fonctionnement de chaque client sera perturbé si le serveur tombe en panne.

1.3 Réseau Peer to Peer

Le terme « Peer-to-Peer », « poste à poste » ou « pair à pair » en français (ou plus couramment P2P), est un mode de communication et de partage de ressource informatique qui ne nécessite pas un serveur centralisé. Dans un réseau P2P, chaque nœud sur le réseau joue à la fois le rôle de client et de serveur, ce qui permet aux utilisateurs d'échanger différents types d'information directement entre eux.

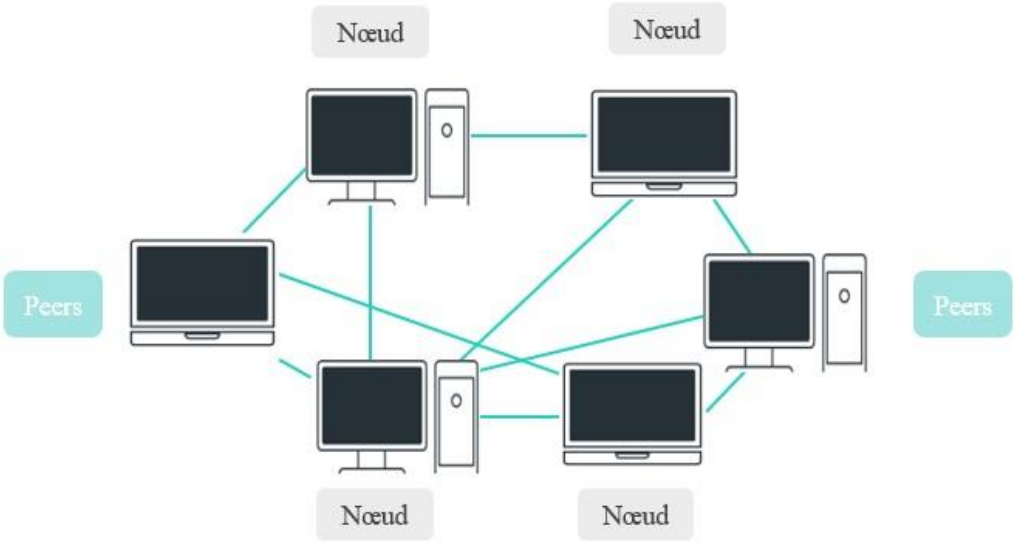


Figure 1.2 : Réseau Peer to Peer.

1.4 Comparaison entre Client/serveur et P2P

caractéristique	Peer to Peer	Client/serveur
Rôle de nœud	Chaque nœud joue le rôle de client et de serveur	Rôles diffèrent entre les serveurs et les clients
Décentralisation	Décentralisé, chaque nœud peut communiquer directement avec les autres nœuds	Centralisé, les clients se connectent aux serveurs pour accéder aux ressources
Sécurité	Moins sécurisé, risques de confidentialité	Plus sécurisé, l'accès aux données est contrôlé par des serveurs centralisés
Utilisation	Partage de fichiers, échange de données	Services en ligne, applications d'entreprise
Coût	Les réseaux Peer-to-Peer sont moins coûteux à mettre en œuvre	L'architecture client-serveur est coûteuse à mettre en œuvre

Tableau 1.1 : Client/serveur vs P2P**.**

1.5 L'historique de réseau Peer to Peer

Dans les années 1980, les réseaux P2P ont été utilisés pour la première fois après l'introduction des ordinateurs personnels. En août 1988, l'Internet Relay Chat a été le premier réseau P2P construit pour partager du texte et discuter [2].

En juin 1999, Napster, un logiciel P2P de partage de fichiers, a été mis au point. Il pouvait également être utilisé pour partager des fichiers audio. Ce logiciel a été fermé en raison du partage illégal de fichiers. Mais le concept de partage en réseau, c'est-à-dire le P2P, est devenu populaire [2].

En juin 2000, Gnutella a été le premier réseau décentralisé de partage de fichiers P2P. Il permettait aux utilisateurs d'accéder aux fichiers se trouvant sur les ordinateurs d'autres utilisateurs par l'intermédiaire d'un dossier désigné [2].

1.6 Les caractéristiques du Peer-to-Peer

La notion paire à paire est devenue de plus en plus populaire ces dernières années en raison de ses caractéristiques uniques qui la distinguent des autres architectures de réseau traditionnelles (telles que les réseaux client-serveur). Parmi les principales caractéristiques du modèle Peer-to-Peer, on cite :

Figure 1.3 : Les caractéristiques Peer to Peer***.***

- **Distribution des données**

Les données mises à disposition par les utilisateurs ne sont pas centralisées dans un serveur : elles sont réparties sur l'ensemble des pairs du système [3].

- **Tolérance aux pannes**

Le fait qu'un pair soit défaillant ne remet pas en cause le fonctionnement global du système [3].

- **Passage à l'échelle**

Le nombre de pairs dans le système et la quantité d'information qu'ils partagent n'empêchent pas le système de fonctionner [3].

- **Dynamisme**

Les pairs sont libres de rejoindre ou de quitter le système à tout moment [3].

- **Autonomie**

Les pairs sont libres dans le choix des ressources qu'ils souhaitent partager et de la manière dont ils veulent les partager [3].

1.7 Application des réseaux P2P

Les réseaux Peer to Peer sont utilisés dans diverses applications et domaines. On trouve donc :

- **Partage de fichiers**

Les réseaux P2P sont souvent utilisés pour le partage de fichiers entre les utilisateurs. Ils peuvent télécharger et partager des fichiers tels que de la musique, des vidéos, des documents, etc. utiliser principalement des logiciels gratuits pour la plupart comme BitTorrent, Napster, eDonkey, IPFS...

- **La voix sur P2P (VoP2P)**

Skype [SKYPE] est un client VoP2P lancé en 2003 qui a atteint plus de 10 millions d'utilisateurs simultanés. Par rapport aux clients VOIP antérieurs, Skype propose à la fois des appels gratuits de bureau à bureau et des appels à faible coût de bureau à réseau téléphonique public commuté (RTPC), y compris des appels internationaux.

Contrairement aux systèmes de partage de fichiers, Skype promet une politique sans logiciel espion [4].

- **Le calcul distribué**

Le calcul distribué sur les réseaux Peer-to-Peer (P2P) est une méthode de traitement de données, qui implique la distribution de calculs complexes entre différents nœuds du réseau. Contrairement aux méthodes de calcul centralisées, où une seule machine effectue tous les calculs, il est utilisé dans de nombreux domaines, notamment pour le traitement de données scientifiques, les simulations et l'apprentissage automatique. Comme exemple, le projet SETI@home (Search for Extra Terrestrial Intelligence).

- **Les moteurs de recherche**

Les moteurs de recherche P2P sont des outils qui permettent de rechercher ou indexer des fichiers sur les nœuds du réseau sans avoir à passer par un serveur centralisé. On peut citer « InfraSearch » comme moteur de recherche P2P, le réseau « Gnutella » et « eDonkey ».

- **Les Plateformes de Développement**

Les plateformes de développement P2P sont conçues pour simplifier et accélérer le développement d'applications P2P en fournissant des outils prêts à l'emploi pour la gestion des ressources, la communication entre pairs, le partage de fichiers et la sécurité des données. Telle que la plateforme « Ethereum » développée par Vitalik Buterin en 2015.

- **Les bases de données distribuées**

Les bases de données distribuées basées sur des réseaux Peer-to-Peer comme « Cassandra » sont conçues pour fournir une grande évolutivité, une disponibilité et une tolérance aux pannes. Elles permettent de gérer de grandes quantités de données structurées réparties sur de nombreux serveurs, appelés nœuds.

1.8 Les architectures des réseaux Peer to Peer

Les architectures des réseaux pair-à-pair sont divisées en deux grandes classes : les réseaux non structurés et les réseaux structurés. Cependant, dans les modèles non structurés, on distingue trois architectures : une centralisée, une décentralisée (pur) et l'autre hybride entre le modèle centralisé et pur.

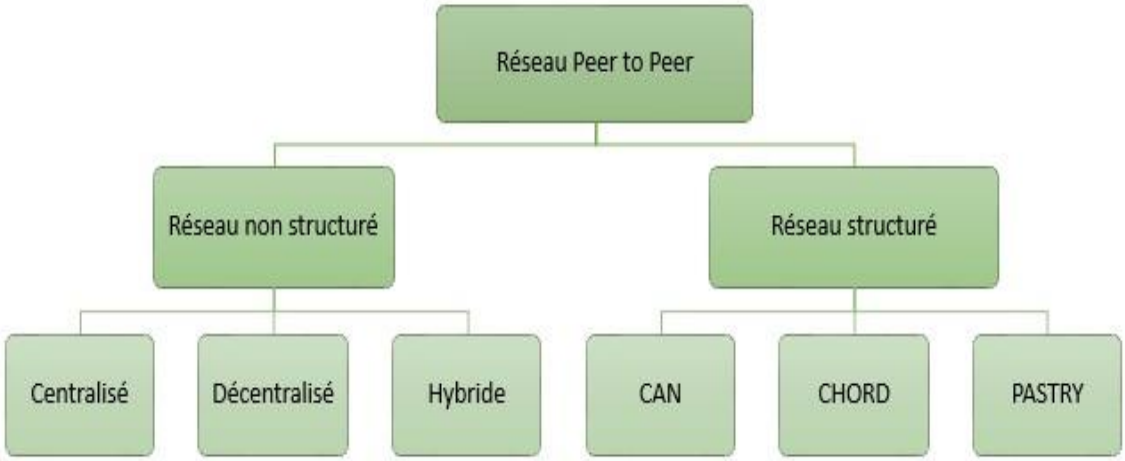


Figure 1.4 : Les architectures des réseaux Peer to Peer.

1.8.1 Réseau Peer to Peer non-structuré

Les réseaux P2P non structurés sont des réseaux où il n'y a ni répertoire centralisé ni contrôle précis sur la topologie du réseau et sur l'emplacement des fichiers. Ces systèmes sont les plus simples et les plus anciens, où la recherche se fait de proche en proche à travers le réseau. Ils se divisent en trois architectures : (centralisée, décentralisée et l'autre hybride) [5].

a Architecture centralisée

Cette architecture est caractérisée par un serveur central qui assure la gestion des données et des requêtes des clients. Ce serveur est chargé de l'indexation des fichiers et de l'attribution des adresses aux utilisateurs qui possèdent les fichiers recherchés. Les échanges de données entre les clients se font directement, sans passer par le serveur central. C'est ce qui distingue principalement cette architecture d'une architecture traditionnelle de type client-serveur.

L'exemple de logiciel le plus courant reposant sur cette architecture est Napster.

❖ **Principe de Fonctionnement**

➢ Un utilisateur recherche un fichier ressource en envoyant une requête au serveur central [6].

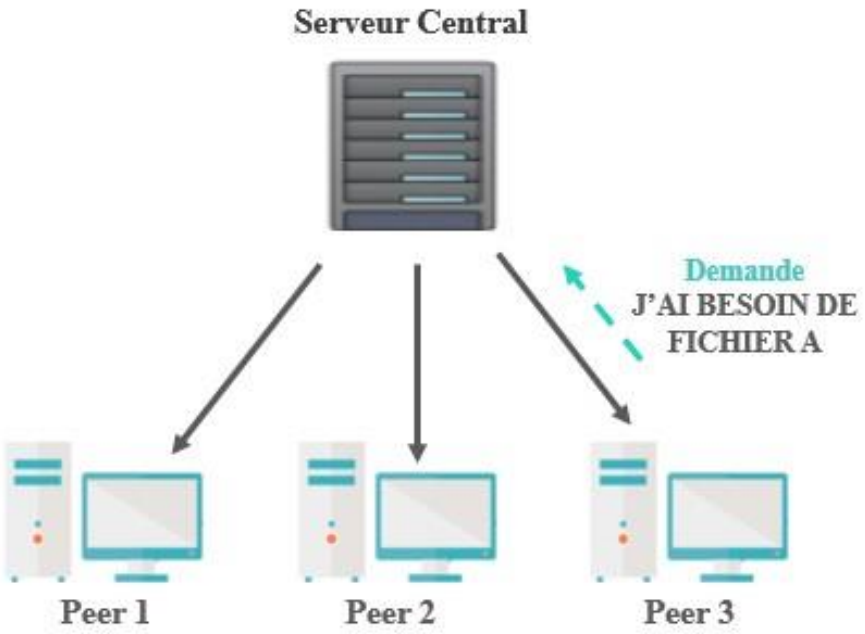


Figure 1.5 : Rechercher un fichier"architecture centralisée".

➢ Le serveur central répond et transmet la liste (adresse IP, nom d'utilisateur, taille du fichier) des ordinateurs utilisateurs proposant le fichier demandé [6].

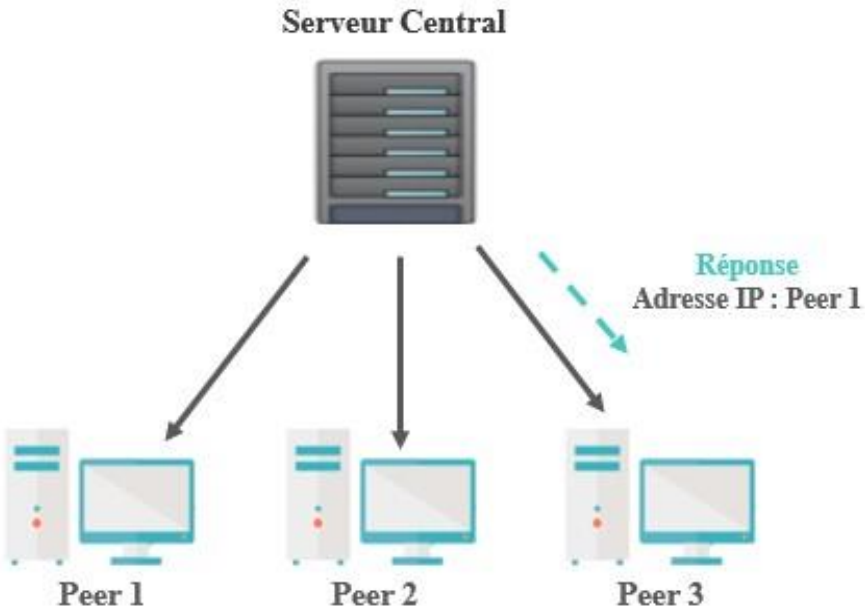


Figure 1.6 : Réponse du serveur"architecture centralisée".

- L'utilisateur télécharge le fichier directement à partir d'un des ordinateurs renseignés par le serveur. Le serveur n'est plus impliqué dans le transfert de fichier [6].

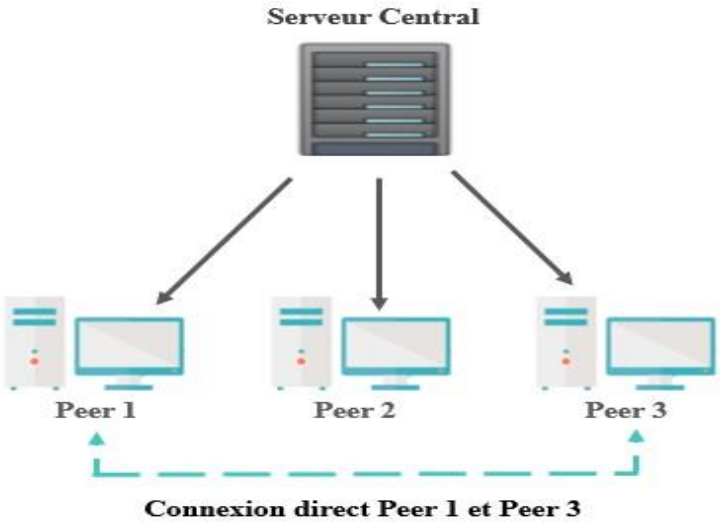


Figure 1.7 : Téléchargement du fichier"architecture centralisée***".***

Pour optimiser les performances de ce système, il est envisageable de créer un anneau (groupe) de serveurs, cela permet d'améliorer la disponibilité du système, vu que l'accès aux données partagées dans chaque serveur est transparent pour les autres clients.

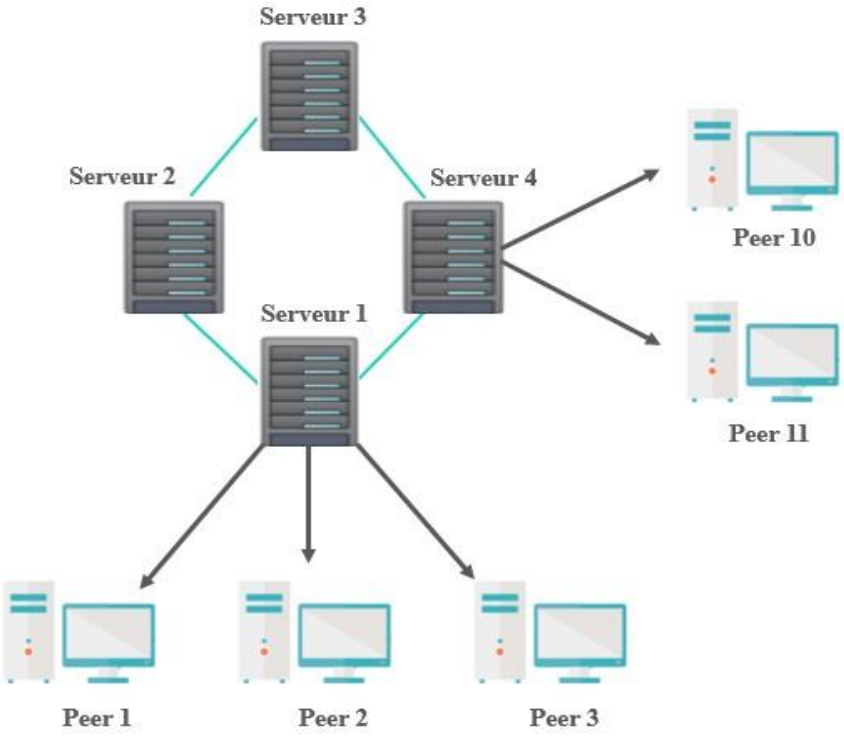


Figure 1.8 : Technique d'amélioration de l'architecture centralisée***.***

En revanche, le réseau centralisé présente une grande vulnérabilité, car la panne du serveur peut entraîner l'arrêt total du réseau. De plus, l'utilisation d'une architecture centralisée, nécessitant un enregistrement, ne garantit pas l'anonymat des utilisateurs.

b Architecture décentralisée (pur)

Dans cette architecture, tous les nœuds sont égaux et jouent le même rôle et communiquent entre eux, sans passer par un serveur, chaque élément du réseau s'appelant **servent (Servent = Serveur + Client)** et indexe lui-même ses propres fichiers. Ceux qui sont à la recherche d'un fichier interrogent, de proche en proche, tous les ordinateurs du réseau. L'exemple le plus connu est le réseau Gnutella [7].

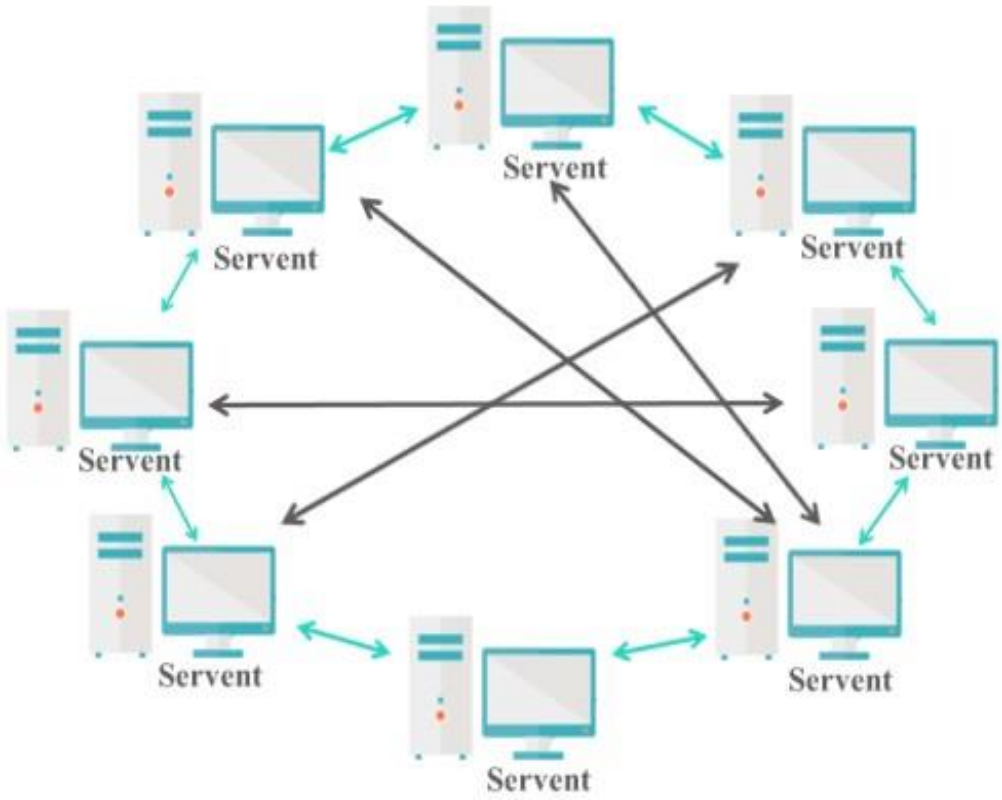


Figure 1.9 : Architecture décentralisée(pur)**.**

❖ **Principe de fonctionnement**

Pour comprendre comment fonctionne cette architecture, on va prendre l'exemple du réseau « Gnutella ». D'où le client a pour but de :

- **Joindre le réseau Gnutella :** le client commence par envoyer un message à ses voisins disponibles (le Ping), qui l'envoient à leur tour à leur voisin et ainsi de suite. Les nœuds qui ont reçu le message de connexion répondent en fournissant leur adresse IP, le numéro de port ainsi que la liste et la taille des données qu'ils partagent (le Pong) [8].

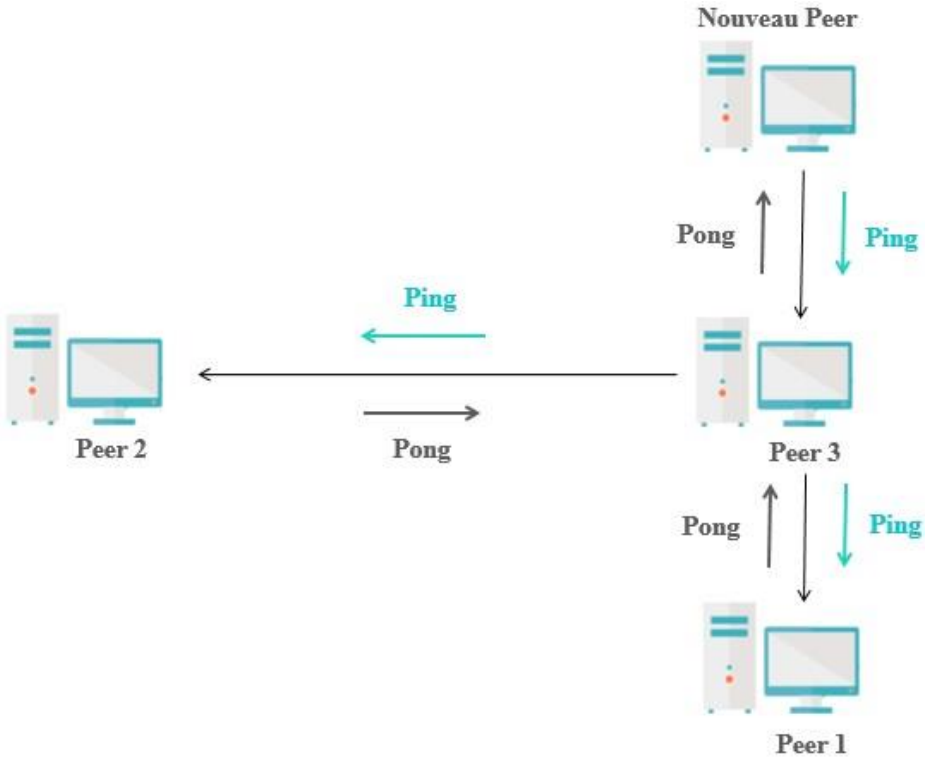


Figure 1.10 : Connexion d'un Peer "architecture décentralisée"**.**

- **Lancer une recherche :** le client envoie une requête (Query) à tous ses voisins qui vont eux même la transmettre à leurs voisins et ainsi de suite jusqu'à TTL=0. Tous les nœuds qui sont concernés par la requête vont répondre à l'émetteur du signal (Queryhit). La réponse remonte de proche en proche jusqu'au client initial qui va par la suite envoyer une requête (Get) de téléchargement directement au client qui possède le fichier. Si les données sont derrière un firewall, un message Push est utilisé [8].

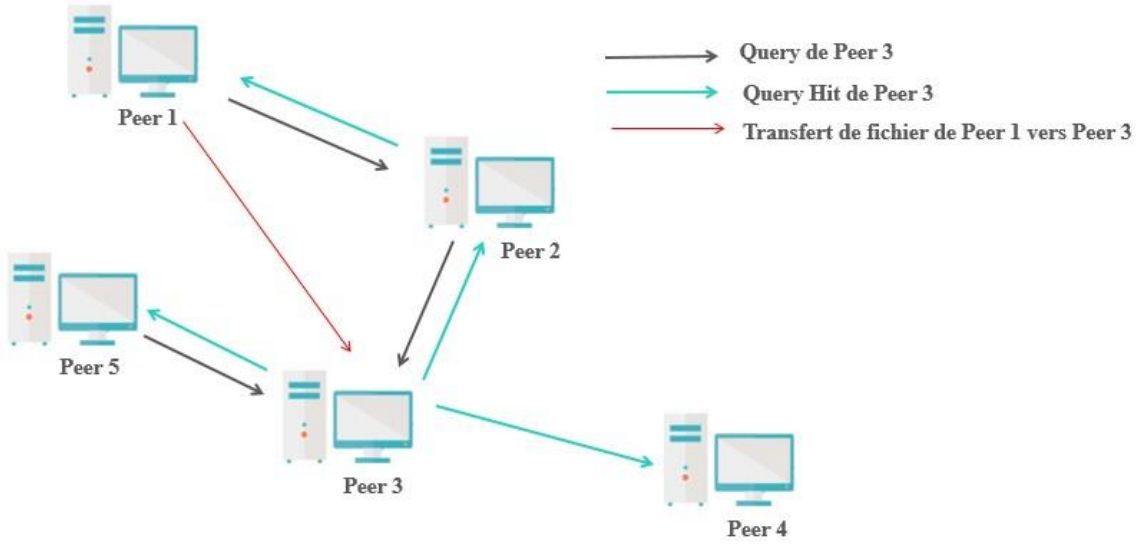


Figure 1.11 : Téléchargement d'un fichier "architecture décentralisée".

Le partage de fichiers partiellement téléchargés n'est possible qu'avec certains clients du réseau Gnutella, par exemple (Shareazza, Limewire…).

L'architecture décentralisée présente des avantages significatifs tels que la tolérance aux pannes et la garantie de l'anonymat des nœuds. Cependant, elle est également caractérisée par un manque de sécurité, ce qui la rend vulnérable aux attaques externes. De plus, elle est souvent très consommatrice de bande passante, ce qui peut engendrer des problèmes de performance et de coûts.

c Architecture Semi-décentralisée (hybride)

L'architecture hybride est une combinaison entre le modèle centralisé et le modèle décentralisé, autrement dit ce dernier est organisé d'une manière hiérarchique. On distingue deux types de nœuds :

- ✓ Les nœuds standards (Peers) : sont caractérisés par une faible bande passante.
- ✓ Les supers nœuds "supers-Peers" : sont caractérisés par une forte capacité du calcul et une bande passante élevée.

La liaison entre les Peers et les super Peers se fait de manière centralisée, chaque super Peer est connecté aux éléments de son groupe, en réalisant l'indexation de leurs ressources. L'interconnexion entre les supers nœuds est située au haut niveau de la hiérarchie, selon le modèle décentralisé. Ce type d'architecture est notamment utilisé dans les réseaux (FastTrack, eDonkey, BitTorrent).

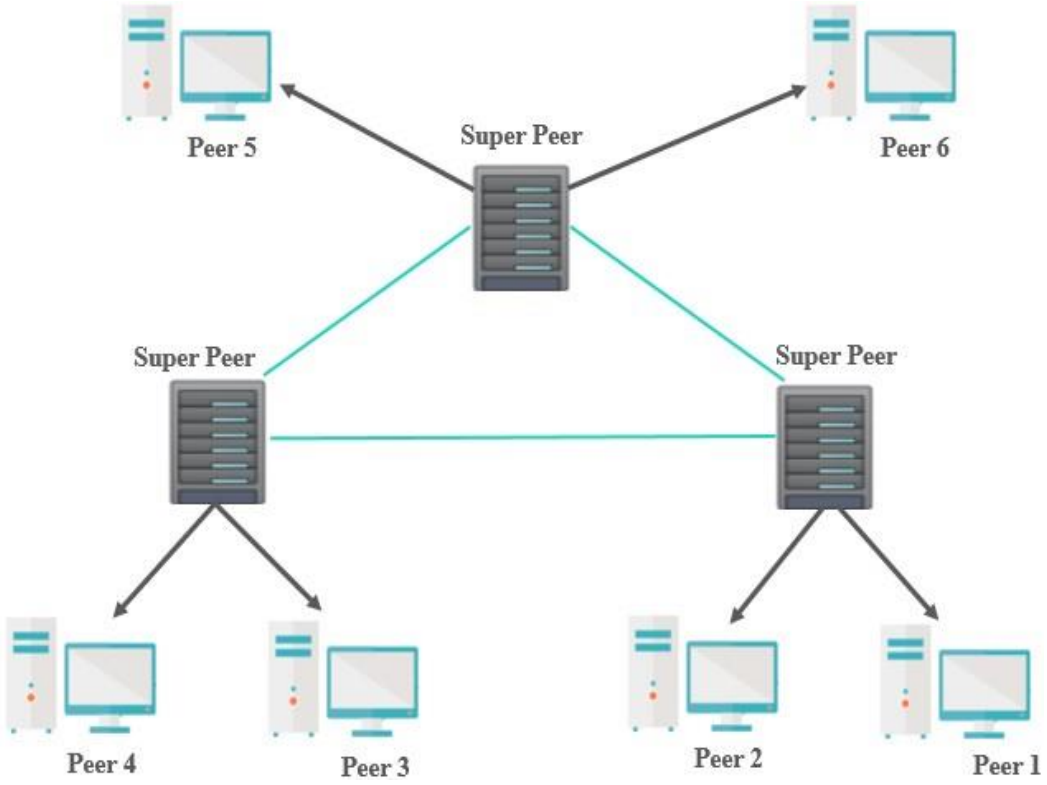


Figure 1.12 : Architecture hybride.

- **Principe de fonctionnement**

Pour accéder au réseau P2P, un nœud doit se connecter à un « super pair » auprès duquel il déclare la liste des ressources qu'il veut partager. Une fois connecté, un pair peut accéder à toutes les ressources du système en envoyant, tout simplement, ses requêtes à son « super pair ».

Ce dernier se chargera de déterminer la localisation des ressources demandées, soit en accédant à ces informations locales (la liste des ressources hébergées par les postes qu'il gère), soit en diffusant la requête aux autres « super pair ».

L'avantage de ce modèle est qu'il ne se base pas sur un nœud central et donc il est plus robuste aux attaques de déni de service. En contrepartie, et que l'anonymat des utilisateurs ne peut plus être garanti.

Ce modèle est utilisé dans des applications telles que Kazaa.

1.8.2 Réseau Peer to Peer structuré

Les réseaux P2P structurés ont été proposés pour améliorer les performances des réseaux P2P non structurés car ils diminuent le temps de recherche et ils permettent de localiser les fichiers très rapidement. Ils utilisent des tables de hachage distribuées (DHT) pour gérer les opérations de recherche. Une entrée dans la table contient la paire (clé, valeur), où la clé est l'identifiant associé à un objet partagé (en hachant son nom) et la valeur correspond à l'information de localisation d'objet recherché (en hachant l'adresse IP).

Parmi les algorithmes structurés, on peut citer : Chord, CAN, Tapestry, Pastry, Kademlia.

1.9 Les avantages du P2P

Les architectures P2P présentent beaucoup d'avantages parmi eux :

- **Faible coût :** Exploite les ressources existantes.
- **Extensibilité :** Il est très facile de rajouter de nouveaux pairs. Ceux-ci sont gérés dynamiquement.
- **Les communications sont directes :** un nœud peut accéder directement à un ou plusieurs nœuds.
- **Disponibilité accrue des ressources :** plus le nombre d'utilisateurs augmente, plus la disponibilité des ressources présentes augmente.

1.10 Les inconvénients du P2P

Les réseaux pair-à-pair rencontrent de nombreux problèmes et présentent quelques inconvénients, on peut citer essentiellement :

- **Le freeloading :** les utilisateurs téléchargent des fichiers à partir d'autres pairs sans jamais contribuer en partageant leurs propres fichiers.
- **Topologie instable :** les pairs d'un réseau pair-à-pair étant dynamiques, ils peuvent apparaître et disparaître à tout moment, par conséquent les ressources aussi.

- **Le partage de fichiers illégaux :** l'anonymat offert par les réseaux P2P peut faciliter le partage de fichiers protégés par les droits d'auteur.
- **La propagation de virus et de logiciels malveillants :** les fichiers partagés peuvent contenir des virus et des logiciels malveillants, qui peuvent infecter les ordinateurs des utilisateurs.

1.11 Les risques d'utiliser les réseaux p2p au sein d'un réseau d'entreprise

Les réseaux p2p sont un outil pratique et utile pour partager et stocker des données sur Internet. Cependant, il est important de noter que leur utilisation au sein d'une entreprise peut comporter des risques et des menaces liées à la sécurité et à la confidentialité du réseau. Parmi ces risques on trouve :

1.11.1 Risque d'infecter le système de l'entreprise par des logiciels malveillants

Les P2P comme BitTorrent sont très connus comme étant une des sources permettant la propagation des logiciels malveillants tels que des virus, des chevaux de Troie, des ransomwares ou des logiciels espions (spyware), lorsque Les employés téléchargent des fichiers à partir de sources non fiables ou partagent des fichiers infectés sur leurs ordinateurs, peuvent introduire des virus qui se propageront très rapidement sur tous les postes de travail, risquant de paralyser le réseau de la société.

1.11.2 Risque d'exposer l'entreprise à différents types d'attaques

a Attaque DOS/DDOS

Le trafic réseau P2P peut entraîner une consommation élevée de la bande passante du réseau d'entreprise, ce qui peut causer des problèmes de surcharge du réseau et ralentir l'ensemble des connexions. De plus, ces réseaux peuvent être détournés à des fins malveillantes pour mener des attaques de déni de service (DoS) et de déni de service distribué (DDoS), le but de ces attaques est de rendre votre serveur, un service ou une infrastructure indisponible.

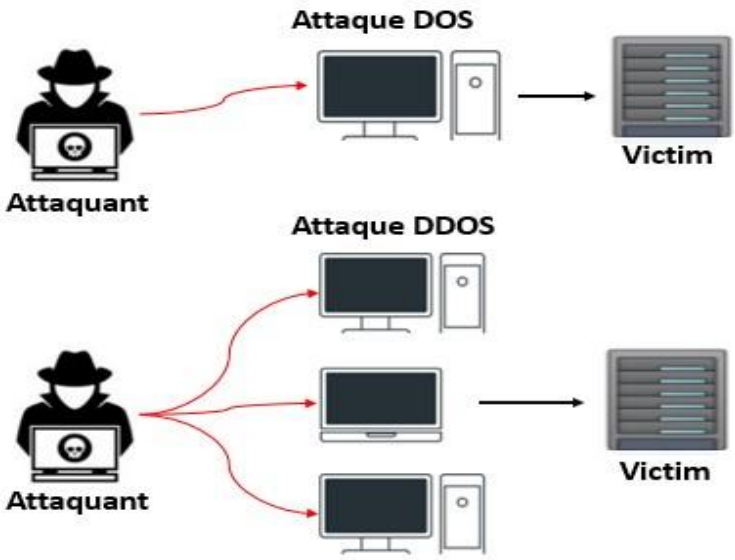


Figure 1. 13 : Attack Dos/DDOS***.***

b Attaque Sybil

L'attaque Sybil consiste à créer un grand nombre de pairs malveillants dans le réseau pour une même entité (des identités multiples) pour gagner le contrôle d'une partie du réseau. Une fois que cela a été accompli, l'attaquant peut brusquer le protocole de communication de n'importe quelle manière. Par exemple, il pourrait gagner la responsabilité de certains fichiers et choisir de les polluer [9].

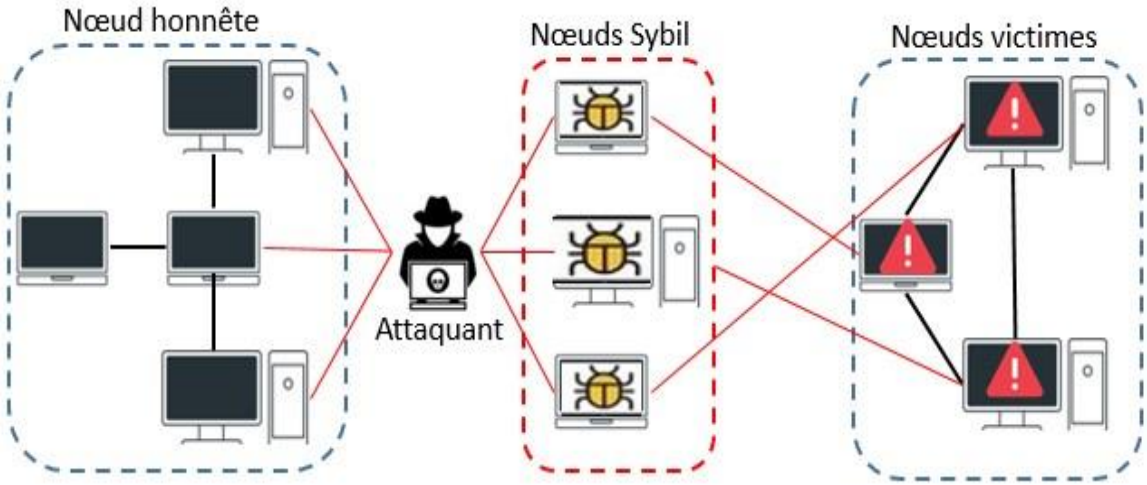


Figure 1.14 : Attack Sybil.

c Attaque Eclipse

L'attaque Eclipse est plus générale que l'attaque Sybil, elle consiste à contrôler une grande partie des voisins d'un bon pair (un pair sain).

Dans cette situation, l'ensemble des nœuds malicieux travaillent ensemble pour tromper le nœud sain en écrivant leurs adresses dans sa table et donc modifier l'utilisation normale du routage. L'attaquant cache alors les bons pairs sur le réseau, d'où le nom d'attaque Eclipse [9].

1.11.3 Risque de nuire à la réputation de l'organisation

Lorsque les employés utilisent des réseaux P2P tels que BitTorrent et IPFS pour stocker et partager du contenu illégal, comme des images pornographiques enfantines, des discours haineux ou du matériel protégé par des droits d'auteur, cela expose l'entreprise à une responsabilité potentielle. En conséquence, l'entreprise peut faire face à des poursuites judiciaires, des amendes et des dommages financiers importants.

1.11.4 Risque de confidentialité des données

Les réseaux P2P peuvent faciliter le partage non autorisé de fichiers, ce qui augmente le risque de fuites de données sensibles de l'entreprise. Si des fichiers contenant des informations confidentielles ou personnelles sont partagés via un réseau P2P, ils pourraient être accessibles à des parties non autorisées, compromettant ainsi la confidentialité des données.

1.12 Solution proposée

Compte tenu des risques mentionnés ci-dessus, il est recommandé aux entreprises et aux administrations de détecter et, si nécessaire, de bloquer les communications qui pourraient être établies vers des nœuds P2P.

Dans le cadre de notre travail, on propose une solution qui vise à repérer l'utilisation des réseaux P2P, plus spécifiquement les protocoles BitTorrent et IPFS qui seront décrits dans le prochain chapitre.

Pour ce faire, on va utiliser l'analyseur de paquets Wireshark afin d'extraire les signatures numériques caractéristiques du trafic P2P. Ensuite, on va implémenter ces signatures au niveau de l'IDS/IPS Suricata pour permettre la détection des utilisateurs P2P et l'interception de leurs activités.

1.13 Conclusion

Au cours de ce premier chapitre, on a présenté généralement les réseaux P2P. Tout d'abord, on a commencé par le réseau client-serveur, ces avantages et ces inconvénients, après, on a vu le réseau Peer to Peer, la comparaison entre le réseau client-serveur et le réseau P2P, l'historique du réseau P2P, ces caractéristiques, ces applications, les architectures, ces avantages et ces inconvénients.

On a aussi examiné les risques liés à l'utilisation des réseaux Peer-to-Peer au sein d'une entreprise et on a proposé une solution pour prévenir toute utilisation abusive de ces réseaux.

Le prochain chapitre sera consacré aux deux protocoles BitTorrent et IPFS.

Chapitre 2

BitTorrent vs IPFS

Chapitre 2 BitTorrent vs IPFS

2.1 Introduction

Les réseaux P2P représentent aujourd'hui une partie considérable des échanges sur Internet, principalement parce qu'ils offrent aux utilisateurs du monde entier un moyen rapide et efficace pour partager et stocker des ressources.

BitTorrent et IPFS sont les protocoles de partage et de stockage des données pair à pair les plus utilisés, dans ce chapitre on va étudier ces deux protocoles ainsi que leur fonctionnement et on finira par une comparaison entre eux.

2.2 Protocole BitTorrent

2.2.1 Définition

BitTorrent est un protocole de partage de fichiers pair à pair, inventé par le programmeur Bram Cohen en 2001. Il repose sur le principe du téléchargement multiple, qui permet de récupérer un fichier à partir de plusieurs sources simultanément. Pour cela, le fichier est divisé en petites parties appelées "blocs" qui sont téléchargées individuellement à partir de différents pairs [10].

La particularité de BitTorrent réside dans la coopération entre les utilisateurs. En effet, pour accélérer le téléchargement de nouvelles parties du fichier, il est nécessaire de partager celles que l'on a déjà téléchargées. Ainsi, plus il y a de pairs qui partagent le fichier, plus la vitesse de téléchargement est élevée. BitTorrent se concentre principalement sur le téléchargement des fichiers, tandis que l'indexation des fichiers est gérée par un autre protocole [10].

2.2.2 Les termes essentiels du protocole BitTorrent

- **Client "Peer" :** c'est l'utilisateur qui participe au partage de fichiers en installant un logiciel BitTorrent tel que, UTorrent, BitTorrent ou Deluge...
- **Seed :** c'est l'utilisateur qui possède une copie complète du fichier et prêt à le partager.
- **Leech :** c'est l'utilisateur qui télécharge les fichiers sans les partager.
- **Swarm :** l'ensemble des Peers (seeders et leechers) qui participent au partage du même torrent.
- **Tracker :** c'est un serveur permettant de mettre en relation les pairs. Il peut être considéré comme un annuaire mettant en relation un fichier avec les clients qui le téléchargent.
- **Scrape :** C'est quand un client envoie une requête au tracker pour obtenir des informations sur les statistiques du torrent.
- **Le fichier MetaData (Torrent) :** c'est un fichier avec l'extension ".torrent" qui permet d'identifier les partages des utilisateurs sur le réseau. Ces partages peuvent être des fichiers tels que des films, musique, images, ou des répertoires.
- **Bloc (Fragment) :** Un bloc est un morceau de fichier. Lorsqu'un fichier est distribué via BitTorrent, il est divisé en plus petits morceaux, ou blocs. En général, la taille du bloc est de 256 Ko. Mais elle peut varier en fonction de la taille du fichier distribué.

2.2.3 La recherche d'informations et l'indexation des torrents

Le protocole BitTorrent ne possède aucune fonction d'indexation, ce qui limite la possibilité de rechercher directement des fichiers à l'aide des clients BitTorrent. Pour trouver des fichiers, il est nécessaire d'utiliser des applications externes. Il existe de nombreux sites web qui offrent des liens vers des torrents, où l'utilisateur peut télécharger le fichier (.torrent). Ensuite, il suffit d'importer ce fichier dans le client BitTorrent pour qu'il se charge du téléchargement.

2.2.4 Principe de fonctionnement

Pour télécharger un fichier en utilisant le protocole BitTorrent, l'utilisateur doit d'abord installer au moins un logiciel BitTorrent tel que (qBitTorrent, utTorrent ou Vuze). Ensuite, il explore le Web pour trouver le lien du fichier MetaData (.torrent) [11].

L'URL inclue dans le fichier (.torrent) est utilisée pour contacter le tracker responsable du fichier afin d'obtenir la liste des pairs qui partagent actuellement le fichier [11].

Le tracker renvoie une liste d'adresses IP avec lesquelles le client va interagir directement pour commencer le téléchargement. Cette liste d'adresses IP est simplement constituée des personnes qui téléchargent le même fichier que nous au même temps. Notre client télécharge le fichier à partir des autres pairs, tandis que ceux-ci téléchargent également chez nous [11].

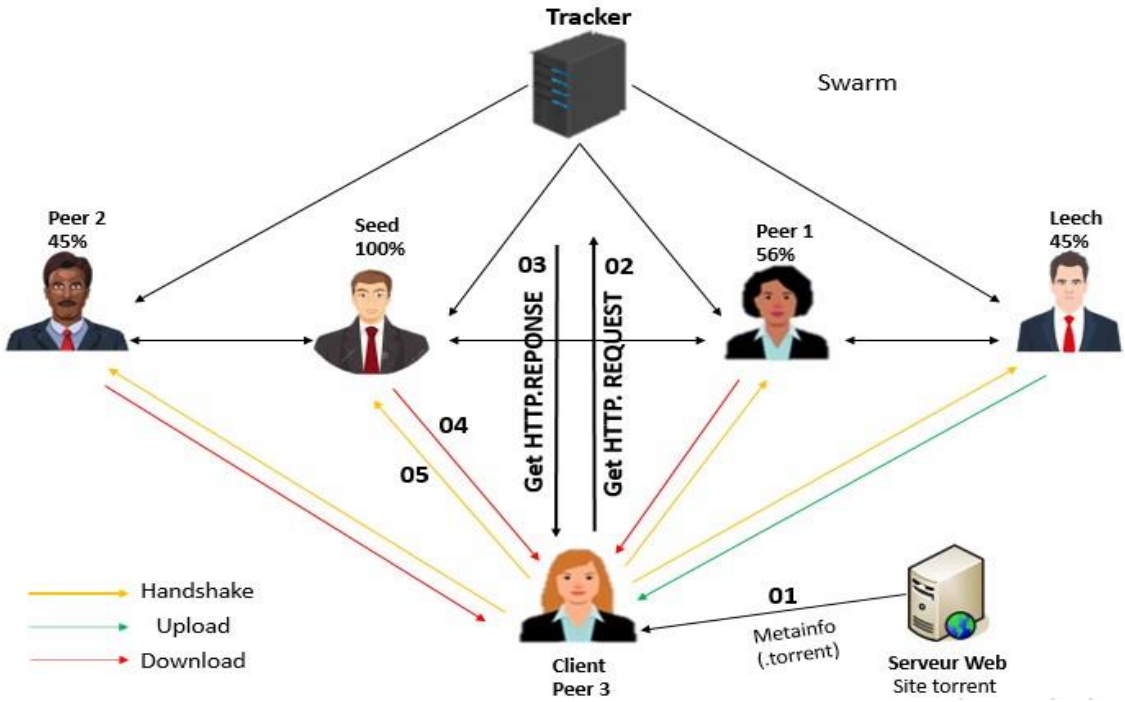


Figure 2.1 : Principe de fonctionnement "Protocole BitTorrent".

a Le fichier métadonnée

Le fichier (.torrent) est utilisé par les clients BitTorrent pour localiser les pairs (Peers) et les trackers, ce qui facilite le partage et le téléchargement des fichiers. Il contient différentes informations (voir table 2.1).

Contenue	Description
Announce	L'URL d'annonce du tracker
Info	Un dictionnaire qui décrit les fichiers
Name	Nom du fichier torrent
Lenght	la taille du fichier en octets
Path	chemin d'un fichier
Pièces	Chaîne de taille multiple de20 octect représentant le code sh1 de chaque partie
Pièce1 enght	La taille d'un block du fichier

Tableau 2.1 : Le contenue d'un fichier MetaData(.torrent)***.***

b Connexion au tracker

Une fois que le client a téléchargé le fichier de métadonnées (.torrent), il extrait l'adresse IP du tracker. Ensuite, il établit une connexion à celui-ci en utilisant le protocole HTTP.

- **La requête**

Tout d'abord, le pair envoie une requête "Get" au tracker. Cette requête permet de fournir au tracker différentes informations sur le client et sur le fichier recherché. Parmi ces paramètres, on trouve :

info_hash	l'empreinte de 20 octets qui correspond au hash SHA1 du fichier Meta Data, et qui permet de l'identifier de manière unique.
peer_id	identifiant unique du client , d'une taille de 20 octets
Port	le port sur lequel le client écoute, par défaut, BitTorrent réserve la plage 6881-6889
Adresse IP	Adresse IP du client
uploaded	le nombre total d'octet envoyés par le client
downloaded	le nombre total d'octet téléchargés par le client

Tableau 2.2 : Le contenue de la requête HTTP tracker.

- **La réponse**

En réponse, le tracker renvoie la liste des Peers partageant actuellement le fichier demandé (ou un message d'erreur le cas échéant). Cette requête contient :

Interval	le temps qu'il doit attendre le client pour contacter le tracker.
Complete	Le nombre de pairs possédant le fichier (seeders)
Incomplete	Le nombre de pairs non seeder(leechers)
Peers	La Liste des Peers, chaque Peer avec une clé.
Peer id	L'Identifiant de client (Peer)
IP	Adresse IP de client (Peer)
Port	Port de client (Peer)

Tableau 2.3 : Le contenue de la réponse http*.*

c Connexion aux paires (téléchargement de fichier)

Une fois la liste des pairs obtenue, le client va chercher à se connecter à ceux-ci par l'intermédiaire des 2 protocoles, TCP et BitTorrent protocole [12].

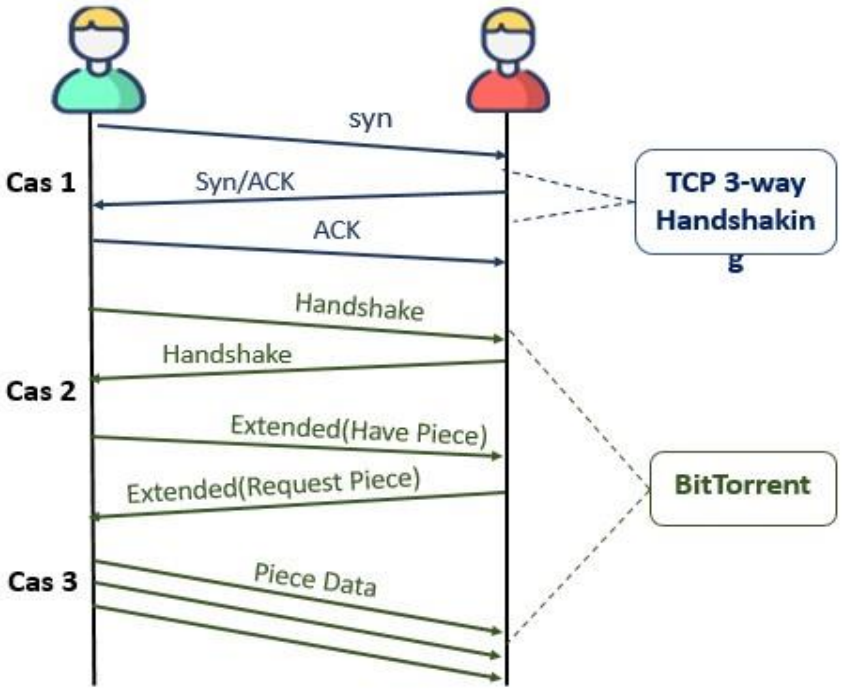


Figure 2.2 : Protocole inter-clients (TCP-Protocole BitTorrent).

❖ **Premier cas : la session TCP**

Un programme client BitTorrent tente de se connecter à une session en envoyant un paquet TCP SYN à des pairs qui existent dans la liste des pairs.

Dans ce cas, il y a deux tentatives de connexion possibles : soit il n'y a pas de sessions connectées (renvoi d'un paquet TCP RST), soit il n'y a qu'une session TCP (connectée via une poignée de main TCP à 3 voies).

❖ **Deuxième cas : le protocole BitTorrent**

Le paquet BitTorrent **« handshake »** inclut la valeur « BitTorrent Protocol » qui représente la signature du protocole BitTorrent, ainsi que l'identifiant du pair « Peer-ID », l'échange de ce paquet suppose la volonté du pair de partager le fichier.

Le paquet **« Extended have pièce »**, Le message « Avoir pièce » signifie Je peux envoyer les pièces que vous demandez en examinant la liste des pièces qu'ils ont. Ainsi, les pairs peuvent demander des pièces à d'autres pairs qui envoient le message « Have, Pièce » à tout moment en envoyant le message « Request, Pièce ».

❖ **Troisième cas : L'échange des données**

Après que toutes les étapes préalables au partage de fichiers ont été accomplies. Un enquêteur peut enregistrer tous les paquets transmis à chaque pair pour cartographier l'environnement de partage et échanger les données d'un fichier.

- **Les différents messages BitTorrent**

Message « **choke** » : est envoyer Lorsqu'un client refuse d'envoyer des données à un autre pair, l'opération inverse, consistant à accepter les requêtes du pair, est appelée « **unchoke** » [12].

2.2.5 Système sans tracker (trackerless)

Le protocole BitTorrent comporte des vulnérabilités, principalement liées au risque de suppression d'un serveur, étant donné que tous les clients partageant le même fichier se connectent au même serveur. Dans le cas où le tracker est indisponible, le téléchargement du fichier devient impossible [13].

Une autre extension importante a été développée pour remédier à cette faiblesse du protocole, en éliminant les trackers qui centralisent le réseau. Les nouvelles versions de BitTorrent prennent en charge des connexions sans tracker (trackerless). Dans ce cas, le client met en œuvre une table DHT (Distributed Hash Table) pour identifier et stocker la liste des différents clients partageant le fichier qui l'intéresse. Cette DHT repose sur le protocole Kademlia [13].

2.3 Protocole IPFS

2.3.1 Définition

IPFS (Interplanetary File System) est un système de stockage et de partage de fichiers paire à pair, créé en 2015 par Juan Benet. Il s'agit d'une technologie clé du Web décentralisé (dWeb) [14].

IPFS se repose sur la fragmentation des fichiers en plusieurs parties plus petites, qui sont ensuite réparties sur un réseau de nœuds, permettant aux utilisateurs d'accéder aux fichiers depuis n'importe quel endroit sur le réseau, sans dépendre d'un serveur central [14].

2.3.2 Différence entre HTTP (Web traditionnel) et IPFS (dWeb)

HTTP utilise URL (Uniform Resource Locator) pour référencer le contenu. Les URL indiquent l'emplacement spécifique du contenu sur Internet, ce qui signifie qu'en cas de déplacement ou de suppression du contenu, l'URL devient invalide [15].

En revanche, IPFS utilise un adressage basé sur le contenu, ce qui signifie que le contenu est référencé en fonction de son hachage de contenu plutôt que de son emplacement physique [15].

Cette approche garantit que le contenu IPFS reste immuable et permanent, même si le nœud d'origine qui l'a partagé se déconnecte.

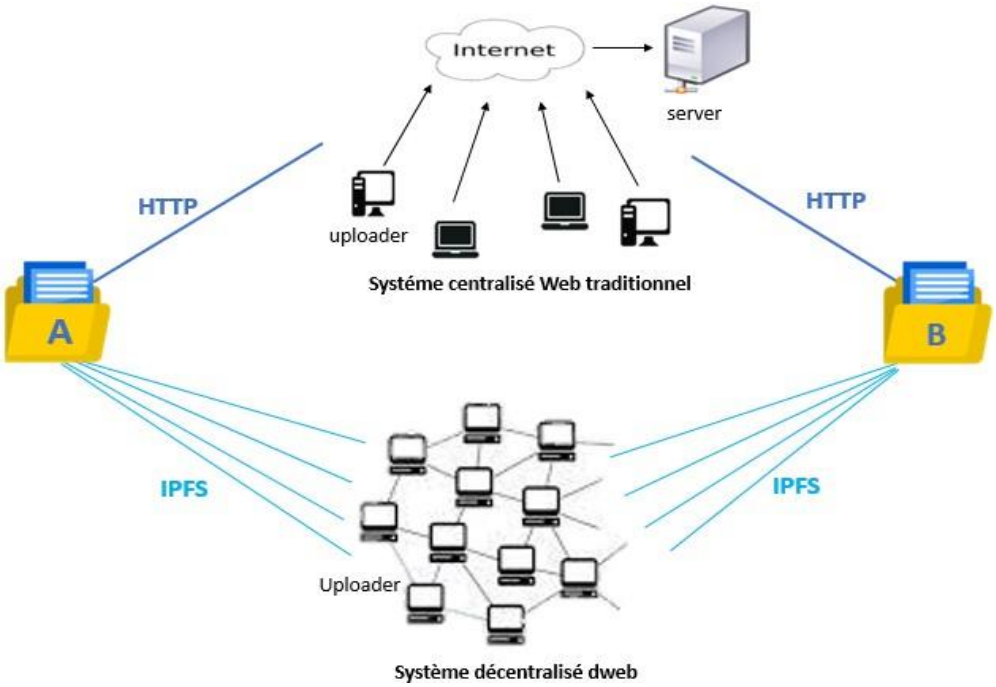


Figure 2.3 : Comparaison entre IPFS et HTTP.

2.3.3 Les sous-systèmes IPFS

Sous-système	But
CID	C'est une valeur unique pour identifier les fichiers stockées sur le réseau IPFS
IPNS	Un système permettant d'adresse un fichier ou un répertoire mutable
DAG	Il Est utilisé pour représenter la structure des données en forme d'arbre
IPLD	Il fournit un moyen de relier et d'adresser des données à travers différent système
DHT	Routage de contenu, liaison entre CID et adresse
MDNS	Découvert rapide, et efficace des nœuds
Libp2p	Connectivité Peer-to-Peer

Tableau 2.4 : Les sous-systèmes IPFS.

2.3.4 Principe de fonctionnement

IPFS permet aux utilisateurs de stocker et de récupérer des fichiers en divisant chaque fichier en blocs de données de taille supérieure à 256 Ko [16].

Ces blocs sont hachés et distribués sur des nœuds IPFS, qui peuvent être des ordinateurs ou des serveurs appartenant aux utilisateurs. Lorsqu'un utilisateur souhaite récupérer un fichier à partir d'IPFS, il interroge les nœuds en utilisant le hash unique du fichier (CID) [16].

Le nœud qui possède une copie du fichier peut alors le transmettre à l'utilisateur. Si plusieurs nœuds ont une copie du fichier, l'utilisateur peut le récupérer depuis n'importe lequel d'entre eux [16].

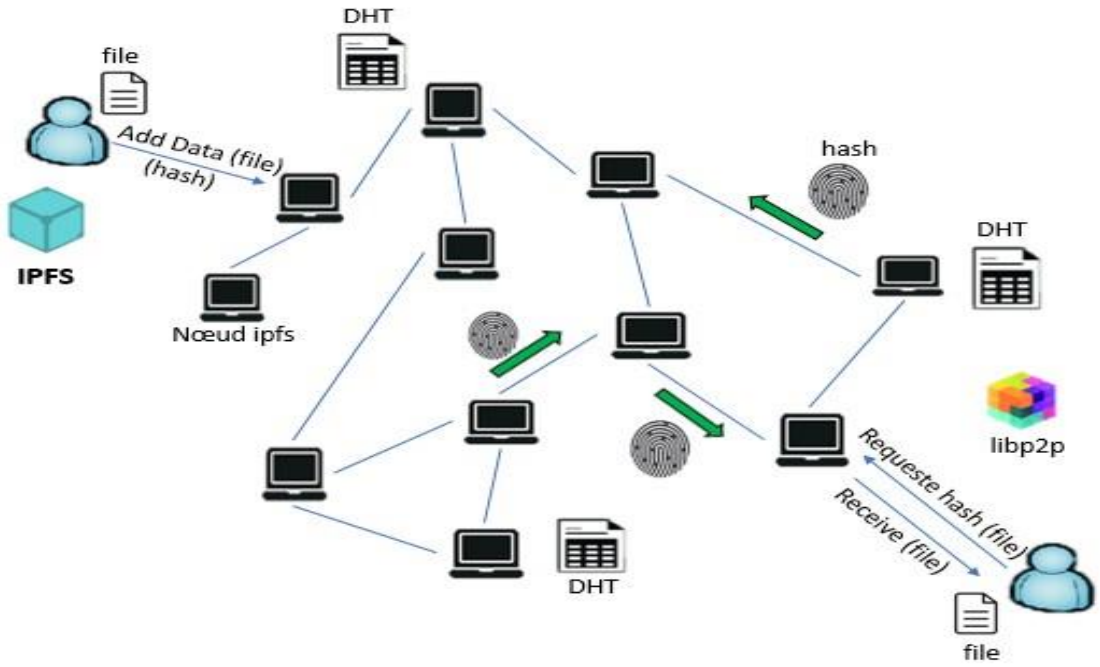


Figure 2.4 : Principe de fonctionnement "Protocole IPFS".

a Adressage de contenu

IPFS est basé sur un système d'adressage pour identifier des fichiers. Chaque fichier est identifié par une empreinte cryptographique unique, comportant 24 caractères appelés content identifier (CID). Cette empreinte est calculée à l'aide d'une fonction de hachage (SHA-256) à partir des données contenues dans le fichier, cela permet de vérifier l'intégrité d'un fichier [16].

Figure 2.5 : Adressage de contenu (CID : Content Identifier).

La modification d'un fichier modifie son hachage, et par conséquent son CID, pour cela IPFS propose l'idée d'IPNS (Interplanetary Name System).

IPNS permet de créer des adresses permanentes qui pointent vers la dernière version d'un fichier stocké sur IPFS en utilisant des noms de domaine plutôt que des adresses de hachage. Il utilise un système de clés publiques et privées pour créer un enregistrement DNS décentralisé qui permet de mapper les noms de domaine aux CID. Ainsi, même si le contenu du fichier est modifié, l'adresse IPNS reste la même, ce qui permet un accès facile et constant au contenu [17].

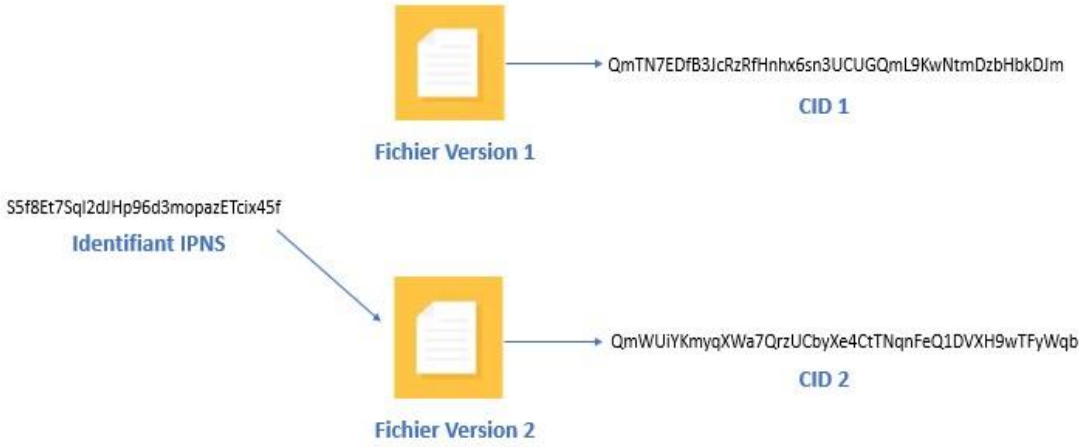


Figure 2.6 : Adressage de contenu (IPNS: Interplanetary Name system).

b La représentation des contenus

IPFS utilise une structure de données appelée DAG (Directed Acyclic Graph), plus précisément des arbres de hachage (arbre de Merkle) [18].

Un arbre de Merkle est une structure arborescente qui contient des empreintes cryptographiques des données.

Cette structure permet de reconstituer l'ensemble du contenu à partir de la racine de l'arbre. L'objectif principal est de vérifier l'intégrité d'un ensemble de données sans avoir à les vérifier individuellement [18].

Toutefois, la structure DAG ne fournit pas de mécanisme pour la récupération de données. C'est pourquoi IPFS utilise IPLD (Interplanetary Linked Data).

IPLD offre la possibilité de représenter les relations entre les données, telles que les répertoires de fichiers et d'autres structures hiérarchiques, en utilisant l'adressage de contenu (CID), il permet également de lier des données entre elles, ce qui simplifie la recherche et la récupération de données.

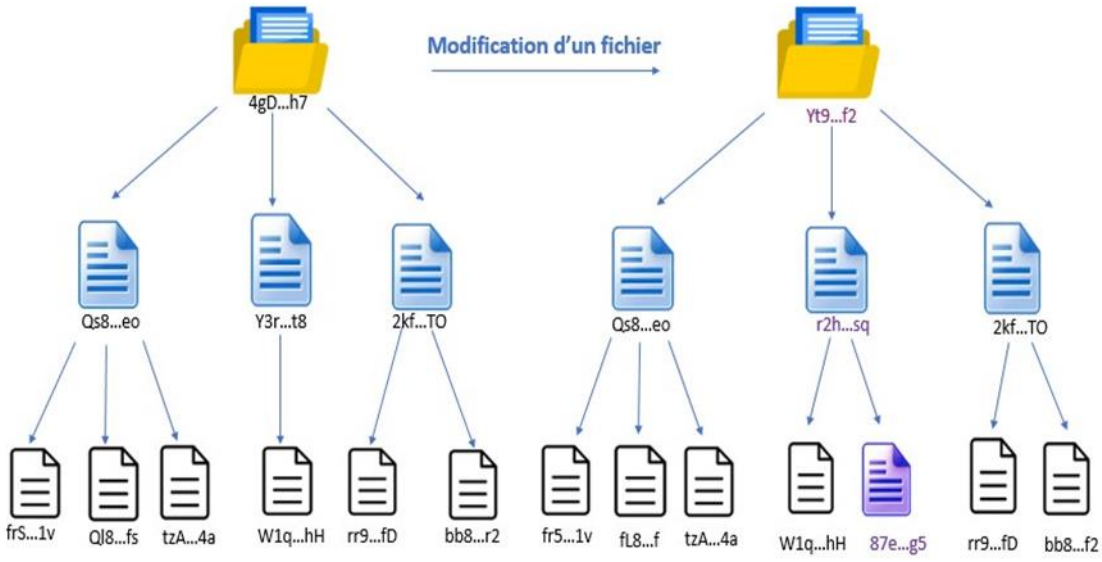


Figure 2.7 : La représentation de contenu "Arbre de Merkle".

c Découverte de nœuds et de contenu

IPFS simplifie la recherche des pairs qui stockent les données recherchées en utilisant le principe des tables de hachage distribuées (DHT) basé sur Kademlia. La gestion de ce mécanisme est assurée par la librairie LibP2P [19].

Chaque nœud dans IPFS possède un identifiant unique et une adresse, ainsi qu'une table de clés/valeurs. Cette table permet d'associer les identifiants de contenu (CID) aux identifiants des nœuds qui hébergent ces contenus. Lorsqu'un nœud souhaite accéder à un contenu, il récupère les identifiants des nœuds "l'adresse IP" où le contenu est présent et téléchargera les données directement à partir d'eux [19].

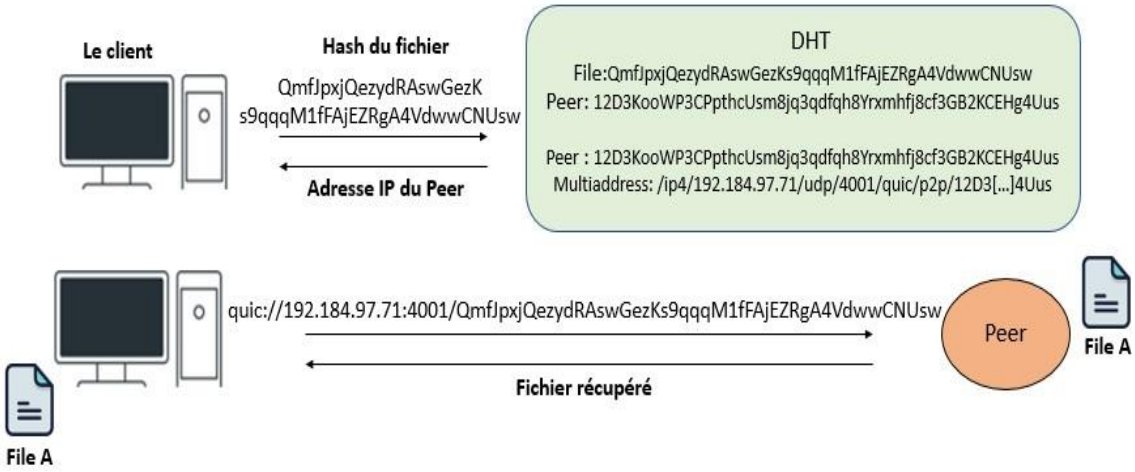


Figure 2.8 : Découverte de nœuds et de contenu "DHT".

IPFS utilise aussi le protocole MDNS (Multicast Domain Name System) pour simplifier la recherche de pairs. Grâce à MDNS, les nœuds IPFS peuvent s'identifier mutuellement et détecter d'autres nœuds IPFS présents sur le réseau sans dépendre d'un serveur centralisé.

d IPFS et Ethereum

Ethereum, une plateforme décentralisée basée sur la technologie de la Blockchain, offre une sécurité élevée. Cependant, le stockage direct de fichiers sur Ethereum peut être coûteux en raison de la charge de traitement associée [20].

Pour résoudre ce problème, une solution a été mise en place en intégrant IPFS (Interplanetary File System) pour héberger les données et fournir des URL immuables pointant vers ces données sur Ethereum. Cette approche réduit considérablement les coûts en publiant les fichiers sur IPFS et en enregistrant leur adresse basée sur le contenu dans la Blockchain Ethereum [20].

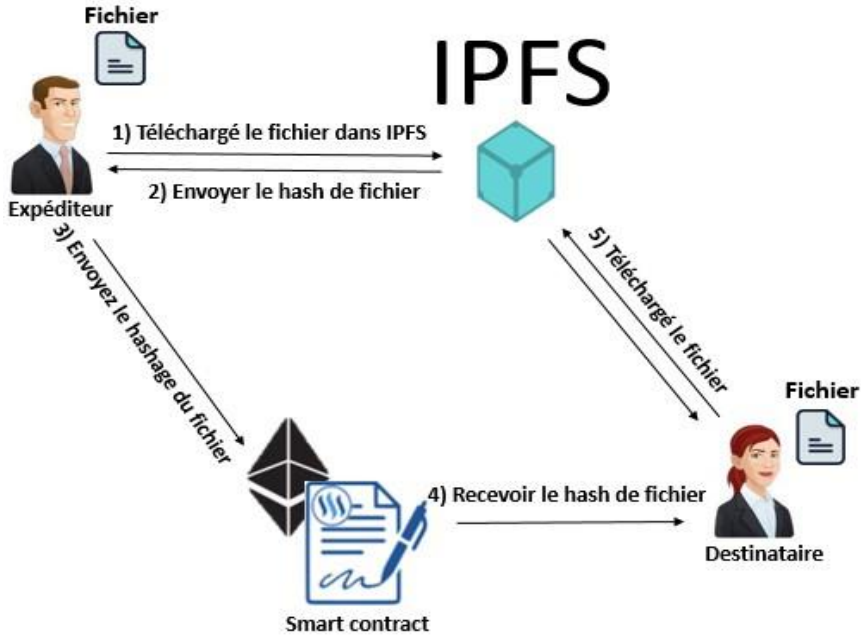


Figure 2.9 : Fonctionnement Ethereum avec IPFS.

2.4 Comparaison entre IPFS et BitTorrent

	BitTorrent	**IPFS**
Cas d'utilisation	Partage de fichier	Stockage de fichier
Modèle de données	DAG merkle	DAG merkle
Pile réseau	TCP/IP	LibP2P
identificateur	Fichier torrent	CID(content identifier)
Composition de l'adresse	Nom de fichier + hachage sh1	CID
Algorithme de hachage	Sha-256	Sha-256

Tableau 2.5 : Comparaison entre IPFS et BitTorrent [21].

- **Bande passante**

IPFS peut consommer plus de ressources machine en terme d'espace disque et de bande passante en raison de son système de stockage distribué, tandis que BitTorrent est conçu pour utiliser moins de ressources en comparaison avec IPFS grâce à la découpe des fichiers en plusieurs morceaux.

Cependant la quantité de ressources requises dépendra toujours de la taille des fichiers, du nombre de pairs impliqués et de la qualité de la connexion internet.

- **Temps de latence**

Dans IPFS, le temps de latence peut être plus élevé en raison de la distribution des fichiers sur plusieurs nœuds du réseau. Lorsqu'un utilisateur souhaite accéder à un fichier, il doit localiser les nœuds qui détiennent les parties du fichier et les télécharger à partir de ces nœuds, ce qui peut prendre du temps, surtout si les parties du fichier sont dispersées sur de nombreux nœuds.

En revanche, dans BitTorrent, le temps de latence peut être plus court car le fichier est partagé entre les pairs qui téléchargent et partagent le même fichier. Les utilisateurs peuvent se connecter directement aux pairs qui possèdent les parties spécifiques du fichier dont ils ont besoin, ce qui permet des téléchargements plus rapides.

2.5 Conclusion

Dans ce chapitre, on a examiné en détail les réseaux Peer-to-Peer, en se concentrant plus particulièrement sur les caractéristiques des protocoles BitTorrent et IPFS ainsi que leurs principes de fonctionnement.

Bien que ces protocoles aient marqué une avancée significative dans le domaine des réseaux Peer-to-Peer, la sécurité reste un problème majeur. Malheureusement, de nombreuses applications P2P sont utilisées à des fins illégales. C'est pourquoi nous proposons une méthode fiable de détection de l'utilisation de ces protocoles (BitTorrent, IPFS) afin de prévenir leur utilisation abusive dans les entreprises.

Chapitre 3

Extraction des empreintes

P2P

Chapitre 3 Extraction des empreintes P2P

3.1 Introduction

Les utilisateurs qui souhaitent télécharger et stocker des données sur Internet peuvent considérer les clients du réseau Peer-to-Peer tels que BitTorrent et IPFS comme des solutions idéales. Cependant, l'utilisation de ces clients au sein d'une entreprise peut présenter plusieurs risques en termes de sécurité et de problèmes juridiques. Pour cette raison, les entreprises commencent à prêter attention aux risques de l'utilisation du P2P dans leurs propres réseaux internes.

À cet effet, on propose dans le cadre de notre travail une solution qui vise à détecter l'utilisation des réseaux Peer-to-Peer. Cette solution repose sur un ensemble d'étapes qu'on décrit ci-dessous et qui seront détaillées dans ce chapitre.

- La capture des paquets venant du trafic de BitTorrent et IPFS avec un outil d'analyse approfondie des paquets "deep packet inspection".
- L'analyse détaillée des paquets capturés.
- L'extraction des empreintes numériques des réseaux BitTorrent et IPFS.

3.2 L'Objectif de notre recherche

Notre recherche a pour but de détecter et bloquer l'utilisation des réseaux BitTorrent et IPFS dans une entreprise, en créant des règles avec un système IDS/IPS suricata à partir des empreintes numériques qu'on a extraites à partir du logiciel d'analyse des paquets Wireshark.

3.3 Plan de travail

Afin d'atteindre notre objectif, on a mis en œuvre les démarches suivantes :

- **L'étude théorique :** afin de comprendre le fonctionnement en détails des deux réseaux.
- **Capteur :** cette phase permet de capturer le trafic BitTorrent et IPFS en utilisant l'analyseur de paquets Wireshark.
- **L'analyse des données :** cette partie permet d'analyser les paquets capturés afin d'extraire les signatures spécifiques à BitTorrent et IPFS.
- **La détection :** implémentation des signatures extraites dans la phase d'analyse au niveau de système IDS/IPS "Suricata" qui va nous permettre d'observer de près l'efficacité de la solution proposée.

3.4 Matériels utilisés

Pour la réalisation de notre travail, on a utilisé le matériel suivant :

	Matériaux	**logiciels**
PC serveur	Un ordinateur bureau Xpress avec un processeur Intel® Core ™ i3-3220U @3.30GHZ x 4 avec une RAM de 8.0 GO et 2 cœurs, fonctionnant sous le système d'exploitation Linux Ubuntu 22.04 LTS 64 bit	-Wireshark version : 3.6.2 -Suricata version: 6.0.12
PC client	Un ordinateur portable LENOVO avec un processeur Intel ® Core ™ i3-7020U @ 2.30 GHZ 2.30 GHZ avec une RAM 4 GO et 2 cœurs, fonctionnant sous le système d'exploitation Windows 10 professionnel version 22H2 64 bit	-BitTorrent version : 7.11.0 - IPFS Desktop version : 0.28

Tableau 3.1 : Equipements et logiciels utilisés**.**

3.5 Environnement

3.5.1 Architecture client/serveur

La figure ci-dessous nous montre l'architecture client/serveur réalisée.

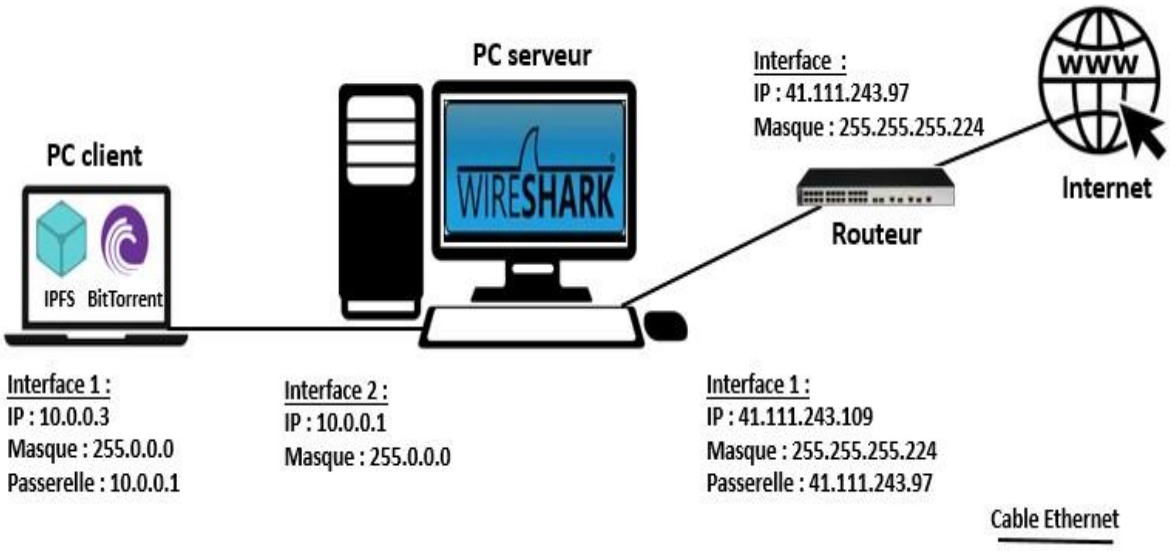


Figure 3.1 : Architecture client/serveur***.***

- Le PC serveur a deux interfaces réseau :

La première interface, configurée avec l'adresse publique : 41.111.243.109, a été connectée à l'internet via un routeur doté d'un câble Ethernet.

La deuxième interface, configurée avec l'adresse : 10.0.0.1, a été connectée au réseau local (10.0.0.0/8) via un adaptateur USB Ethernet branché sur le port USB.

- Le PC client est connecté au réseau local (10.0.0.0) par l'adresse : 10.0.0.3 et la passerelle : 10.0.0.1 (adresse du PC serveur).

a Serveur Ubuntu

Pour le serveur, on a choisi le système d'exploitation Ubuntu 22.04 LTS Linux. Ubuntu est un système d'exploitation Linux basé sur la distribution Debian. Il est apprécié dans le monde entier en raison de sa simplicité d'utilisation, de sa gratuité et de sa nature open source. Une caractéristique importante d'Ubuntu est la disponibilité

de versions à long terme appelé "LTS" (Long Term Support) qui est maintenue pendant cinq ans [22].

b Logiciel BitTorrent

Le client BitTorrent est principalement conçu pour permettre des services de communication d'égal à égal via le protocole BitTorrent pour le partage, le téléchargement et l'amorçage de données vers un groupe de pairs connectés à l'échelle mondiale et alimentés par le même protocole. Ce client est largement utilisé en raison de sa légèreté et de sa simplicité d'utilisation. La version actuelle de BitTorrent pour Windows est la « 7.11.0 », mise en ligne le 23/12/2022 [23].

Figure 3.2 : Logo BitTorrent [23].

C Logiciel IPFS Desktop

IPFS Desktop est une application conçue pour exécuter un nœud IPFS local et permettre l'interaction avec le réseau IPFS. Il est basé sur le protocole Peer-to-Peer, il offre une interface conviviale et des fonctionnalités avancées pour la gestion des fichiers, la recherche de contenu et le partage de données via IPFS. La version actuelle de IPFS Desktop pour Windows est la « 0.28.0 », mise en ligne le 10/05/2023 [24].

Figure 3.3 : Logo IPFS Desktop [24].

3.6 Outil d'analyse & capture « Wireshark »

Wireshark est un outil open source pour profiler le trafic réseau et analyser les paquets. Un tel outil est souvent appelé analyseur de réseau, analyseur de protocole réseau ou renifleur, il est utilisé dans l'analyse des réseaux informatiques et le développement de protocoles [25].

La capture de paquets peut fournir des informations sur des paquets individuels tels que l'heure de transmission, la source, la destination, le type de protocole et les données d'en-tête [25].

- Parmi ces fonctionnalités les plus intéressantes :
  - ✓ Inspection approfondie de centaines de protocoles.
  - ✓ Capturez des données de paquets en direct (temps réel).
  - ✓ Multiplateforme : Fonctionne sous Windows, Linux, MacOs...
  - ✓ Navigateur de paquets standard à trois volets.
  - ✓ Affichage colorisé des paquets en fonction des filtres.
- Principe de fonctionnement :

L'analyseur de paquets Wireshark suit les processus suivants :

- **Collecter :** sélectionnez une interface pour la surveillance du trafic et la capture des paquets réseau.
- **Convertir :** une interface graphique est utilisée pour convertir les paquets dans une forme d'information facilement compréhensible.
- **Analyser :** l'utilisation de fonctionnalités statistiques et graphiques permettent d'analyser le trafic réseau en ce qui concerne les paquets, les protocoles, les données chiffrées et bien d'autres éléments.

Après la capture des paquets, l'interface graphique de Wireshark s'affiche comme suit :

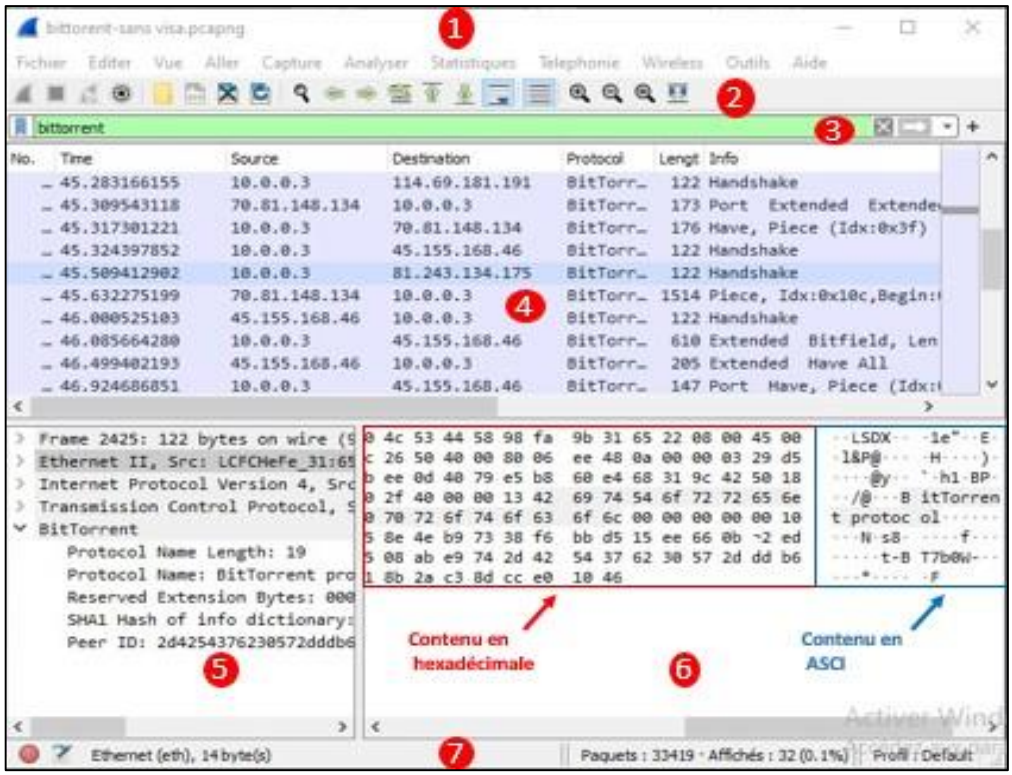


Figure 3.4 : Interface Wireshark.

01- Le menu : Est utilisé pour démarrer les actions.

02- La barre d'outils principale : Fournit un accès rapide aux éléments fréquemment utilisés à partir du menu.

03- La barre d'outils de filtrage : Permet aux utilisateurs de définir des filtres d'affichage pour filtrer les paquets affichés.

04- Le volet de la liste des paquets : Affiche la liste des paquets capturés.

05- Le volet des détails des paquets : Affiche les détails sur les paquets sélectionnés de la liste du haut.

06- Le volet d'octets de paquet : Reproduit le contenu en hexadécimal du même paquet.

07- La barre d'état : Affiche des informations sur l'état actuel du programme et les données capturées.

3.7 La structure d'analyse

L'inspection approfondie des paquets à l'aide de logiciel « Wireshark » va nous permettre de détecter les protocoles BitTorrent et IPFS en étudiant les logiciels BitTorrent et IPFS Desktop.

3.7.1 BitTorrent

Pour télécharger un fichier via BitTorrent, vous devez d'abord obtenir le fichier .torrent correspondant. Ensuite, vous utilisez un client BitTorrent pour ouvrir ce fichier et démarrer le processus de téléchargement. Donc les états de fonctionnement de ce client sont représentés ci-dessous :

- Obtention du fichier Metainfo (.torrent)
- Exécution du fichier Metainfo : Connexion au tracker
- Connexion aux pairs
- Mode crypté : Forcé

a Obtention du fichier Metainfo (.torrent)

La première étape du téléchargement BitTorrent consiste à récupérer le fichier Metainfo depuis un site de torrent. Parmi les sites Web populaires qui offrent des téléchargements de fichiers torrent, on trouve :

- https://cpasbien.com
- https://1337x.com
- https://RARBG.com

Exemple d'un fichier Metainfo obtenu à partir du site « cpasbien ».

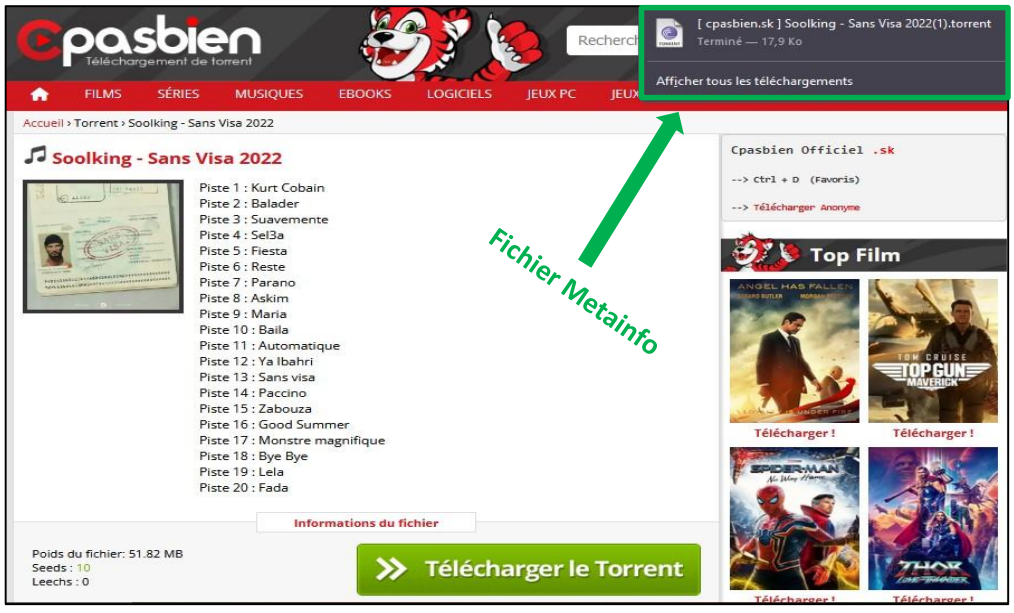


Figure 3.5 : Téléchargement du fichier Metainfo site « cpasbien »**.**

b Exécution du fichier Metainfo : Connexion au tracker

Après avoir récupéré le fichier Torrent, l'utilisateur doit l'exécuter pour établir une connexion avec le tracker en envoyant des requêtes **HTTP/GET**. Cette connexion permet d'obtenir des informations sur tous les clients qui partagent le fichier.

À cette étape, on peut visualiser les trackers qui gèrent le téléchargement, activer ou désactiver la DHT, limiter la consommation de bande passante et effectuer d'autres configurations de téléchargement.

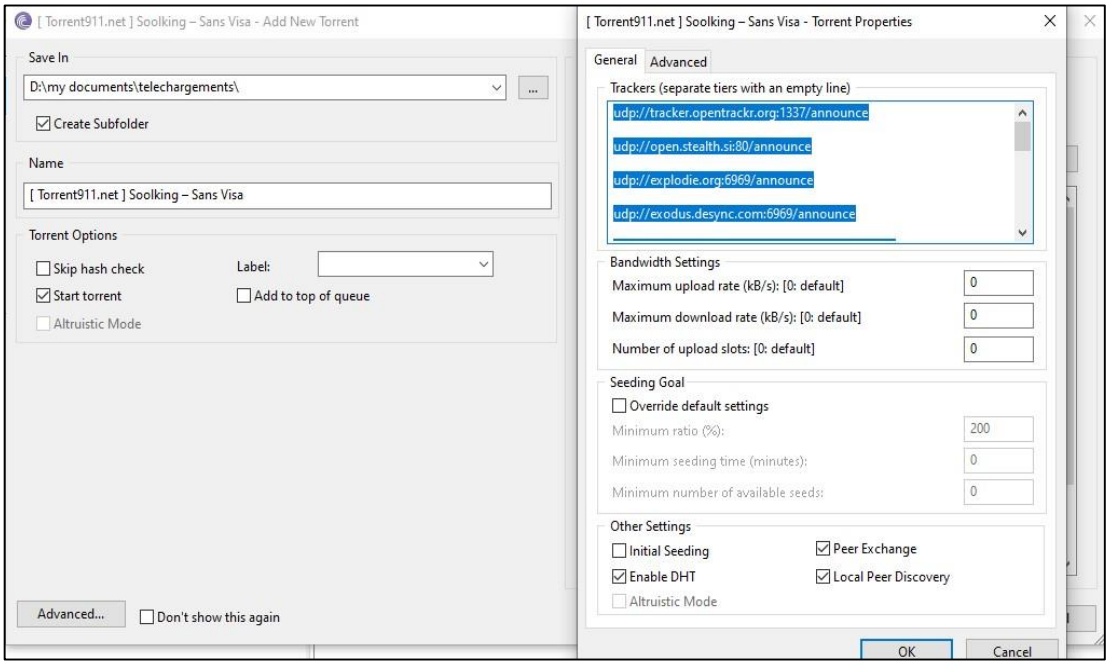


Figure 3.6 : Fenêtre de configuration du téléchargement torrent.

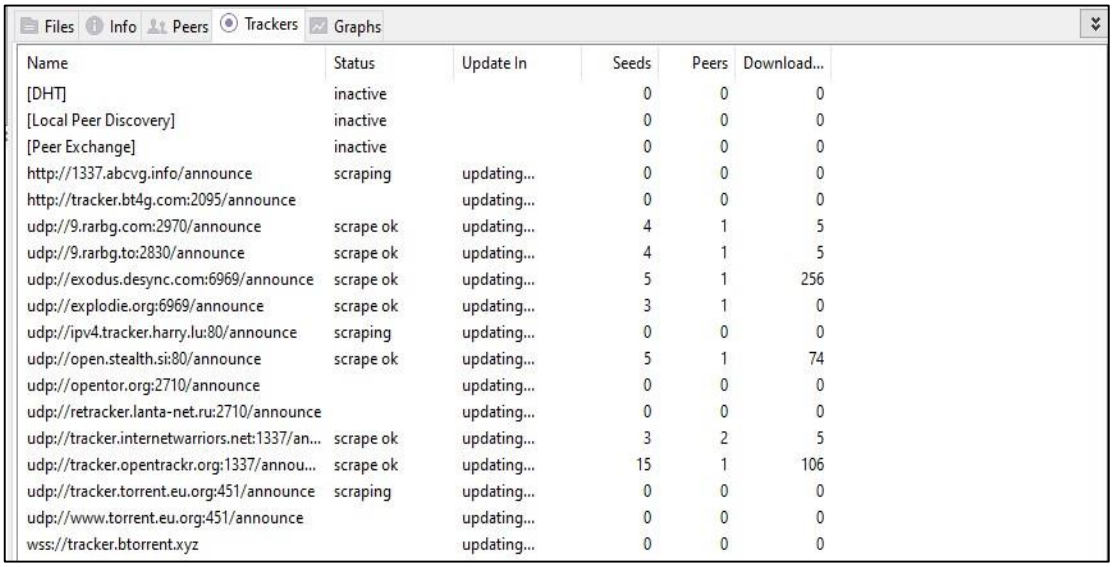


Figure 3.7 : La Liste des trackers**.**

c Connexion aux pairs

Lorsque la connexion aux pairs est établie et que le téléchargement débute, le logiciel affiche toutes les informations relatives au fichier ainsi que leurs sources correspondantes.

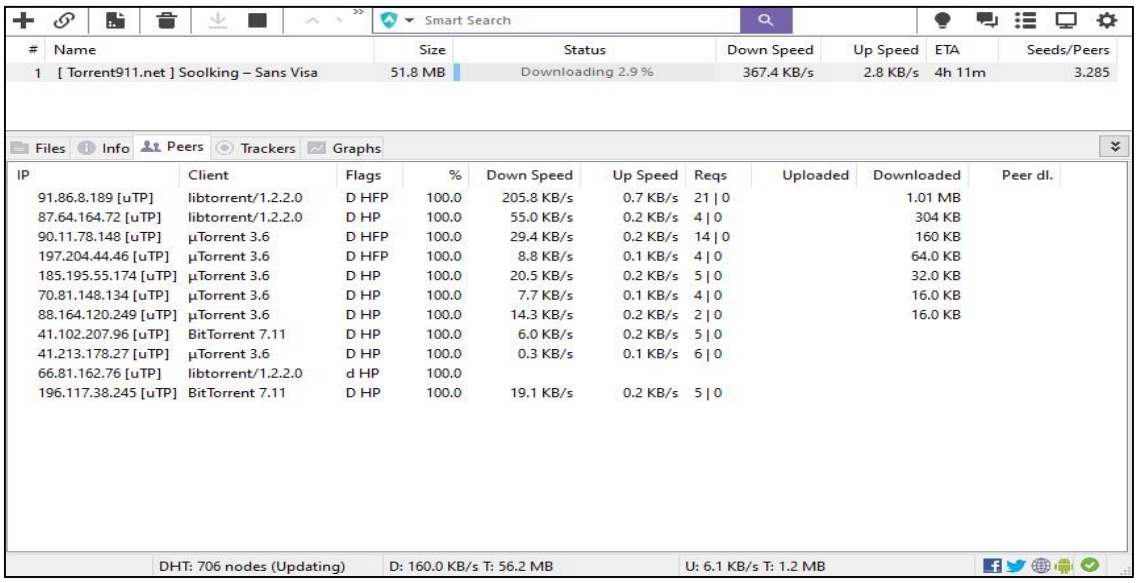


Figure 3.8 : Interface graphique du logiciel BitTorrent*.*

d Mode crypté : Forcé

Les clients transmettent généralement des paquets en clair. Il existe des extensions de chiffrement qui permettent de crypter la communication entre les pairs afin de protéger la confidentialité de nos messages. De sorte que la communication cryptée entre les pairs est limitée aux pairs qui sont cryptés uniquement. Les sources de fichiers disponibles peuvent donc être restreintes.

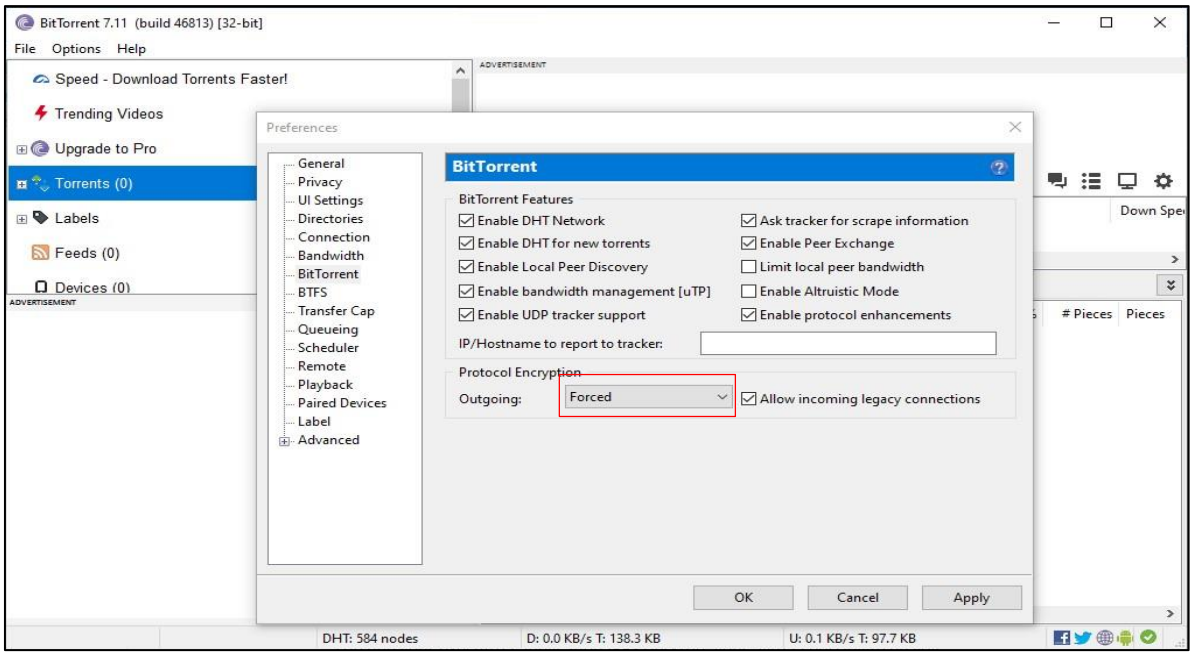


Figure 3.9 : Extensions du protocole BitTorrent sous BitTorrent.

3.7.2 IPFS DESKTOP

La structure d'analyse du trafic pour le client IPFS Desktop est comme suit :

- Etablissement de connexion
- Ajouter un fichier
- Accomplissement d'une recherche d'un fichier
- Consulter le contenu d'un fichier

a Etablissement de connexion

Lorsque l'on exécute l'application IPFS Desktop, on a la possibilité de voir l'état de connexion de notre nœud IPFS. L'application affiche des informations telles que l'identifiant de notre nœud (Peer ID), des détails sur les pairs auxquels on est connecté, la bande passante utilisée, etc. Ces informations nous aident à déterminer si notre nœud est correctement connecté au réseau IPFS.

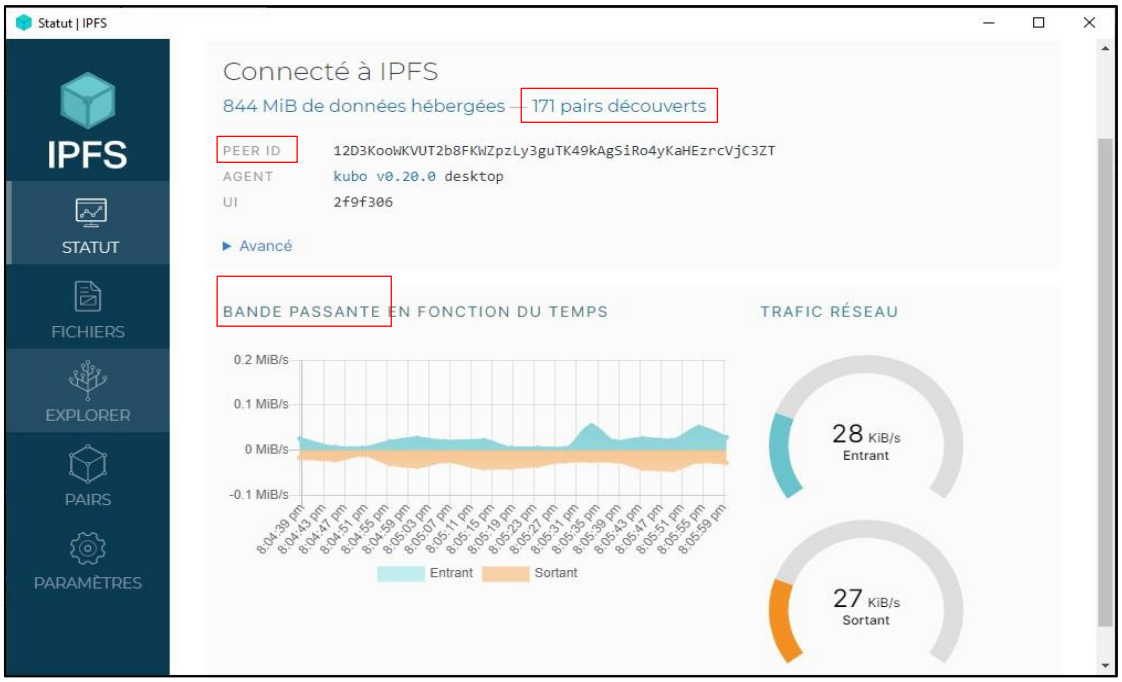


Figure 3.10 : Statut de connexion "client IPFS Desktop".

b Ajouter un fichier

Une fois que notre nœud IPFS local est connecté, on peut commencer à utiliser les fonctionnalités de l'application IPFS Desktop. On peut ajouter des fichiers à partir de notre système de fichier local en utilisant l'option « **importer** ». Chaque fichier ajouté reçoit un hachage unique appelé CID qui identifie ce fichier de manière précise sur IPFS.

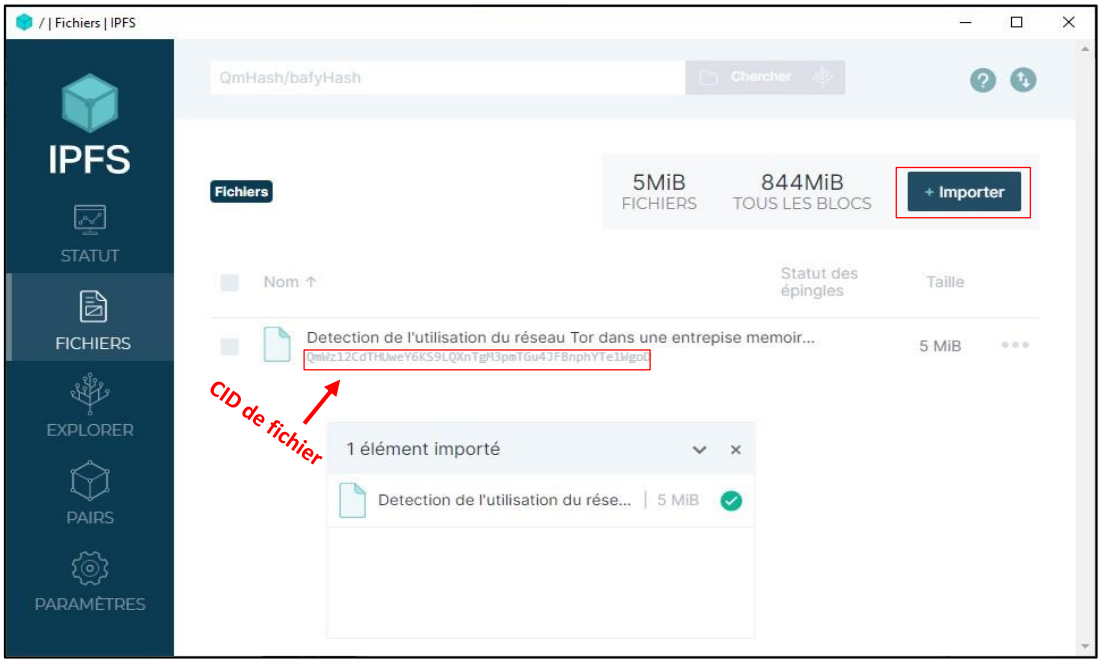


Figure 3.11 : Téléchargement du fichier via IPFS Desktop.

c Accomplissement d'une recherche d'un fichier

Pour effectuer une recherche d'un fichier sur IPFS, on doit tout d'abord obtenir le CID du fichier que l'on recherche. Ensuite, on clique sur l'icône « chercher », ce qui nous permet de lancer une recherche. Si le fichier est disponible et que des nœuds dans le réseau le partagent, IPFS Desktop affiche les résultats de la recherche avec des informations telles que la taille du fichier, son hash et d'autres métadonnées.

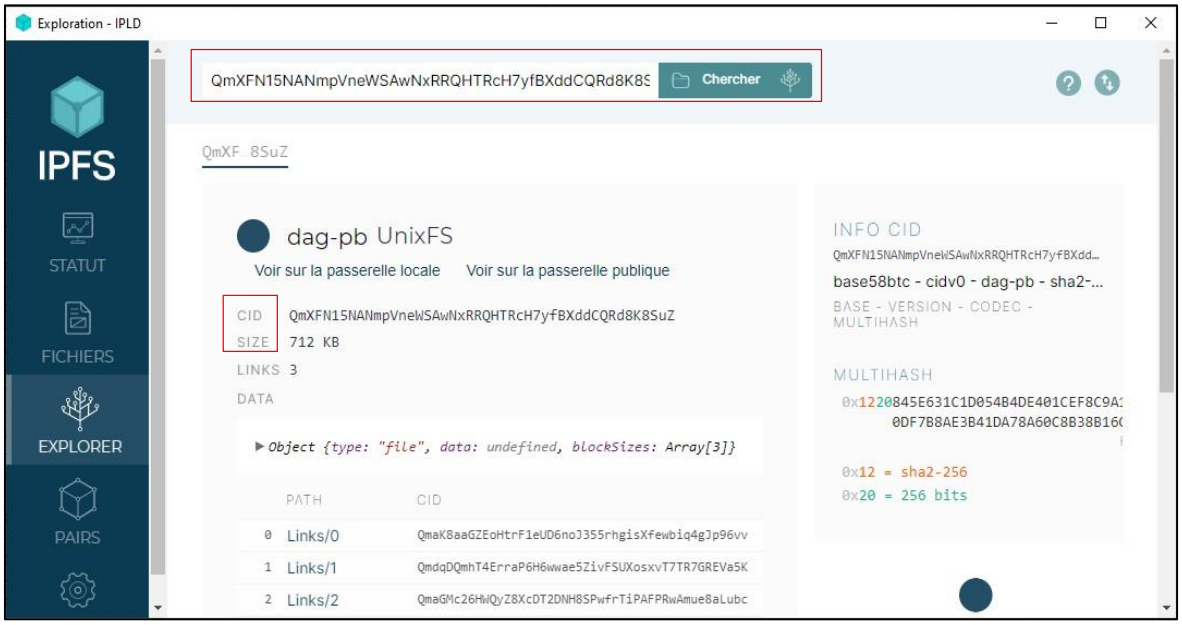


Figure 3.12 : Résultats de recherche avec le client IPFS Desktop**.**

d Consulter le contenu d'un fichier

On peut utiliser l'application IPFS Desktop pour visualiser les fichiers stockés sur IPFS, que ce soit en les affichant directement dans l'application ou en les ouvrant avec un navigateur Web.

- À l'aide de l'application IPFS Desktop : une fois qu'on a ajouté le fichier, on peut le sélectionner dans l'interface de l'application pour afficher son contenu.

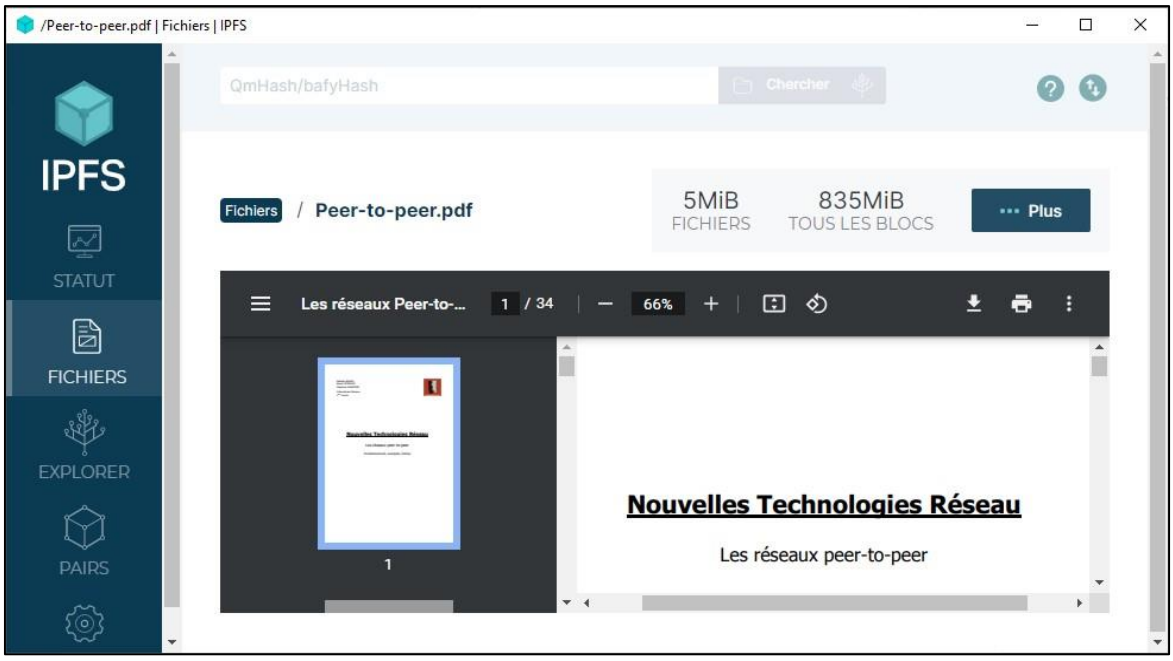


Figure 3.13 : Accès au contenu d'un fichier via le logiciel IPFS Desktop**.**

- À l'aide d'un navigateur Web, on peut utiliser une passerelle IPFS pour accéder au fichier via le navigateur. Les passerelles IPFS sont des serveurs qui permettent de récupérer les contenus IPFS via une URL standard. L'URL est de la forme « **https://ipfs.io/ipfs/<CID>** », où **<CID>** est remplacé par le CID du fichier.

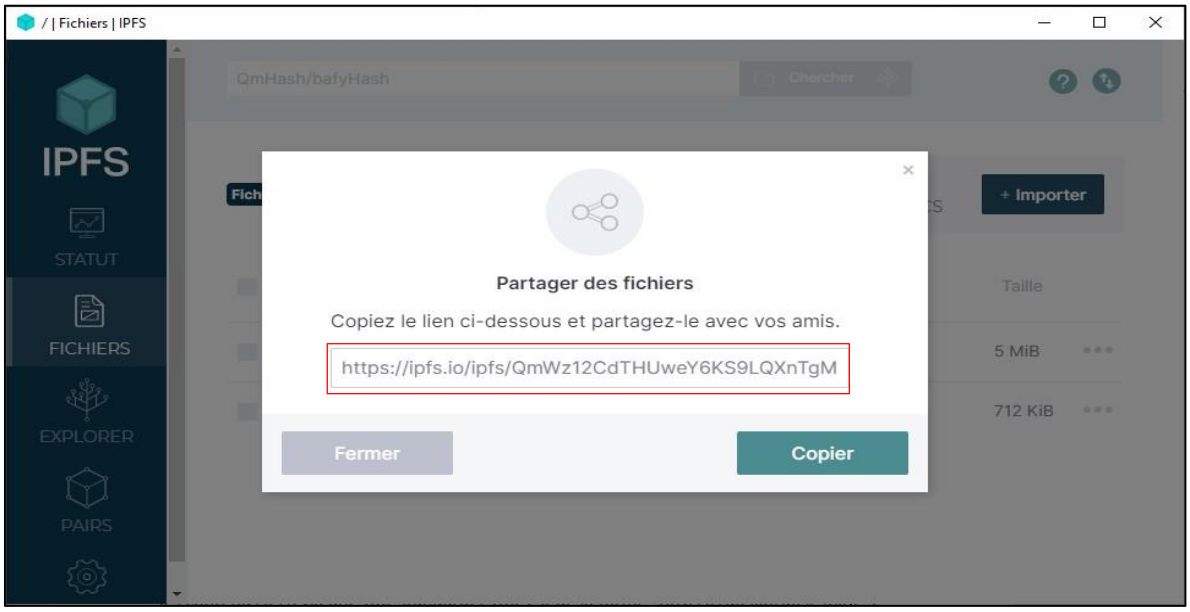


Figure 3.14 : L'URL utilisée pour récupérer le fichier via un navigateur Web**.**

3.8 Analyse et capture du trafic réseau

Pour identifier l'utilisation de chaque protocole, une analyse approfondie des paquets pour chaque état de fonctionnement précédent sera effectuée afin d'extraire les empreintes numériques correspondantes.

3.8.1 BitTorrent

a Obtention du fichier Metainfo

L'utilisation du site web « cpasbien » permet l'obtention du fichier MetaData, La figure (3.15) montre clairement la trace du site « cpasbien » lorsque l'utilisateur veut télécharger le fichier (.torrent).

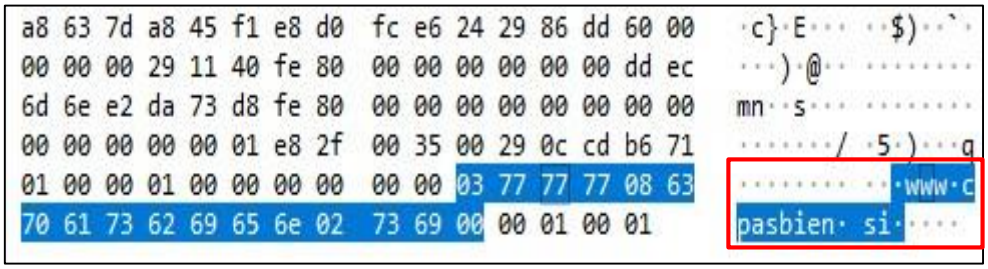


Figure 3.15 : La trace du site Web cpasbien.

b Exécution du fichier Metainfo : Connexion au tracker

Lors de l'exécution du fichier Metainfo, le client BitTorrent initie la connexion avec le tracker. Les paquets capturés durant cette étape sont illustrés dans la figure (3.16).

No.	Time	Source	Destination	Protocol	Lengt	Info
2127	42.096311943	10.0.0.3	172.67.130.211	HTTP	257	GET /scrape?info_hash=%8eN%b9s
2151	42.198226214	172.67.130.211	10.0.0.3	HTTP	59	HTTP/1.1 200 OK (text/plain)
2251	43.084183743	10.0.0.3	172.64.134.19	HTTP	251	GET /scrape?info_hash=%8eN%b9s
2267	43.134765655	172.64.134.19	10.0.0.3	HTTP	59	HTTP/1.1 200 OK (text/html)
14990	55.953812646	10.0.0.3	172.64.135.19	HTTP	431	GET /announce?info_hash=%8eN%b
15051	56.005295466	172.64.135.19	10.0.0.3	HTTP	59	HTTP/1.1 200 OK (text/html)
17150	57.939314570	10.0.0.3	172.67.130.211	HTTP	437	GET /announce?info_hash=%8eN%b
17269	58.045914858	172.67.130.211	10.0.0.3	HTTP	59	HTTP/1.1 200 OK (text/plain)

Figure 3.16 :Les requêtes principales Peer-Tracker.

- Le client établit une connexion avec le tracker « **172.67.130.211** » en lui envoyant une requête **GET /SCRAPE** contenant l'ID du fichier « **info_hash** ». Cette requête vise à obtenir des informations sur le fichier «**.torrent** » ainsi que sur tous les trackers hébergés.
- La figure ci-dessous montre le contenu de la requête **HTTP GET SCRAPE**.

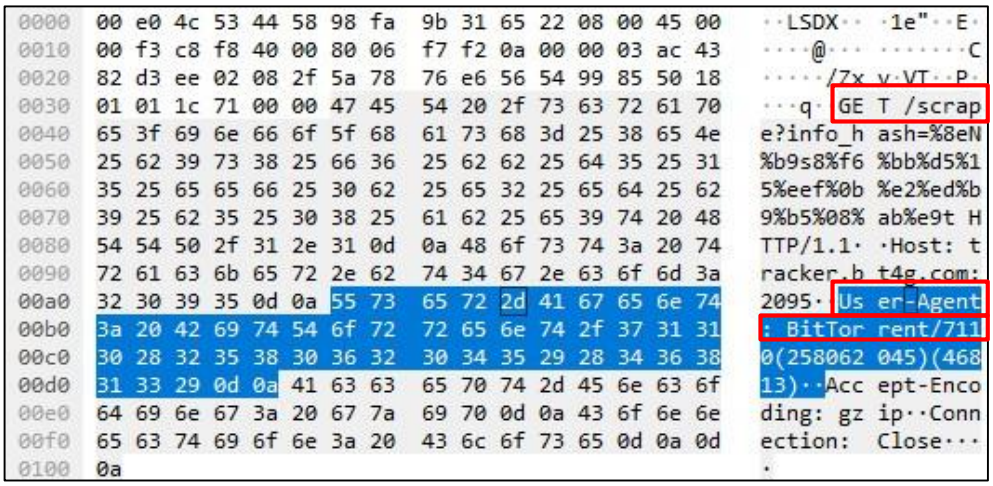


Figure 3.17 : Contenu de la requête "http GET SCRAPE"***.***

- Le client envoie des requêtes **HTTP GET ANNOUNCE** à tous les trackers disponibles. Ces requêtes contiennent le même identifiant « **info_hash** » ainsi que d'autres informations nécessaires pour établir la connexion avec eux.
- La figure suivante représente le contenu de la requête **HTTP GET ANNOUNCE**.

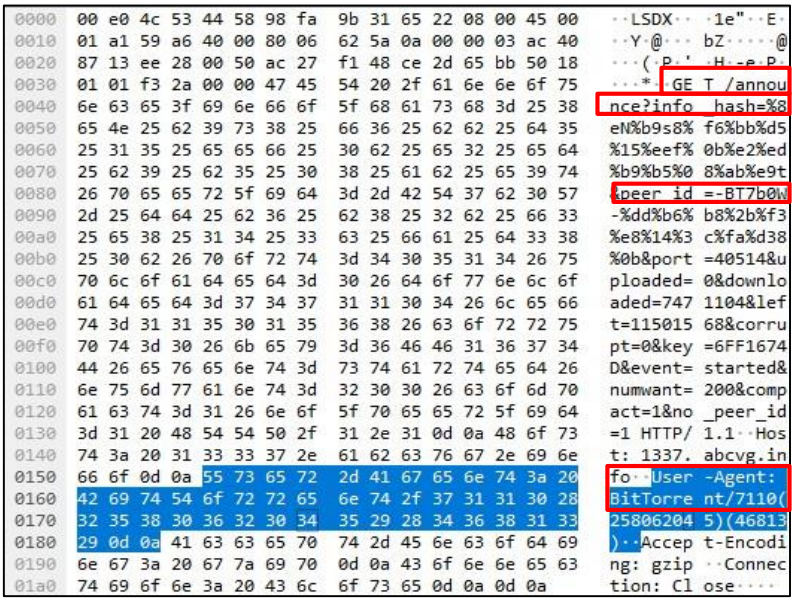


Figure 3.18 : Contenu de la requête "http GET ANNOUNCE".

- L'envoi de la réponse **HTTP 200 OK** par l'un de ces trackers « **172.67.130.211** » implique l'établissement de la connexion avec le client. Cette réponse permet ensuite au client d'établir des liens avec les pairs qui partagent le fichier. Voici le contenu de cette réponse.

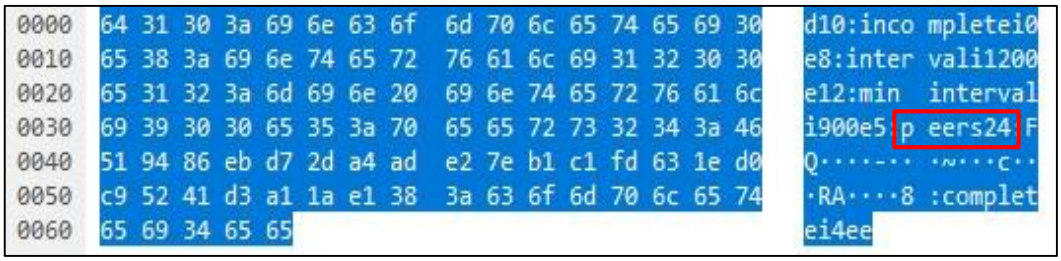


Figure 3.19 : Contenu de la réponse HTTP GET ANOUNCE.

Note : Seul le protocole HTTP est utilisé pour établir la connexion avec le tracker.

c Connexion aux pairs

Le protocole BitTorrent permet la connexion avec les Peers. La figure suivante représente les principales requêtes de ce protocole.

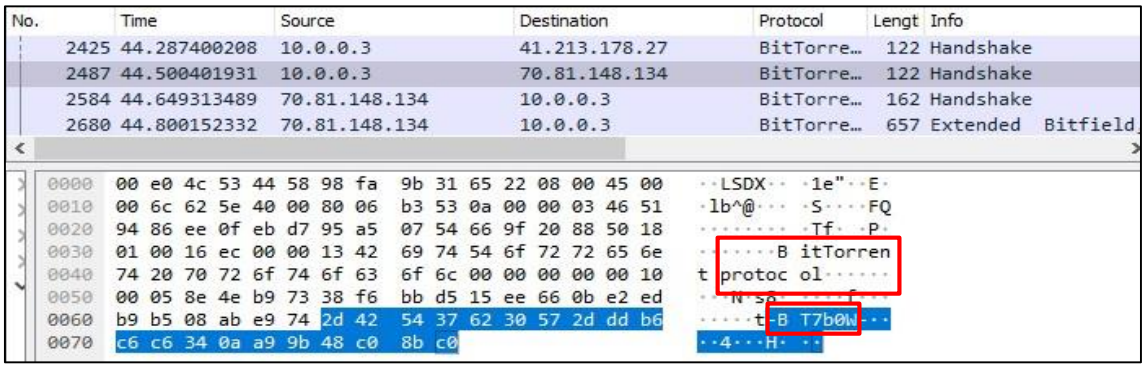


Figure 3.20 : BitTorrent HANDSHAKE.

- Le **BitTorrent Handshake** est le premier message envoyé par le client aux pairs. La figure (3.20) montre clairement la signature de « **BitTorrent Protocol** » dans les paquets capturés, suivie de la version de logiciel BitTorrent utilisée sous cette forme « **BT7b0w** » (Peer ID).

NOTE : Ce processus sera répété pour chaque Peer envoyé par le tracker.

- Lorsque le pair destinataire reçoit le message **handshake**, il répond avec le message **Extended BITFIELD** qui contient des informations, telles que la taille du fichier, le mode de fonctionnement (upload ou download), le logiciel utilisé ainsi que sa version.
- La réponse **Extended BITFIELD** est représentée dans la figure (3.21).

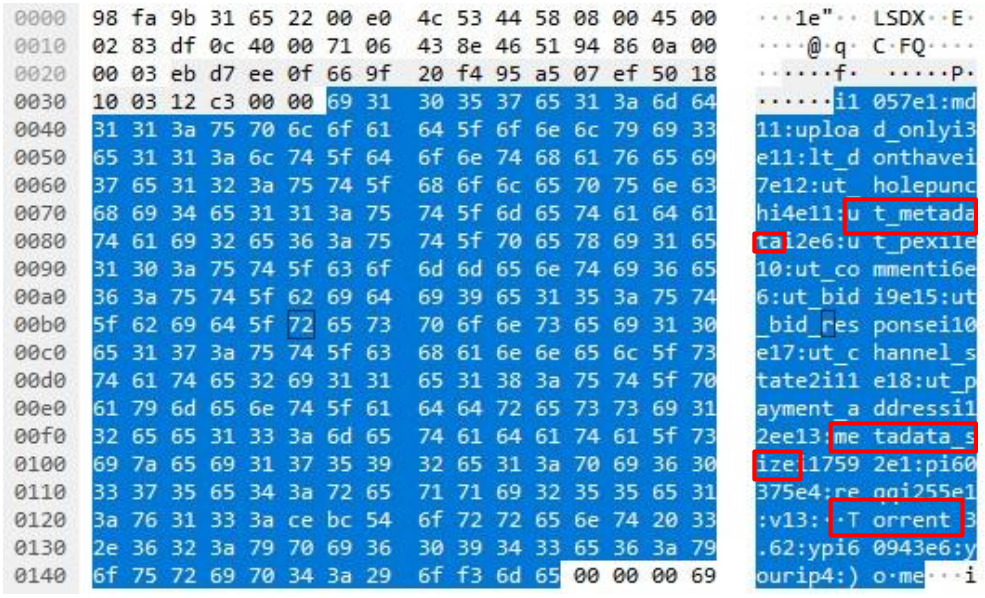


Figure 3.21 : Réponse BitTorrent "Extended BITFIELD"**.**

- Le client se connecte ensuite au SWARM de téléchargement et commence à négocier avec les autres Peers sur les pièces disponibles ou manquantes. Ces négociations sont cruciales pour le processus de téléchargement du fichier.

Note : le fonctionnement du BitTorrent dépend des deux protocoles « BitTorrent, HTTP ».

- BT-DHT est une extension du protocole BitTorrent qui permet de localiser rapidement les Peers de manière décentralisée, sans avoir besoin d'un tracker centralisé.

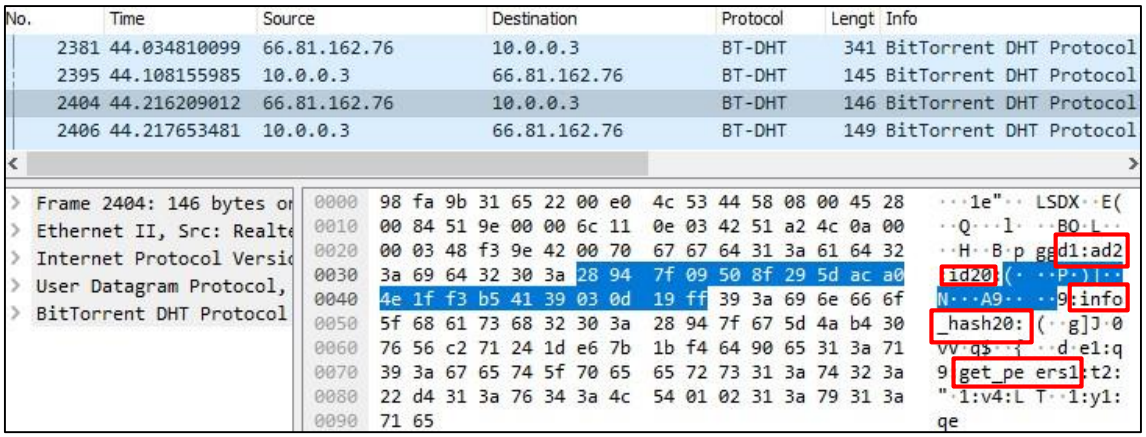


Figure 3.22 : DHT Ping Peer.

- Selon la figure (3.22), la chaîne **« d1 :ad2 :id20 »** représente le Ping, tandis que **« info_hash20 »** représente les informations sur le torrent et « **get_peers1** » représente la demande de pairs.

- Le protocole **BT-DHT** est utilisé aussi pour effectuer des Ping vers le tracker. La figure ci-dessous montre que la requête envoyée contient uniquement un indice lié au tracker « **ANNOUNCE** ».

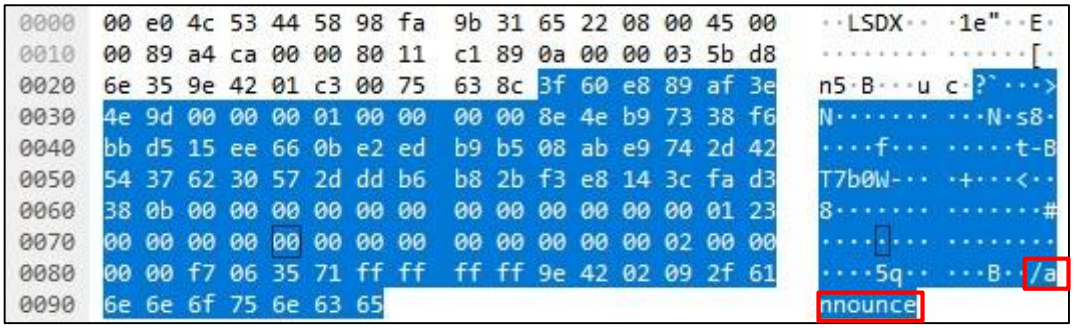
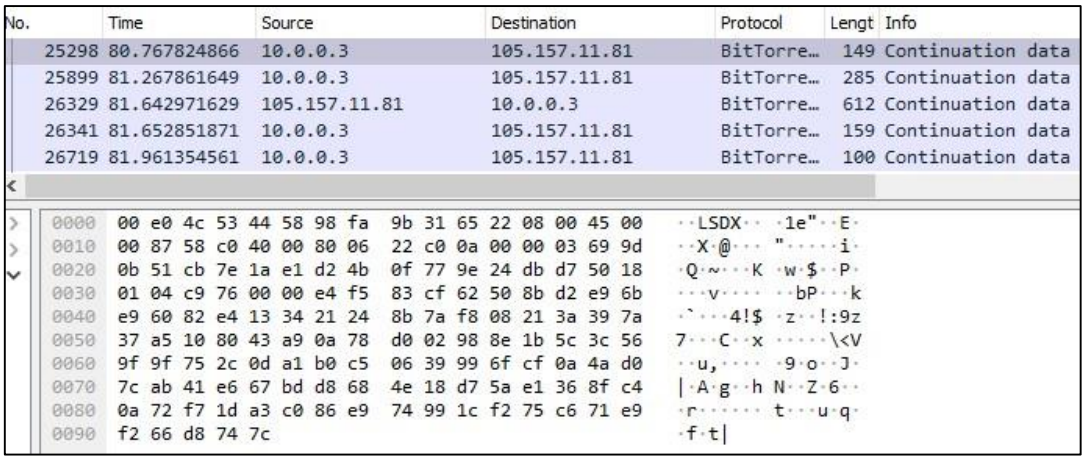


Figure 3.23 : DHT Ping tracker***.***

d Mode crypté : Forcé

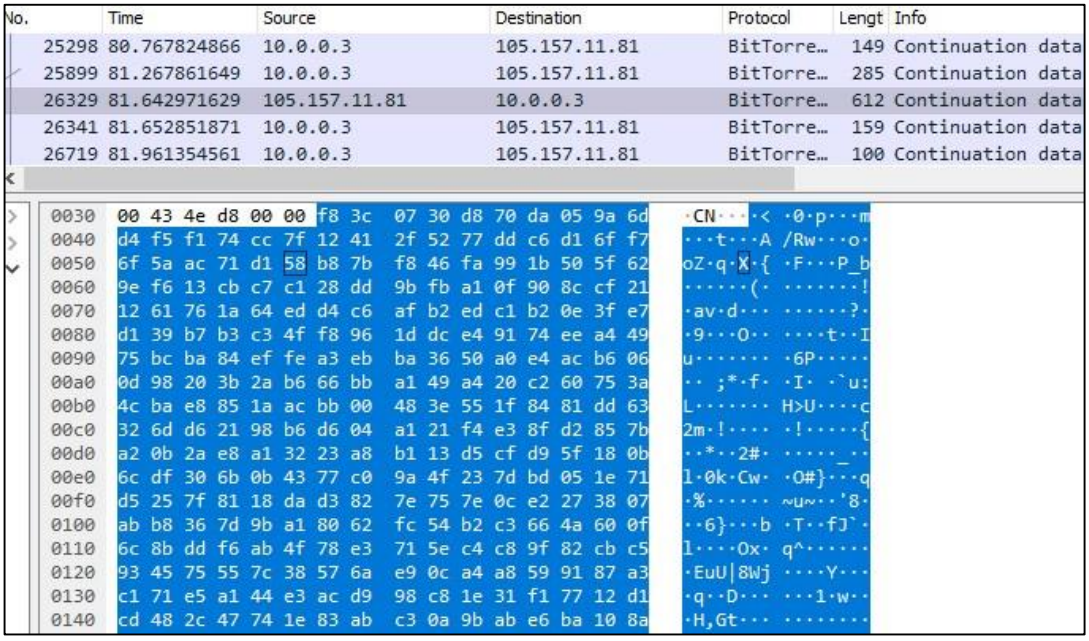
Jusqu'à présent, toutes les signatures du protocole BitTorrent ont été observées en mode clair, ce qui implique que le contenu des échanges est lisible et compréhensible. Dès que le mode crypté est activé, l'analyse du trafic devient comme le montre la figure (3.24).



No.	Time	Source	Destination	Protocol	Lengt	Info
25298	80.767824866	10.0.0.3	105.157.11.81	BitTorre…	149	Continuation data
25899	81.267861649	10.0.0.3	105.157.11.81	BitTorre…	285	Continuation data
26329	81.642971629	105.157.11.81	10.0.0.3	BitTorre…	612	Continuation data
26341	81.652851871	10.0.0.3	105.157.11.81	BitTorre…	159	Continuation data
26719	81.961354561	10.0.0.3	105.157.11.81	BitTorre…	100	Continuation data

```
0000  00 e0 4c 53 44 58 98 fa  9b 31 65 22 08 00 45 00   ··LSDX·· ·1e"··E·
0010  00 87 58 c0 40 00 80 06  22 c0 0a 00 00 03 69 9d   ··X·@··· "·····i·
0020  0b 51 cb 7e 1a e1 d2 4b  0f 77 9e 24 db d7 50 18   ·Q·~···K ·w·$··P·
0030  01 04 c9 76 00 00 e4 f5  83 cf 62 50 8b d2 e9 6b   ···v···· ··bP···k
0040  e9 60 82 e4 13 34 21 24  8b 7a f8 08 21 3a 39 7a   ·`···4!$ ·z··!:9z
0050  37 a5 10 80 43 a9 0a 78  d0 02 98 8e 1b 5c 3c 56   7···C··x ·····\<V
0060  9f 9f 75 2c 0d a1 b0 c5  06 39 99 6f cf 0a 4a d0   ··u,···· ·9·o··J·
0070  7c ab 41 e6 67 bd d8 68  4e 18 d7 5a e1 36 8f c4   |·A·g··h N··Z·6··
0080  0a 72 f7 1d a3 c0 86 e9  74 99 1c f2 75 c6 71 e9   ·r····· t···u·q·
0090  f2 66 d8 74 7c                                    ·f·t|
```

Figure 3.24 : BitTorrent HANDSHAKE Mode Crypté**.**



No.	Time	Source	Destination	Protocol	Lengt	Info
25298	80.767824866	10.0.0.3	105.157.11.81	BitTorre…	149	Continuation data
25899	81.267861649	10.0.0.3	105.157.11.81	BitTorre…	285	Continuation data
26329	81.642971629	105.157.11.81	10.0.0.3	BitTorre…	612	Continuation data
26341	81.652851871	10.0.0.3	105.157.11.81	BitTorre…	159	Continuation data
26719	81.961354561	10.0.0.3	105.157.11.81	BitTorre…	100	Continuation data

```
0030  00 43 4e d8 00 00 f8 3c  07 30 d8 70 da 05 9a 6d   ·CN····< ·0·p···m
0040  d4 f5 f1 74 cc 7f 12 41  2f 52 77 dd c6 d1 6f f7   ···t···A /Rw···o·
0050  6f 5a ac 71 d1 58 b8 7b  f8 46 fa 99 1b 50 5f 62   oZ·q·X·{ ·F···P_b
0060  9e f6 13 cb c7 c1 28 dd  9b fb a1 0f 90 8c cf 21   ·····(· ·······!
0070  12 61 76 1a 64 ed d4 c6  af b2 ed c1 b2 0e 3f e7   ·av·d··· ·····?·
0080  d1 39 b7 b3 c3 4f f8 96  1d dc e4 91 74 ee a4 49   ·9···O·· ····t··I
0090  75 bc ba 84 ef fe a3 eb  ba 36 50 a0 e4 ac b6 06   u····· ·6P·····
00a0  0d 98 20 3b 2a b6 66 bb  a1 49 a4 20 c2 60 75 3a   ·· ;*·f· ·I· ·`u:
00b0  4c ba e8 85 1a ac bb 00  48 3e 55 1f 84 81 dd 63   L····· H>U····c
00c0  32 6d d6 21 98 b6 d6 04  a1 21 f4 e3 8f d2 85 7b   2m·!···· ·!·····{
00d0  a2 0b 2a e8 a1 32 23 a8  b1 13 d5 cf d9 5f 18 0b   ··*··2#· ·····_··
00e0  6c df 30 6b 0b 43 77 c0  9a 4f 23 7d bd 05 1e 71   l·0k·Cw· ·O#}···q
00f0  d5 25 7f 81 18 da d3 82  7e 75 7e 0c e2 27 38 07   ·%····· ~u~··'8·
0100  ab b8 36 7d 9b a1 80 62  fc 54 b2 c3 66 4a 60 0f   ··6}···b ·T··fJ`·
0110  6c 8b dd f6 ab 4f 78 e3  71 5e c4 c8 9f 82 cb c5   l····Ox· q^······
0120  93 45 75 55 7c 38 57 6a  e9 0c a4 a8 59 91 87 a3   ·EuU|8Wj ····Y···
0130  c1 71 e5 a1 44 e3 ac d9  98 c8 1e 31 f1 77 12 d1   ·q··D··· ···1·w··
0140  cd 48 2c 47 74 1e 83 ab  c3 0a 9b ab e6 ba 10 8a   ·H,Gt··· ········
```

Figure 3.25 : Réponse de BitTorrent Extended BITIFIELD Mode Crypté.

- Les échanges basés sur le protocole BitTorrent sont chiffrés, mais les deux protocoles HTTP et BT-DHT mentionnés précédemment continuent de fonctionner en clair, même si le chiffrement BitTorrent est activé.

Solution proposée : On va se concentrer sur le fonctionnement du protocole HTTP et BT-DHT pour détecter les connexions torrent cryptées.

3.8.2 IPFS Desktop

a Etablissement de connexion

Voici un extrait des paquets reçus lors d'une connexion à base d'un client IPFS (voir figure 3.26).

Protocol	Lengt	Info
HTTP	356	POST /api/v0/id HTTP/1.1
HTTP/JSON	49	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
HTTP	444	POST /api/v0/stats/bw HTTP/1.1
HTTP/JSON	49	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
HTTP	447	POST /api/v0/config/show HTTP/1.1
HTTP/JSON	49	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
HTTP	476	POST /api/v0/swarm/peers?verbose=true&timeout=10000ms HTTP/1.1
HTTP/JSON	49	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
HTTP	454	POST /api/v0/files/stat?arg=%2F HTTP/1.1
HTTP/JSON	49	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
HTTP	489	POST /api/v0/ls?arg=QmWe7gRFzCebV656aCHHMjPMzk8ZrHEGNdXD9EptUV8
HTTP/JSON	49	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)

Figure 3.26 : Paquets IPFS capturés lors d'établissement de connexion.

Lors du lancement de l'application IPFS Desktop, on constate l'envoi de plusieurs paquets. Ces derniers sont la première étape du fonctionnement IPFS, qui nous permettra ensuite le stockage et le partage des fichiers. Si on veut identifier l'utilisation du protocole IPFS, il faudra étudier et analyser ces requêtes en détail.

- Selon la Figure (3.26), l'établissement de la connexion avec le réseau IPFS commence par la requête **« POST /api/v0/id »**. Cette requête est utilisée pour obtenir des détails sur le nœud IPFS, la requête est représentée dans la figure ci-dessous :

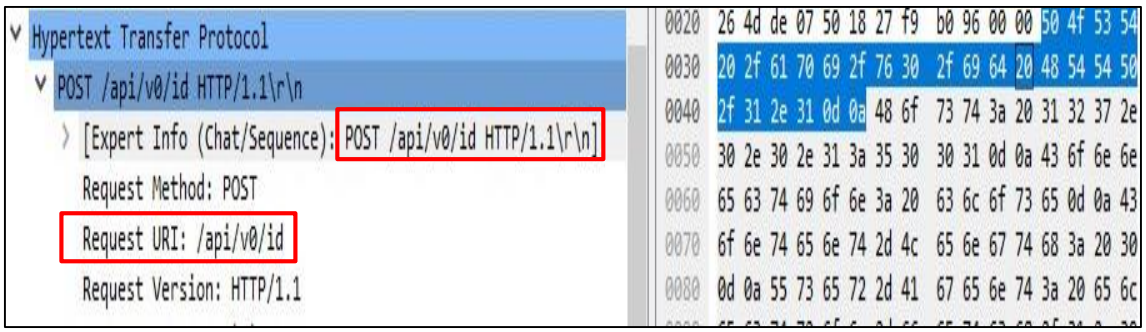


Figure 3.27 : Capture du paquet "POST /api/v0/id".

- La réponse à cette requête contient toutes les informations essentielles permettant de comprendre l'état du nœud IPFS. Parmi ces informations, on trouve l'identifiant du nœud IPFS (Peer-ID), la version du logiciel IPFS utilisé (AgentVersion), la clé publique associée (PublicKey) ...

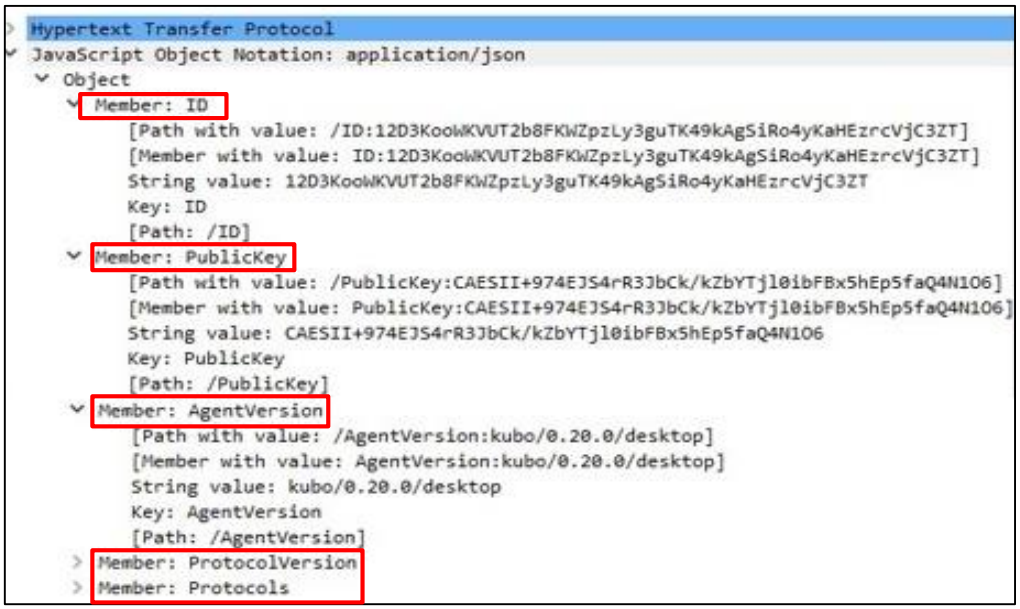


Figure 3.28 : La réponse de la requête "POST /api/v0/id".

❖ Le client envoie la requête « **POST /api/v0/stats/bw** » afin d'avoir des informations sur la consommation de bande passante de son nœud IPFS. Cette requête lui permet de collecter des statistiques sur la quantité de données téléchargées et téléversées par son nœud.

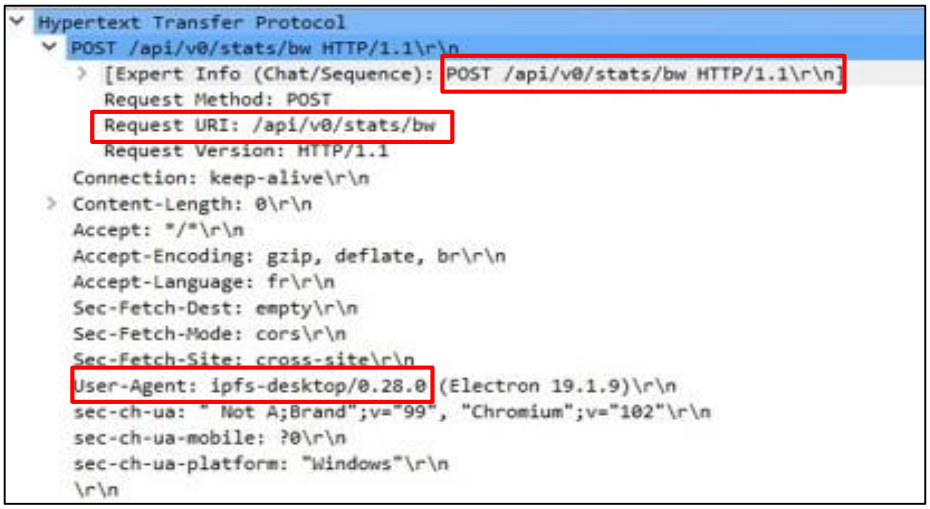


Figure 3.29 : Capture du paquet " POST /api/v0/stats/bw ".

❖ La figure (3.30) montre la réponse à la requête « **POST /api/v0/stats/bw** », qui inclut les informations suivantes : le nombre total d'octets téléchargés et téléversés depuis le nœud IPFS (TotalIn, TotalOut), ainsi que le taux actuel de téléchargement et de téléversement en octets par seconde (RateIn, RateOut).

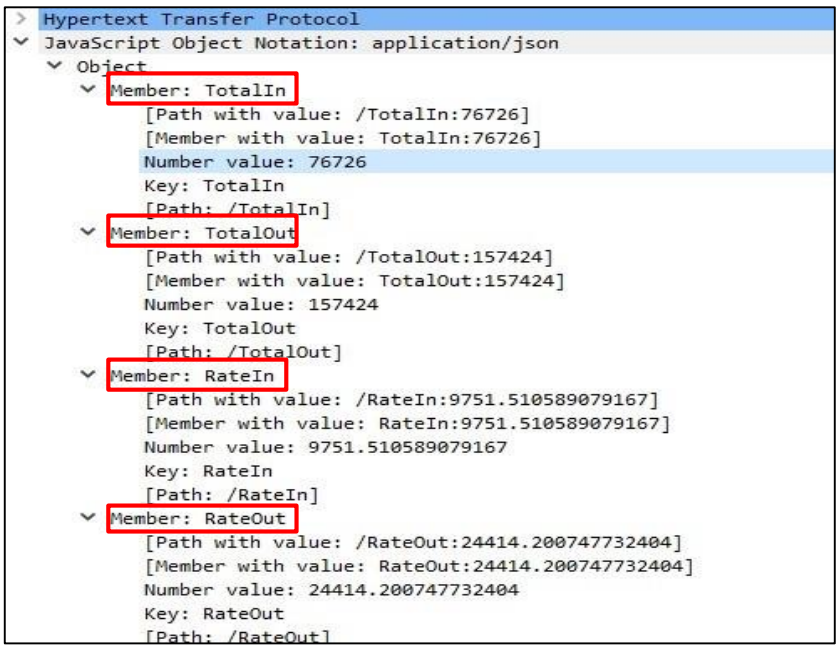


Figure 3.30 : La réponse de la requête " POST /api/v0/stats/bw ".

❖ Pour initialiser le fichier de configuration IPFS (IPFS-config), le client envoie la requête « **POST /api/v0/config/show** ».

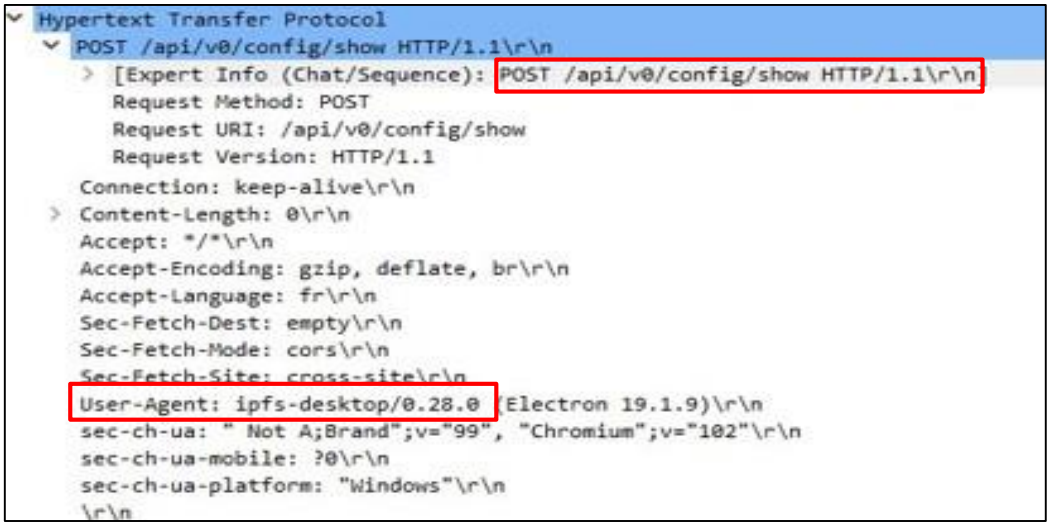


Figure 3.31 : Capture du paquet " POST /api/v0/config/show ".

❖ Par la suite, le serveur répond avec tous les paramètres concernant le fichier de configuration.

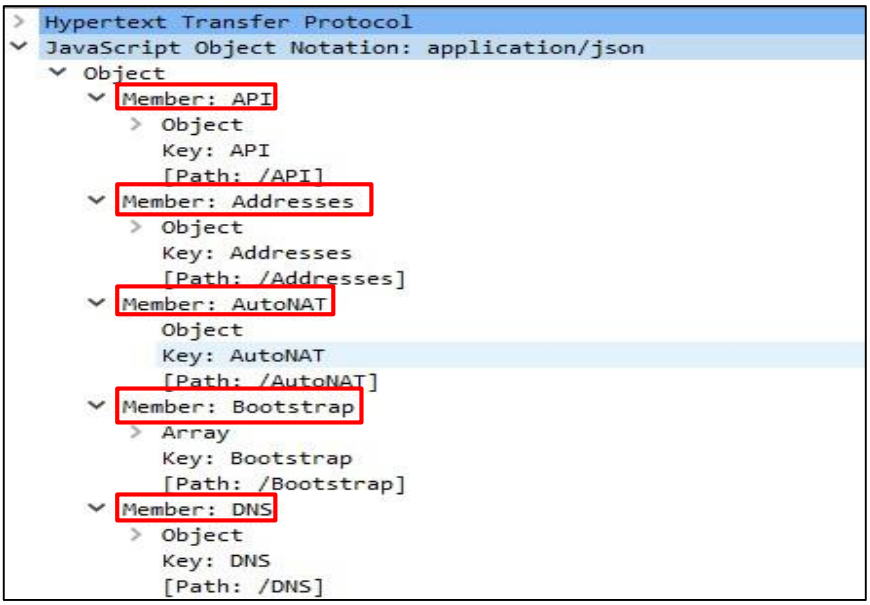


Figure 3.32 : La réponse de la requête " POST /api/v0/config/show ".

❖ Afin de se connecter aux pairs présents sur le réseau IPFS, le client envoie la requête *« **POST /api/v0/swarm/peer?verbose** »*, qui est représentée comme suit :

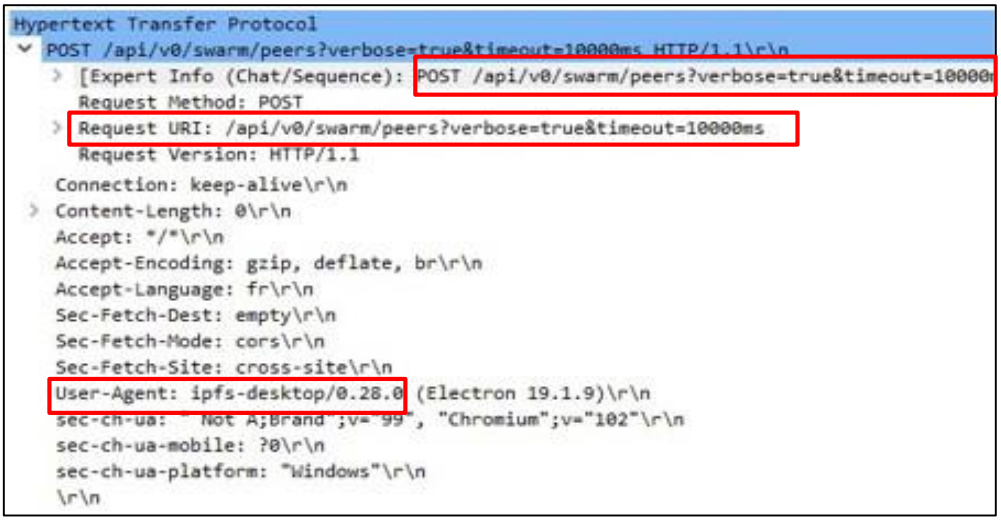


Figure 3.33 : Capture du paquet " POST /api/v0/swarm/peer?verbose ".

❖ La réponse permet au client d'établir des connexions avec tous les Peers disponibles dans le réseau.

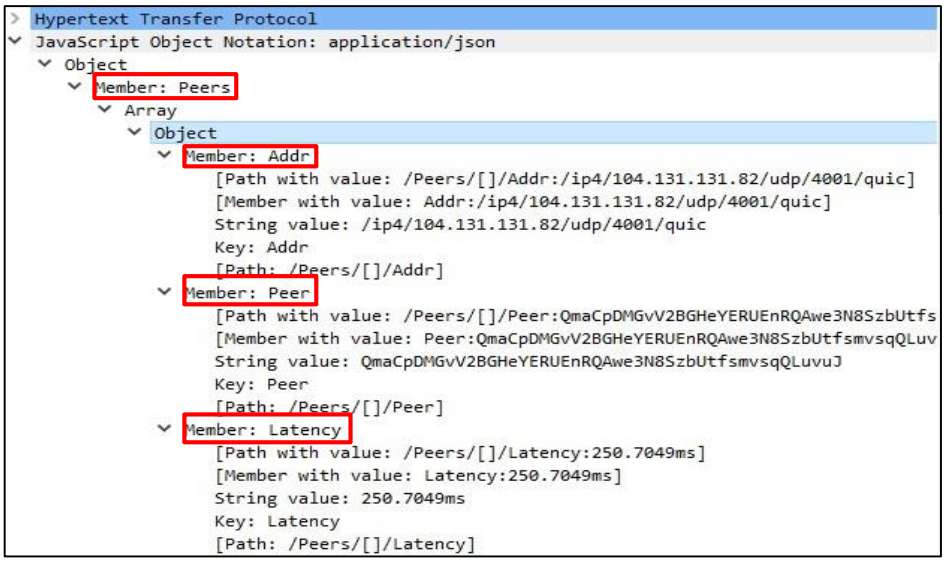


Figure 3.34 : La réponse de la requête " POST /api/v0/swarm/peer?verbose ".

Note : la mise à jour des Peers se fait de manière périodique.

❖ Le client utilise la requête *« **POST /api/v0/files/stat?arg** »* afin de demander des informations concernant le répertoire racine qui englobe tous les fichiers stockés au niveau de l'application IPFS Desktop.

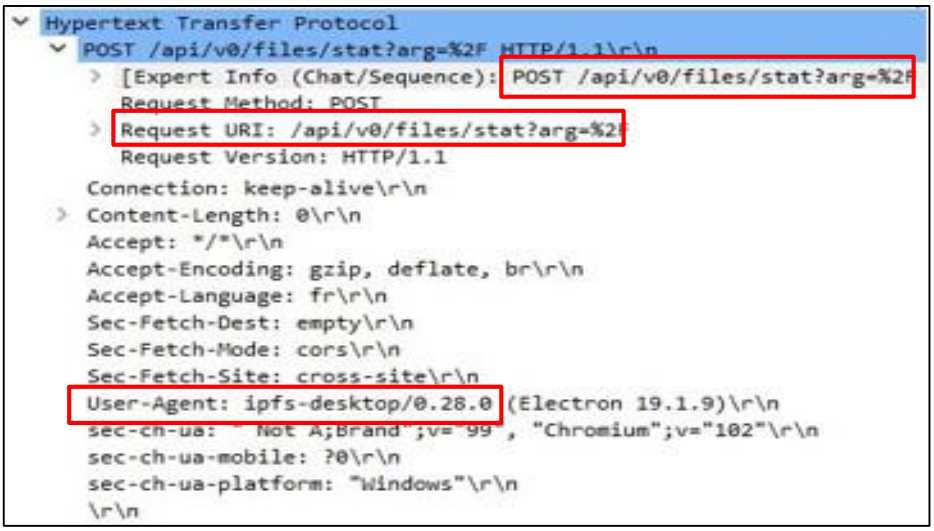


Figure 3.35 : Capture du paquet " POST /api/v0/files/stat?arg ".

❖ La réponse permet au client de récupérer des détails sur le répertoire principal, tels que sa taille, le nombre de fichiers stockés, son empreinte de hachage, et d'autres informations.

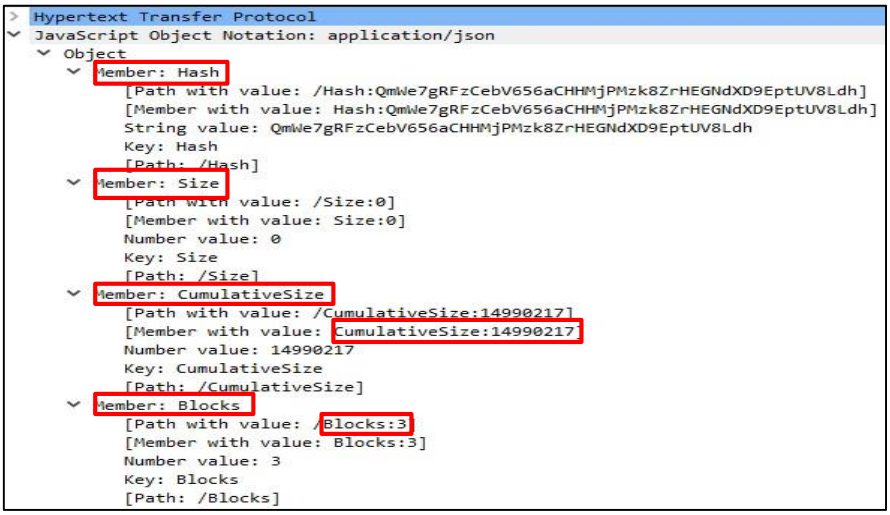


Figure 3.36 : La réponse de la requête " POST /api/v0/files/stat?arg ".

❖ Avec la requête *« **POST /api/v0/ls?arg** »*, le client peut obtenir des informations spécifiques sur chaque fichier contenu dans le répertoire racine de l'application IPFS Desktop.

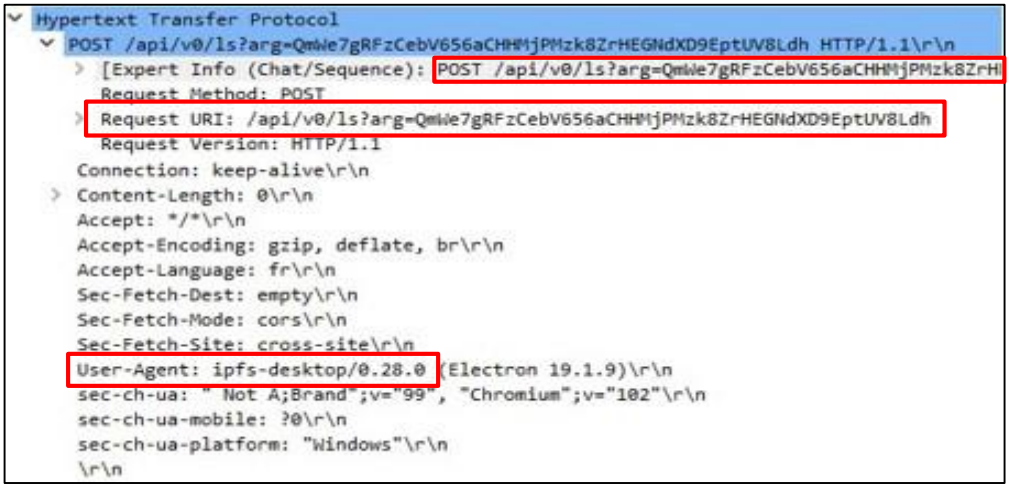


Figure 3.37 : Capture du paquet " POST /api/v0/ls?arg ".

❖ La réponse à la requête « POST /api/v0/ls?arg » permet au client de connaître en détail les caractéristiques de chaque fichier, tels que le nom du fichier, sa taille et son hachage (CID).

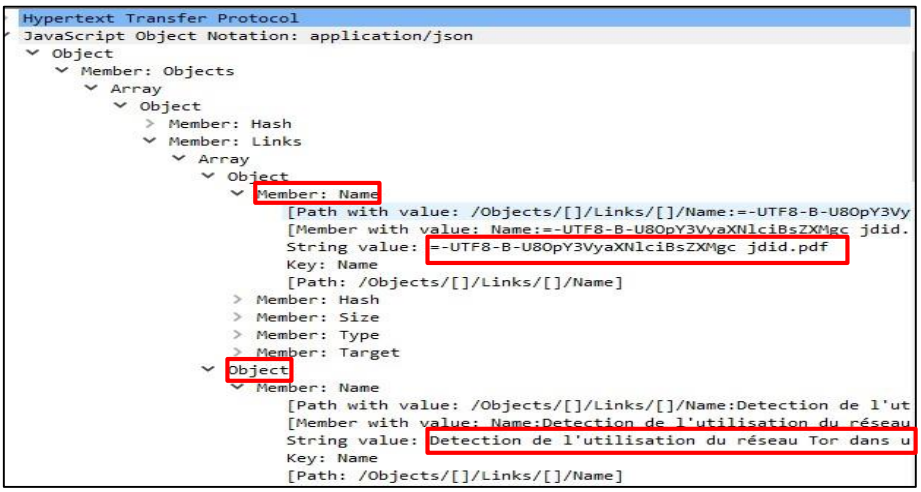


Figure 3.38 : La réponse de la requête " POST /api/v0/ls?arg ".

b Ajouter un fichier

Le téléversement des fichiers locaux vers le nœud IPFS pour les stocker et les distribuer dans le réseau IPFS est basé sur le protocole HTTP. La figure (3.39) montre les requêtes échangées.

Figure 3.39 : Capture du paquet " POST /api/v0/add?stream-channels ".

- Cette demande est envoyée lorsque l'utilisateur souhaite ajouter un fichier.

D'après la figure (3.40), on constate que la réponse est dédiée pour renseigner le client sur les paramètres de fichier ajoutés, tels que le nom, la taille du fichier et son hachage (CID).

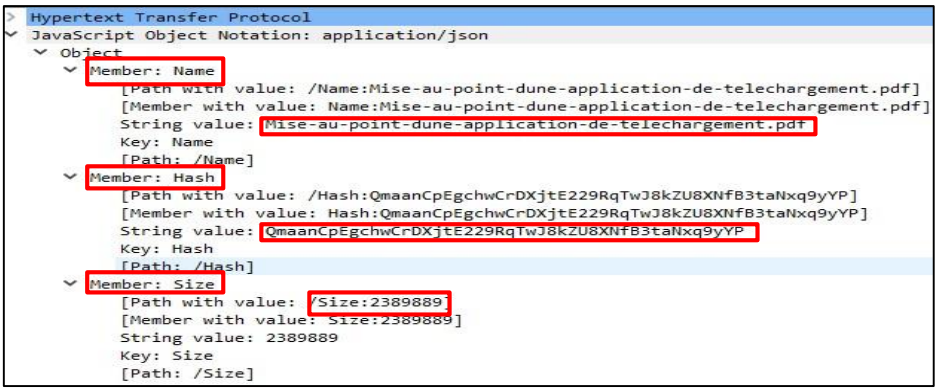


Figure 3.40 : La réponse de la requête " POST /api/v0/add?stream-channels ".

c Accomplissement d'une recherche d'un fichier

Le client peut effectuer une recherche au niveau d'IPFS Desktop en utilisant le hachage du fichier (CID). Les requêtes envoyées sont représentées comme suit.

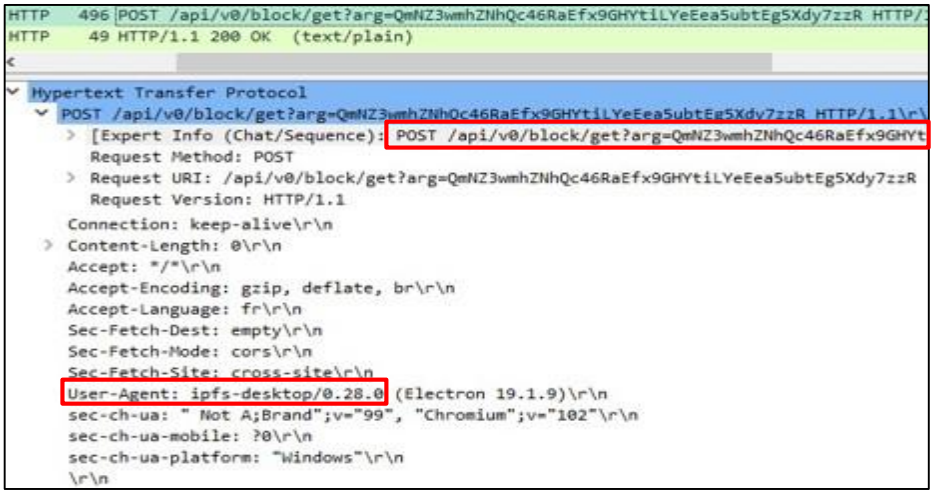


Figure 3.41 : Capture du paquet " POST /api/v0/block/get?arg ".

- Le client envoie la requête *«* **POST /api/v0/block/get?arg** *»* pour récupérer les blocs de fichier "chunks". La réponse permet au client de recevoir le contenu du fichier désiré.

d Consulter le contenu d'un fichier

Après avoir stocké un fichier ou récupéré le CID d'un fichier sur IPFS Desktop, le client a la possibilité de visualiser le contenu de ce dernier, soit directement dans l'application, soit via un navigateur Web.

❖ Le client utilise la requête « **POST /api/v0/cat?arg** » pour afficher le contenu dans IPFS Desktop.

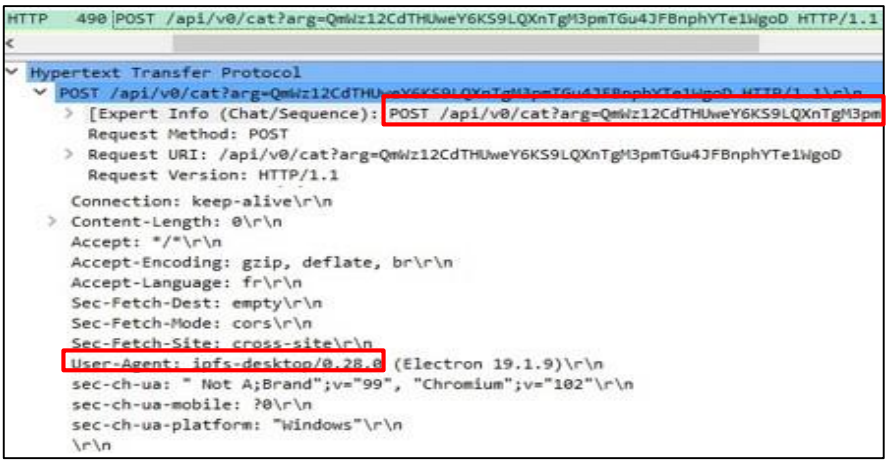


Figure 3.42 : Capture du paquet " POST /api/v0/cat?arg ".

❖ Le client utilise la requête « **GET/ipfs** » pour afficher le contenu dans un navigateur Web.

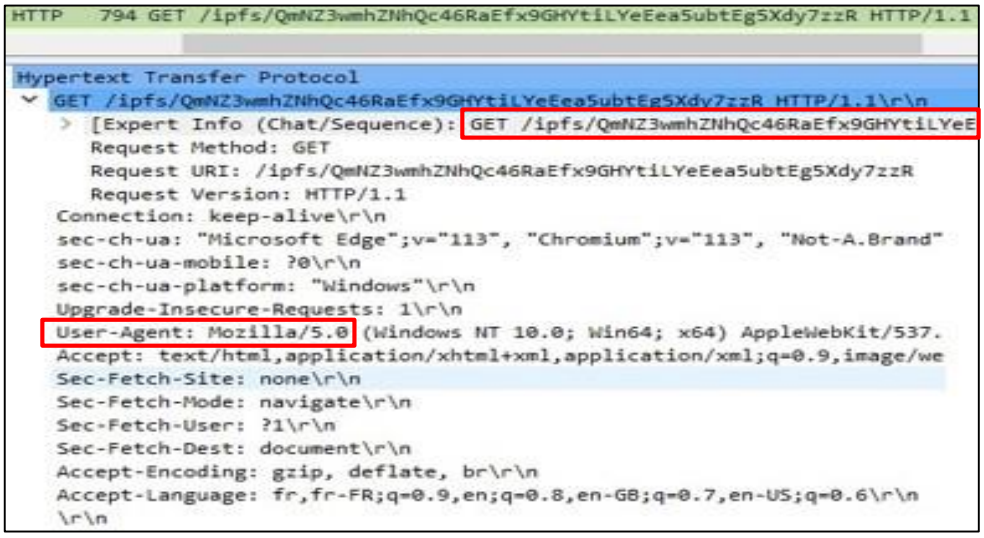


Figure 3.43 : Capture du paquet " GET /ipfs ".

3.9 Résultat d'analyse : Signatures d'applications

3.9.1 BitTorrent

À travers notre analyse, on déduit le tableau des signatures du protocole BitTorrent représenté ci-dessous :

N	Nom	Contenu	Transport protocole
1	GET SCRAPE	Get /scrape?info_hash	TCP
		User-Agent: BitTorrent	
2	GET ANNOUNCE	Get /announce?info_hash	TCP
		User-Agent : BitTorrent	
3	HANDSHAKE	BitTorrent protocol	TCP
4	EXTENDED BITFIELD	ut_metadata	TCP
		metadata_size	
5	DHT Ping-Peer	d1:ad2:id20	UDP
		info_hash20	
		get_peers1	
6	DHT Ping tracker	/announce	UDP

Tableau 3.2 : Empreintes numériques du protocole BitTorrent**.**

3.9.2 IPFS Desktop

Les résultats d'analyse des signatures du protocole IPFS sont résumés dans le tableau suivant :

N	Nom	Contenu	Transport protocole
1	Peer ID	Post /api/v0/id	TCP
2	Bande passante	Post /api/v0/stats/bw User-Agent : ipfs-desktop	TCP
3	Config_IPFS	Post /api/v0/config/show User-Agent : ipfs-desktop	TCP
4	Swarm	Post /api/v0/swarm/peers?verbose User-Agent : ipfs-desktop	TCP
5	Répertoire racine	Post /api/v0/files/stat?arg User-Agent : ipfs-desktop	TCP
6	Fichier stocké	Post /api/v0/ls?arg User-Agent : ipfs-desktop	TCP
7	Ajouter un fichier	Post /api/v0/add?stream-channels User-Agent : ipfs-desktop	TCP
8	Récupérer un fichier	Post / api/v0/block/get?arg User-Agent : ipfs-desktop	TCP
9	Consulter le contenu d'un fichier	Post /api/v0/cat?arg User-Agent : ipfs-desktop Get /ipfs	TCP

Tableau 3.3 : Empreintes numériques du protocole IPFS**.**

3.10 Conclusion

Au cours de ce chapitre, on a analysé le trafic Peer to Peer provenant des applications BitTorrent et IPFS. L'étude de cette analyse nous a permis d'extraire des empreintes numériques qui permettent d'identifier l'utilisation des protocoles BitTorrent et IPFS, afin de les implémenter dans un système de détection et de prévention d'intrusion "IDS/IPS".

Dans le prochain chapitre, on va mettre en œuvre ces empreintes dans l'IDS/IPS "Suricata" et tester leurs validités.

Chapitre 4

Implémentation des empreintes

Chapitre 4 Implémentation des empreintes

4.1 Introduction

Dans ce chapitre on va faire la détection en exploitant l'analyse et l'extraction des signatures faites dans le chapitre précédent. Tout d'abord, on va commencer par une étude théorique sur IDS/IPS suricata, ensuite, on va créer des règles en utilisant les signatures extraites qui vont être utilisées comme un paramètre de détection de trafic BitTorrent et IPFS.

4.2 Système de détection d'intrusion IDS

Un système de détection d'intrusion ou ***«*IDS : intrusion Detection system*»*** est un équipement matériel ou bien logiciel permettant de surveiller l'activité d'un réseau ou d'un hôte donné, afin de détecter toute activité malveillante [26].

Certains termes sont souvent employés quand on parle d'IDS [26]:

- **Faux positif :** une alerte provenant d'un IDS, mais qui ne correspond pas à une attaque réelle.
- **Faux négatif :** une intrusion réelle qui n'a pas été détectée par l'IDS.

Il existe deux catégories principales d'IDS [26].

- **Les NIDS** (Network Based Intrusion Detection System) : ce sont des IDS utilisés pour protéger un réseau. Ils comportent généralement une sonde (machine par exemple) qui écoute et surveille en temps réel tout le trafic réseau, puis analyse et génère des alertes s'il détecte des intrusions ou des paquets semblent dangereux.
- **Les HIDS** (Host Based Intrusion Detection System) : Ces IDS sont implémentés directement sur les machines du réseau. Ils analysent le comportement et le

fonctionnement de la machine afin de détecter une activité inhabituelle. En cas d'intrusion, une alerte est envoyée à l'utilisateur du poste ou à l'administrateur réseau.

4.3 Système de prévention d'intrusion

Les systèmes de prévention des intrusions contrôlent l'accès à un réseau informatique et le protègent contre les abus et les attaques. Ces systèmes sont conçus pour surveiller les données d'intrusion et prendre les mesures nécessaires pour éviter qu'une attaque ne se déclenche [27].

Il existe deux approches de prévention [27]:

- **Approche basée sur les signatures :** Cette approche utilise des signatures prédéfinies de menaces réseau bien connues. Lorsqu'une attaque correspondant à l'une de ces signatures ou à l'un de ces modèles est lancée, le système prend les mesures nécessaires.
- **Approche basée sur les anomalies :** Permet de surveiller tout comportement anormal ou inattendu sur le réseau. Si une anomalie est détectée, le système bloque immédiatement l'accès à l'hôte cible.

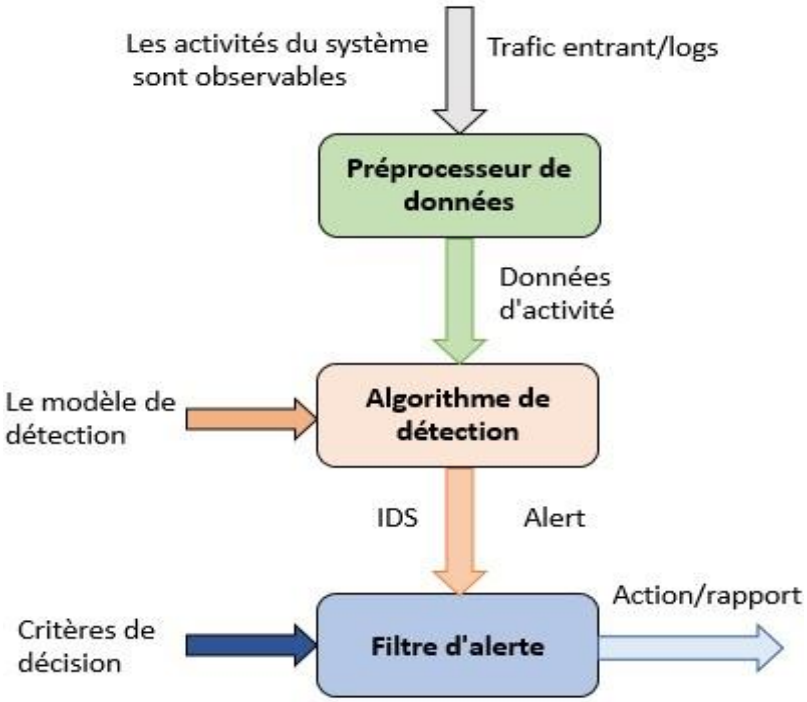


Figure 4.1 : Le fonctionnement d'un IDS/IPS.

4.4 Suricata

Suricata est un système de détection d'intrusion et de prévention d'intrusion open source conçu pour détecter et prévenir des cybermenaces en temps réel. Il peut être configuré pour alerter les administrateurs et prendre des mesures pour atténuer les menaces identifiées.

Format des règles de Suricata

Voici un exemple de règle suricata :

```
alert icmp any any -> any any (msg:"icmp
packet found"; sid: 10004861; rev:1;)
```

Figure 4.2 : Exemple de règle suricata.

Les règles Suricata sont divisées en deux sections logiques qui sont [28]:

1- **L'entête de la règle :** elle contient l'action de la règle, le protocole, des adresses IP, des ports et de la direction de la règle.

- **L'action de règle :** qui détermine ce qui se passe lorsque le paquet correspond aux critères de la règle. Il existe 4 types d'actions :
  - **Alerte :** Suricata générera une alerte et l'enregistrera pour une analyse plus approfondie.
  - **Passer :** Suricata arrêtera d'analyser le paquet et l'autorisera, sans générer d'alerte.
  - **Rejeter** : Lorsque Suricata exécute le mode IPS, Suricata supprimera le paquet correspondant.
  - **Drop :** Lorsque vous travaillez en mode IPS, Suricata arrête immédiatement de traiter le paquet et génère une alerte.
- **Le protocole :** il y a quatre protocoles utilisés pour la transmission de données : IP, TCP, UDP et ICMP.
- **Les adresses IP :** Les adresses IP source ou destination et on peut utiliser le mot clé "ANY" Pour définir n'importe quelle adresse.

- **Les numéros de ports :** Les ports source ou destination et on peut utiliser le mot clé "ANY" Pour définir n'importe quel port.
- **L'opérateur de direction :** L'opérateur de direction "->" indique l'orientation du flux de paquet.

2- **Les options de la règle :** déterminant les spécificités de la règle.

- **Msg :** affiche un message dans les alertes et journalise les paquets.
- **Sid :** (signature identifier), il donne à chaque signature son propre ID.
- **Rev :** il représente la version de la signature.

4.5 Création / implémentation des règles dans Suricata :

Dans cette partie, on va aborder la création des règles spécifiquement conçues pour détecter le trafic des réseaux P2P, ainsi que leur intégration dans le système Suricata.

4.5.1 Création des règles suricata

En se basant sur les signatures numériques extraites précédemment, on peut créer des règles permettant de détecter (alerte) et bloquer (drop) l'utilisation des protocoles IPFS et BitTorrent.

a Les règles de détection

La structure de création des règles de détection de ces deux protocoles repose entièrement sur le protocole de transport (UDP/TCP). Tel mentionné précédemment, les deux protocoles sont identifiés par le contenu du paquet.

❖ **BitTorrent**

Règle N° 01 : La première règle identifie l'accès au site "cpasbien". Cette règle va lancer une alerte avec le message "attention utilisation BitTorrent : cpasbien".

```
alert tcp any any -> any any (msg: "Attention utilisation BitTorrent: cpasbien ";
content:"GET /scrape?info_hash"; content:"User-Agent: BitTorrent"; sid:2; rev:1; )
```

- La deuxième et la troisième règle détectent les connexions avec les trackers "get scrape", "get announce".

Règle N° 02 : Cette règle va lancer une alerte avec le message "attention utilisation BitTorrent : get scrape".

```
alert tcp any any -> any any (msg: "Attention utilisation BitTorrent: get scrape "; content:"GET /scrape?info_hash"; content:"User-Agent: BitTorrent"; sid:2; rev:1; )
```

Règle N° 03 : Cette règle va lancer une alerte avec le message "attention utilisation BitTorrent : get announce".

```
alert tcp any any -> any any (msg: "Attention utilisation BitTorrent: get announce"; content:"GET /announce?info_hash"; content:"User-Agent: BitTorrent"; sid:3; rev:1;
```

Règle N° 04 : La quatrième règle identifie la requête handshake. Cette règle va lancer une alerte avec le message "Attention utilisation BitTorrent : handshake".

```
alert tcp any any -> any any (msg: "Attention utilisation BitTorrent: handshake "; content:"BitTorrent protocol"; sid:4; rev:1; )
```

Règle N° 05 : La cinquième règle identifie la réponse Extended Bitfield. Cette règle va lancer une alerte avec le message "Attention utilisation BitTorrent : Extended Bitfield".

```
alert tcp any any -> any any (msg: "Attention utilisation BitTorrent: Extended Bitfield "; content:"ut_metadata"; content:"metadata_size"; sid:5; rev:1; )
```

Règle N° 06 : La sixième règle identifie la connexion avec les trackers via DHT. Cette règle va lancer une alerte avec le message "attention utilisation BitTorrent : DHT Ping tracker".

```
alert udp any any -> any any (msg: "Attention utilisation BitTorrent: DHT Ping tracker"; content:"/announce"; sid:6; rev:1; )
```

Règle N° 07 : La septième règle identifie la connexion avec les Peers via DHT. Cette règle va lancer une alerte avec le message "attention utilisation BitTorrent : DHT Ping Peer".

```
alert udp any any -> any any (msg: "Attention utilisation BitTorrent: DHT Ping Peer"; content:"d1:ad2:id20"; content:"info_hash20"; content:"get_peers1"; sid:7; rev:1; )
```

❖ **IPFS Desktop**

Règle N° 01 : La première règle identifie le Peer ID. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : Peer ID ".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: Peer_ID";
content:"Post /api/v0/id"; sid:1; rev:2; )
```

Règle N° 02 : La deuxième règle identifie la bande passante. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : Bande passante".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: Bande Passante";
content:"Post /api/v0/stats/bw"; content:"User-Agent: ipfs-desktop"; sid:2; rev:2; )
```

Règle N° 03 : La troisième règle détecte le fichier de configuration. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : config IPFS".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: config_IPFS";
content:"Post /api/v0/config/show"; content:"User-Agent: ipfs-desktop"; sid:3; rev:2; )
```

Règle N° 04 : La quatrième règle identifie les connexions avec les Peers. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : swarm".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: swarm"; content:"Post
/api/v0/swarm/peers?verbose"; content:"User-Agent: ipfs-desktop"; sid:4; rev:2; )
```

Règle N° 05 : La cinquième règle identifie le répertoire racine des fichiers. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : répertoire racine".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: repertoire racine";
content:"Post /api/v0/Files/stat?arg"; content:"User-Agent: ipfs-desktop"; sid:5;
```

Règle N° 06 : La sixième règle identifie les fichiers stockés dans l'IPFS. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : fichier stocké".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: fichier stocké";
content:"Post /api/v0/ls?arg"; content:"User-Agent: ipfs-desktop"; sid:6; rev:2; )
```

Règle N° 7 : : La septième règle détecte l'ajout des fichiers dans l'IPFS. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : ajouter un fichier".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: ajouter un fichier"; content:"Post /api/v0/add?stream-channels"; content:"User-Agent: ipfs-desktop"; sid:10; rev:2; )
```

Règle N° 08 : La huitième règle détecte la récupération des fichiers. Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : récupérer un fichier".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: récupérer un fichier"; content:"Post /api/block/get?arg"; content:"User-Agent: ipfs-desktop"; sid:8; rev:2; )
```

- La neuvième et la dixième règle détectent la visualisation des fichiers via IPFS Desktop ou via le navigateur.

Règle N° 09 : Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : consulter un fichier".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: consulter un fichier"; content:"Post /api/v0/cat?arg"; content:"User-Agent: ipfs-desktop"; sid:7; rev:2; )
```

Règle N° 10 : Cette règle lancera une alerte en indiquant le message "Attention utilisation IPFS : afficher un fichier".

```
alert tcp any any -> any any (msg: "Attention utilisation IPFS: afficher un fichier"; content:"GET /ipfs"; sid:9; rev:2; )
```

b Les règles de blocage

Pour empêcher la transmission des paquets associés aux protocoles IPFS et BitTorrent, on va modifier les règles d'alerte existantes en remplaçant l'action « alert » par « drop ».

- Exemple pour le protocole BitTorrent.

```
drop tcp any any -> any any (msg: "Attention utilisation BitTorrent: get announce"; content:"GET /announce?info_hash"; content:"User-Agent: BitTorrent"; sid:8; rev:1; )
```

- Exemple pour le protocole IPFS.

```
drop tcp any any -> any any (msg: "Attention utilisation IPFS: Peer_ID";
content:"Post /api/v0/id"; sid:11; rev:2; )
```

❖ **Effet « drop » sur les deux protocoles**

Le tableau ci-dessous représente l'effet du "drop" sur le protocole BitTorrent.

Les règles	L'effet du drop
Règle N°1	Bloquer le site torrent «Cpasbien»
Règle N°2 et N°3	Bloquer la connexion avec le tracker
Règle N° 4 et N°5	Bloquer la connexion avec les Peers
Règle N°6 et N°7	Bloquer la connexion avec le tracker et les Peers via la DHT

Tableau 4.1 : Effet "drop"sur le protocole BitTorrent.

Le tableau ci-dessous représente l'effet du **«** drop **»** sur le protocole IPFS.

Les règles	L'effet du drop
Règle N°1	Bloquer l'identifiant du pair (Peer ID)
Règle N°2	Bloquer l'utilisation de la bande passante (bloquer l'application entièrement)
Règle N°3	Supprimer le fichier de configuration
Règle N°4	Bloquer la connexion avec tous les Peers
Règle N°5 / N°6	Suppression des fichiers stockés dans le client IPFS
Règle N°7 / N°8	Empêcher l'ajout et la récupération des fichiers sur le client IPFS
Règle N°9 / N°10	Restreindre l'accès au contenu des fichiers

Tableau 4.2 : Effet "drop" sur le protocole IPFS*.*

4.5.2 Implémentation des règles dans suricata

On a implémenté les règles BitTorrent dans le fichier ***«* bitt.rules *»*** et les règles IPFS dans le fichier ***«* ipfs.rules *»***.

```
default-rule-path: /var/lib/suricata/rules

rule-files:
  - #suricata.rules
  - bitt.rules
  - ipfs.rules
```

Figure 4.3 : L'implémentation des fichiers dans "suricata.yml".

```
root@p2pserver: /var/lib/suricata/rules
 GNU nano 6.2                                  bitt.rules *
alert tcp any any -> any any (msg:"site torrent cpasbien"; content:"www.cpasbien.sk"; sid:1; rev:1;

alert tcp any any -> any any (msg:"Attention utilisation BitTorrent:get scrape"; content:"GET /scrape?info_hash"; content:"User-Age>

alert tcp any any -> any any (msg:"Attention utilisation BitTorrent:get anounnce"; content:"GET /announce?info_hash"; content:"User>

alert tcp any any -> any any (msg:"Attention utilisation BitTorrent:handshake"; content:"BitTorrent protocol"; sid:4; rev:1; )

alert tcp any any -> any any (msg:"Attention utilisation BitTorrent:Extended Bitfield"; content:"ut_metadata"; content:"metadata_si>

alert udp any any -> any any (msg:"Attention utilisation BitTorrent:DHT Ping announce"; content:"/announce"; sid:6; rev:1; )

alert udp any any -> any any (msg:"Attention utilisation BitTorrent:DHT Ping Peer"; content:"d1:ad2:id20" ; content:"info_hash20"; >


drop tcp any any -> any any (msg:"site torrent cpasbien"; content:"www.cpasbien.sk"; sid:8; rev:1; )

drop tcp any any -> any any (msg:"Attention utilisation BitTorrent:get scrape"; content:"GET /scrape?info_hash"; content:"User-Agen>

drop tcp any any -> any any (msg:"Attention utilisation BitTorrent:get anounnce"; content:"GET /announce?info_hash"; content:"User->

drop tcp any any -> any any (msg:"Attention utilisation BitTorrent:handshake"; content:"BitTorrent protocol"; sid:11; rev:1; )

drop tcp any any -> any any (msg:"Attention utilisation BitTorrent:Extended Bitfield"; content:"ut_metadata" ; content:"metadata_si>

drop tcp any any -> any any (msg:"Attention utilisation BitTorrent:DHT Ping announce"; content:"/announce"; sid:13; rev:1; )

drop udp any any -> any any (msg:"Attention utilisation BitTorrent:DHT Ping Peer"; content:"d1:ad2:id20" ; content:"info_hash20"; c>
```

Figure 4.4 : L'implémentation des règles BitTorrent dans le fichier "bitt.rules"**.**

```
 GNU nano 6.2                                  ipfs.rules *
alert tcp any any -> any any (msg:"Attention utilisation ipfs:Peer_ID"; content:"POST /api/v0/id"; sid:1; rev:2; )

alert tcp any any -> any any (msg:"Attention utilisation ipfs:Bande Passante"; content:"POST /api/v0/stats/bw"; content:"User-Agent>

alert tcp any any -> any any (msg:"Attention utilisation ipfs:Config_IPFS"; content:"POST /api/v0/config/show"; content:"User-Agent>

alert tcp any any -> any any (msg:"Attention utilisation ipfs:Swarm"; content:"POST /api/v0/swarm/peers?verbose"; content:"User-Age>

alert tcp any any -> any any (msg:"Attention utilisation ipfs:repertoire racine"; content:"POST /api/v0/Files/stat?arg"; content:"U>

alert tcp any any -> any any (msg:"Attention utilisation ipfs:fichier stockés"; content:"POST /api/v0/ls?arg"; content:"User-Agent:>

alert tcp any any -> any any (msg:"Attention utilisation ipfs:ajouter un fichier"; content:"POST /api/v0/add?stream-channels"; cont>

alert tcp any any -> any any (msg:"Attention utilisation ipfs:récupérer un fichier"; content:"POST /api/v0/block/get?arg"; content:>

alert tcp any any -> any any (msg:"Attention utilisation ipfs:consulter un fichier"; content:"POST /api/v0/cat?arg"; content:"User->

alert tcp any any -> any any (msg:"Attention utilisation ipfs:afficher le fichier"; content:"GET /ipfs"; sid:10; rev:2; )


drop tcp any any -> any any (msg:"Attention utilisation ipfs:Peer_ID"; content:"POST /api/v0/id"; sid:11; rev:2; )

drop tcp any any -> any any (msg:"Attention utilisation ipfs:Bande Passante"; content:"POST /api/v0/stats/bw"; content:"User-Agent:>

drop tcp any any -> any any (msg:"Attention utilisation ipfs:Config_IPFS"; content:"POST /api/v0/config/show"; content:"User-Agent:>

drop tcp any any -> any any (msg:"Attention utilisation ipfs:Swarm"; content:"POST /api/v0/swarm/peers?verbose"; content:"User-Agen>

drop tcp any any -> any any (msg:"Attention utilisation ipfs:repertoire racine"; content:"POST /api/v0/Files/stat?arg"; content:"Us>

drop tcp any any -> any any (msg:"Attention utilisation ipfs:fichier stockés"; content:"POST /api/v0/ls?arg"; content:"User-Agent: >

drop tcp any any -> any any (msg:"Attention utilisation ipfs:ajouter un fichier"; content:"POST /api/v0/add?stream-channels"; conte>

drop tcp any any -> any any (msg:"Attention utilisation ipfs:récupérer un fichier"; content:"POST /api/v0/block/get?arg"; content:">

drop tcp any any -> any any (msg:"Attention utilisation ipfs:consulter un fichier"; content:"POST /api/v0/cat?arg"; content:"User-A>

drop tcp any any -> any any (msg:"Attention utilisation ipfs:afficher le fichier"; content:"GET /ipfs"; sid:20; rev:2; )
```

Figure 4.5 : L'implémentation des règles IPFS dans le fichier "ipfs.rules"**.**

4.6 Test de fonctionnement

Pour tester le fonctionnement de nos règles, on va d'abord lancer suricata et vérifier le statut de suricata pour s'assurer que suricata est en marche.

- **Mode IDS**

Par défaut, suricata est configuré pour fonctionner comme un système de détection d'intrusion « IDS », qui génère uniquement des alertes et enregistre le trafic suspect.

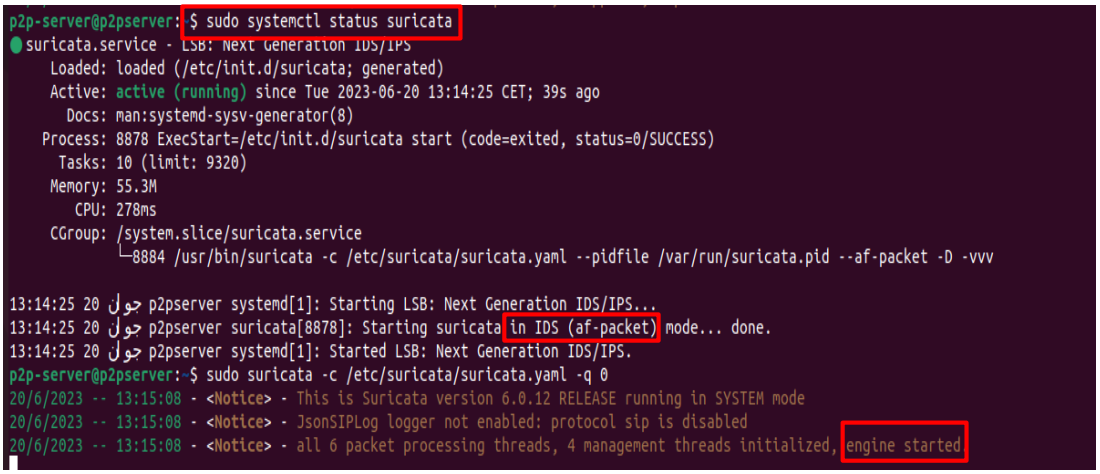


Figure 4.6 : Statu suricata active "Mode IDS".

Pour consulter le fichier d'alerte de suricata, on va exécuter la commande suivante « **tail –f /var/log/suricata/fast.log** ».

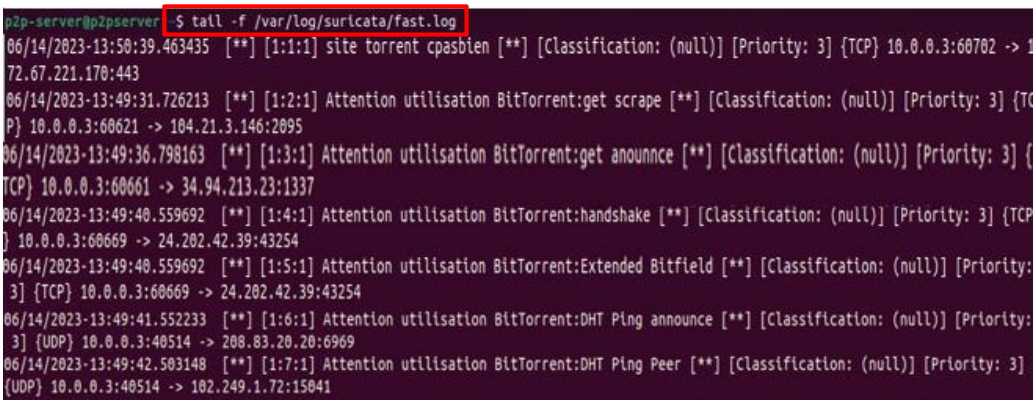


Figure 4.7 : Le fichier d'alerte de suricata "alert BitTorrent".

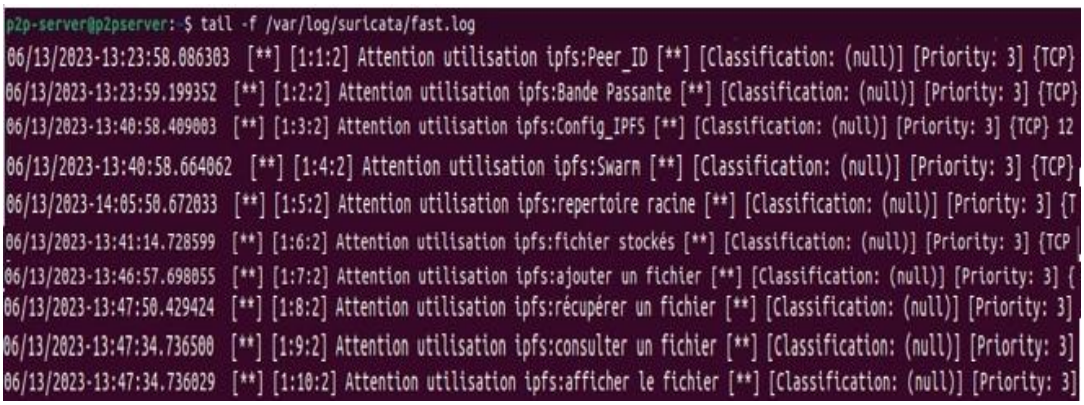


Figure 4.8 : Le fichier d'alerte de suricata "alert IPFS".

- **Mode IPS**

Lorsqu'on active le mode IPS, Suricata peut bloquer automatiquement le trafic réseau suspect.

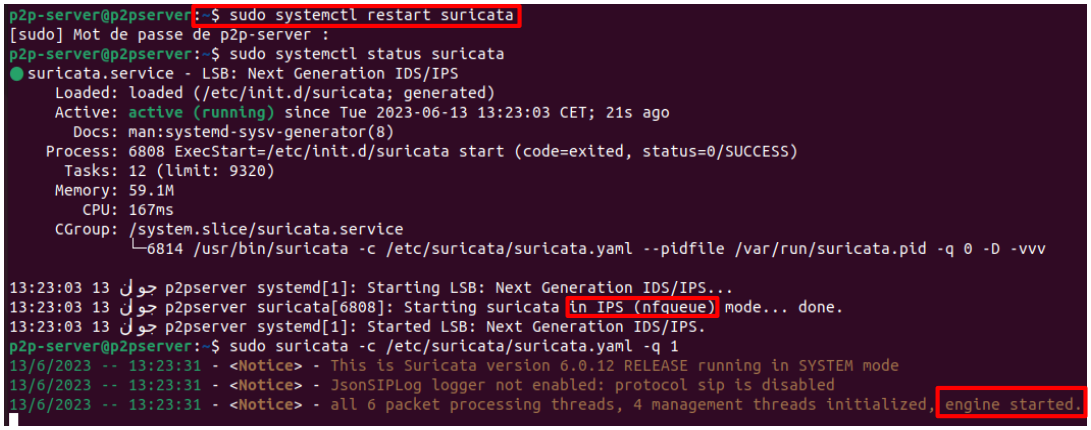


Figure 4.9 : Statu suricata active "Mode IPS".

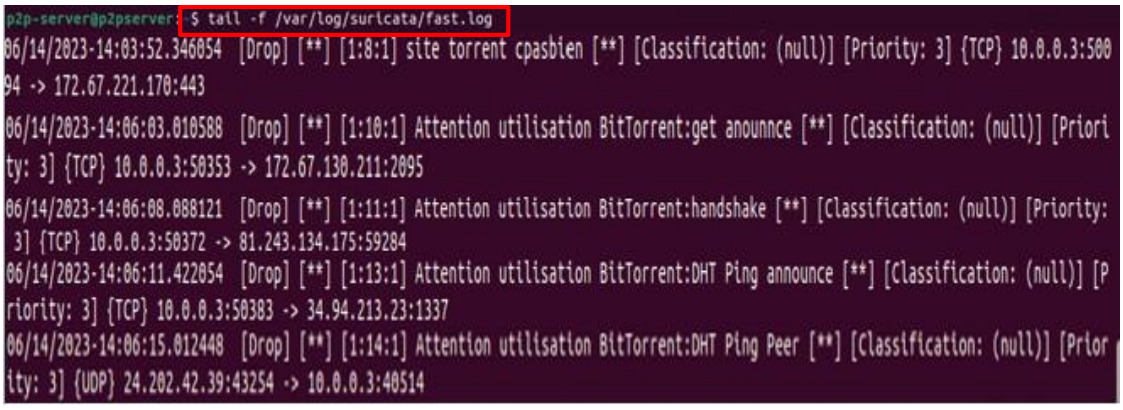


Figure 4.10 : Le fichier d'alerte de suricata "drop BitTorrent".

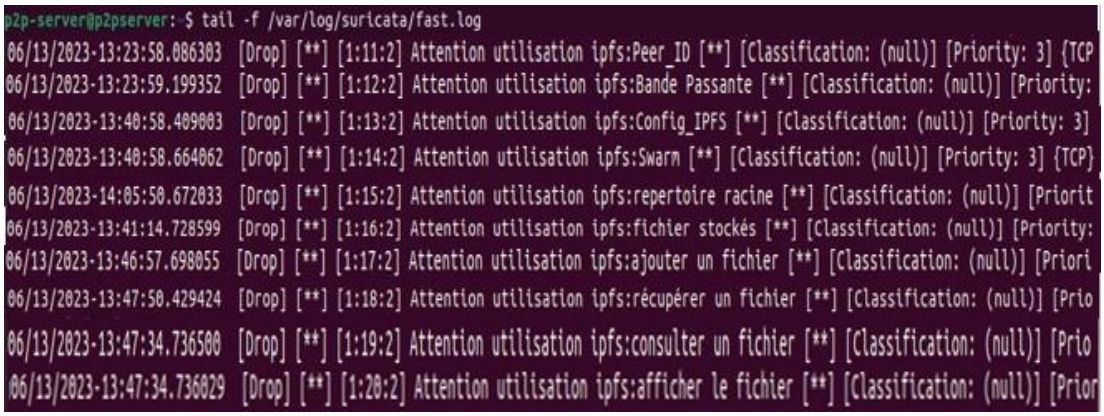


Figure 4.11 : Le fichier d'alerte de suricata "drop IPFS".

Après avoir lancé IDS/IPS Suricata, on observe que toutes les règles qu'on a créées génèrent des alertes. Ces alertes affichent la date à laquelle elles ont été déclenchées ainsi qu'une brève description de l'alerte.

Afin d'améliorer l'affichage des alertes et faciliter leur lecture, on a décidé d'intégrer suricata avec une plateforme appelée Wazuh.

4.7 Intégration de suricata avec la plateforme Wazuh

Wazuh est une plateforme gratuite et open source qui offre des fonctionnalités de gestion des informations et des événements de sécurité (SIEM). Elle est conçue pour la détection de menaces, les tentatives d'intrusion, les anomalies du système et les actions non autorisées des utilisateurs. Wazuh peut être utilisé pour surveiller les terminaux, les services Cloud et les conteneurs, ainsi que pour agréger et analyser les données provenant de sources externes [29].

La plateforme Wazuh est basée sur les composantes suivantes [29]:

- **Serveur Wazuh :** ce composant central est utilisé pour analyser les données reçues des agents.
- **L'agent Wazuh :** est installé sur des terminaux tels que des ordinateurs portables, bureau, des serveurs…, il transmet les données collectées au serveur Wazuh.
- **L'indexeur Wazuh :** ce composant central indexe et stocke les alertes générées par le serveur Wazuh.

- **Le tableau de bord Wazuh « dashboard » :** est l'interface utilisateur Web pour la visualisation et l'analyse des données.

Dans notre cas, on utilise Wazuh pour stocker et analyser les journaux (logs) provenant du serveur PC.

Afin de mettre en place cette configuration, on a procédé à l'installation du serveur Wazuh sur un nouveau PC, Par la suite, on a installé l'agent Wazuh sur le PC serveur où Suricata est déjà installé. L'agent Wazuh a pour rôle de collecter et de transmettre les journaux d'alerte de Suricata au serveur Wazuh.

4.8 Architecture du test final

L'architecture du test final est montrée dans la figure ci-dessous :

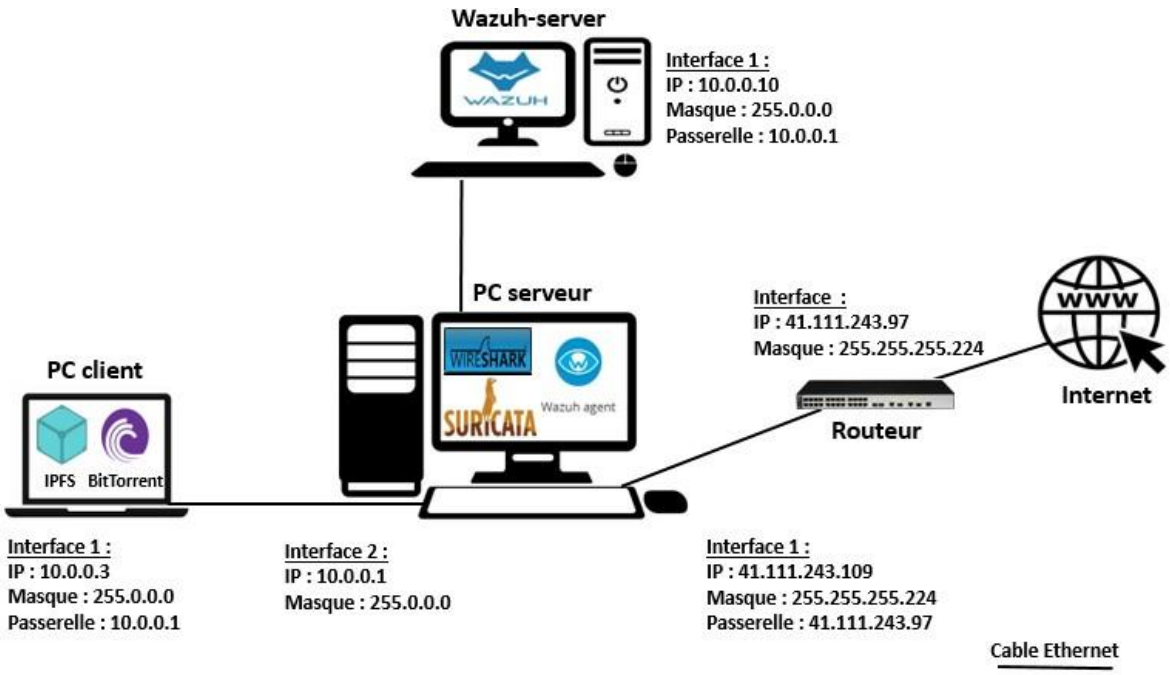


Figure 4.12 : Architecture du test final.

Dans le pc serveur on a installé Wireshark, suricata IDS/IPS, l'agent Wazuh, tous le trafic en provenance du pc client passe par le pc serveur pour être détecter et bloquer par IDS/IPS suricata, ensuite, affiché les alertes générées par ce dernier dans le serveur Wazuh.

4.9 Test de fiabilité

Afin de vérifier l'efficacité de nos règles, on va procéder à une série de tests. Tout d'abord, on va lancer l'application BitTorrent pour évaluer si Suricata peut détecter et bloquer son utilisation conformément à nos règles. Ensuite, on va effectuer un test similaire en lançant l'application IPFS Desktop.

On va lancer aussi la plateforme Wazuh pour visualiser les alertes en temps réel.

4.9.1 BitTorrent

Une fois que le client lance le téléchargement torrent, l'interface Wazuh affiche les alertes suivantes :

>	Jun 14, 2023 @ 14:49:37.017	Peer_to_Peer	Suricata: Alert - Attention utilisation BitTorrent:DHT Ping announce	3	86601
>	Jun 14, 2023 @ 14:49:37.017	Peer_to_Peer	Suricata: Alert - Attention utilisation BitTorrent:DHT Ping Peer	3	86601
>	Jun 14, 2023 @ 14:49:41.031	Peer_to_Peer	Suricata: Alert - Attention utilisation BitTorrent:Extended Bitfield	3	86601
>	Jun 14, 2023 @ 14:49:41.029	Peer_to_Peer	Suricata: Alert - Attention utilisation BitTorrent:handshake	3	86601
>	Jun 14, 2023 @ 14:49:37.065	Peer_to_Peer	Suricata: Alert - Attention utilisation BitTorrent:get anounnce	3	86601
>	Jun 14, 2023 @ 14:49:32.998	Peer_to_Peer	Suricata: Alert - Attention utilisation BitTorrent:get scrape	3	86601
>	Jun 14, 2023 @ 14:50:43.068	Peer_to_Peer	Suricata: Alert - site torrent cpasbien	3	86601

Figure 4.13 : Les alertes lancées par suricata "Client BitTorrent".

- Pour une analyse plus approfondie, on peut cliquer sur l'une des alertes par exemple *«* get announce *»* cela donnera plus de détails (voir Figure 4.14).

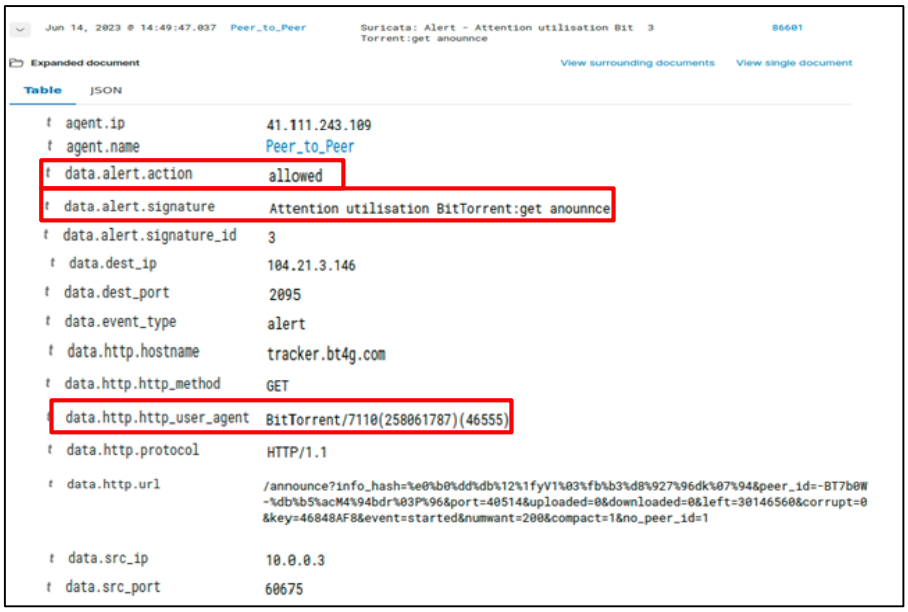


Figure 4.14 : Détail de l'alerte "get announce".

À partir de ces informations, on peut obtenir certains détails comme la signature d'alerte "Attention utilisation BitTorrent : *«* get announce *»*, adresse IP *«* 10.0.0.3*»* (c'est l'adresse de notre client BitTorrent) et User-Agent utilisé "BitTorrent7.11". De là on constate que nos règles sont correspondantes au trafic BitTorrent.

- Dans la figure ci-dessous, on montre la même alerte *«* get announce *»* qui a été également bloqué.

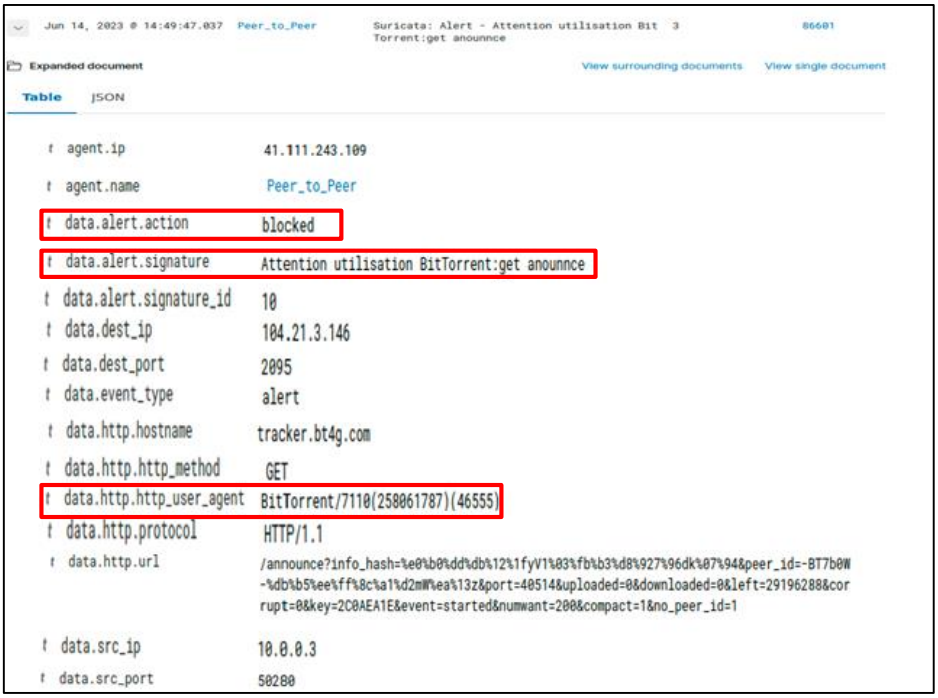


Figure 4.15 : L'alerte get announce "drop".

- Le diagramme de la figure ci-dessous, nous montre le taux d'alerte le plus élevé (DHT Ping Peer) :

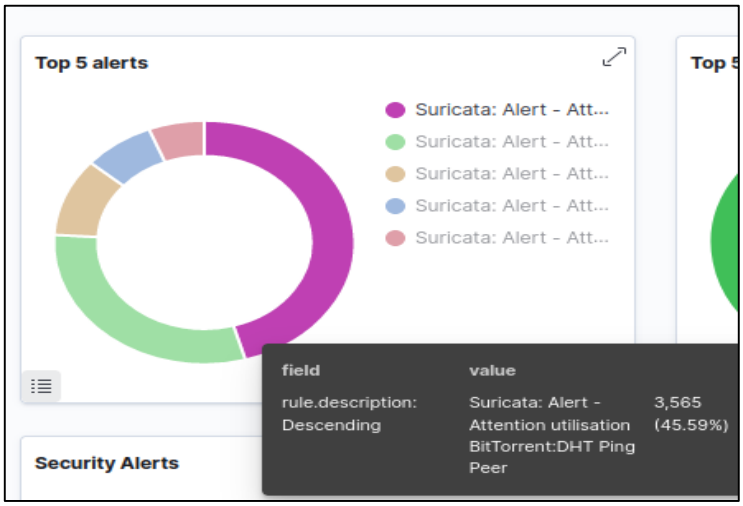


Figure 4.16 : Diagramme d'alerte "BitTorrent".

❖ **Constatation**

✓ On constate que Suricata a réussi à détecter et à bloquer l'utilisation de l'application BitTorrent.

✓ Toutes nos règles génèrent des alertes qui sont illustrées dans la figure (4.13).

✓ On remarque aussi que le taux d'alerte généré par la règle « **DHT Ping Peer** » est plus élevé (**49.26 %)** et cela est totalement normal parce que cette règle génère une alerte sur chaque requête envoyée vers le Peer.

✓ Le protocole BitTorrent repose initialement sur le protocole **HTTP** pour établir la connexion avec le tracker.

Note : Jusqu'à présent, le test a été réalisé en mode non chiffré (en clair).

a BitTorrent (Mode crypté) :

On va répéter le même processus de téléchargement, mais cette fois-ci en activant le mode crypté. Les alertes affichées par Suricata sont représentées dans la figure ci-dessous.

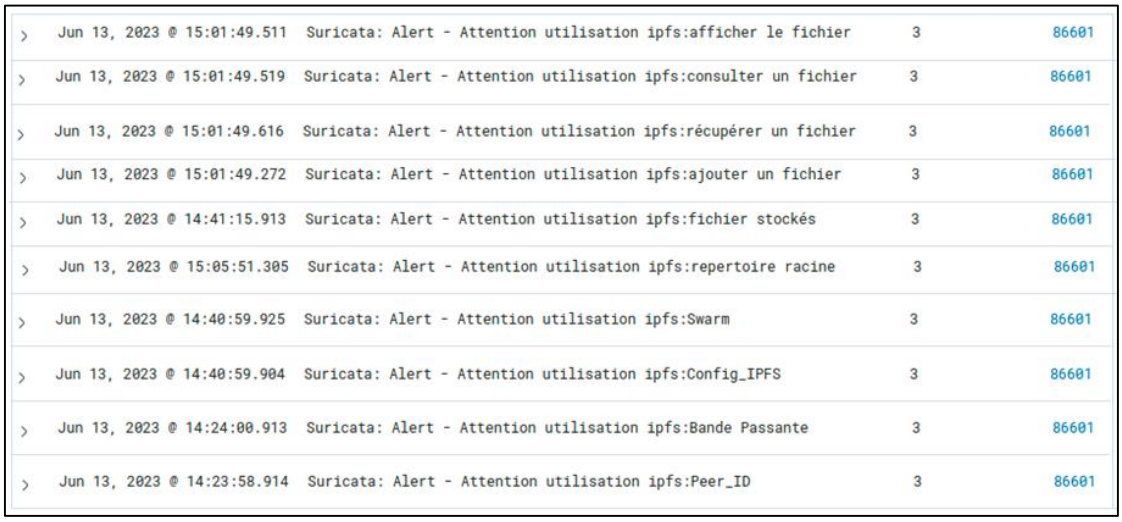
Jun 20, 2023 @ 14:01:14.779	Suricata: Alert - Attention utilisation BitTorrent:DHT Ping Peer	3	86601
Jun 20, 2023 @ 14:01:10.773	Suricata: Alert - Attention utilisation BitTorrent:DHT Ping Peer	3	86601
Jun 20, 2023 @ 14:01:04.765	Suricata: Alert - Attention utilisation BitTorrent:DHT Ping Peer	3	86601
Jun 20, 2023 @ 14:00:58.763	Suricata: Alert - Attention utilisation BitTorrent:DHT Ping Peer	3	86601
Jun 20, 2023 @ 14:00:56.777	Suricata: Alert - Attention utilisation BitTorrent:DHT Ping announce	3	86601
Jun 20, 2023 @ 14:00:54.783	Suricata: Alert - Attention utilisation BitTorrent:get scrape	3	86601
Jun 20, 2023 @ 14:00:54.773	Suricata: Alert - Attention utilisation BitTorrent:get anounnce	3	86601
Jun 20, 2023 @ 13:58:24	Suricata: Alert - site torrent cpasbien	3	86601

Figure 4.17 : Les alertes lancées par suricata "BitTorrent Mode crypté".

- Suricata détecte l'utilisation en mode crypté de l'application BitTorrent et émet des alertes qui répondent à nos **5** règles.
- On remarque que les deux règles « **Handshake** » et « **Extended bitfield** » du protocole BitTorrent ne déclenchent pas d'alertes en raison du cryptage.

4.9.2 IPFS Desktop

Une fois que le client utilise IPFS Desktop, Suricata lance des alertes qui sont représentées dans la figure suivante :



Jun 13, 2023 @ 15:01:49.511	Suricata: Alert - Attention utilisation ipfs:afficher le fichier	3	86601
Jun 13, 2023 @ 15:01:49.519	Suricata: Alert - Attention utilisation ipfs:consulter un fichier	3	86601
Jun 13, 2023 @ 15:01:49.616	Suricata: Alert - Attention utilisation ipfs:récupérer un fichier	3	86601
Jun 13, 2023 @ 15:01:49.272	Suricata: Alert - Attention utilisation ipfs:ajouter un fichier	3	86601
Jun 13, 2023 @ 14:41:15.913	Suricata: Alert - Attention utilisation ipfs:fichier stockés	3	86601
Jun 13, 2023 @ 15:05:51.305	Suricata: Alert - Attention utilisation ipfs:repertoire racine	3	86601
Jun 13, 2023 @ 14:40:59.925	Suricata: Alert - Attention utilisation ipfs:Swarm	3	86601
Jun 13, 2023 @ 14:40:59.904	Suricata: Alert - Attention utilisation ipfs:Config_IPFS	3	86601
Jun 13, 2023 @ 14:24:00.913	Suricata: Alert - Attention utilisation ipfs:Bande Passante	3	86601
Jun 13, 2023 @ 14:23:58.914	Suricata: Alert - Attention utilisation ipfs:Peer_ID	3	86601

Figure 4.18 : Les alertes lancées par suricata "Client IPFS".

- On peut cliquer sur l'alerte « **Bande Passante** » pour obtenir des informations détaillées sur celle-ci.

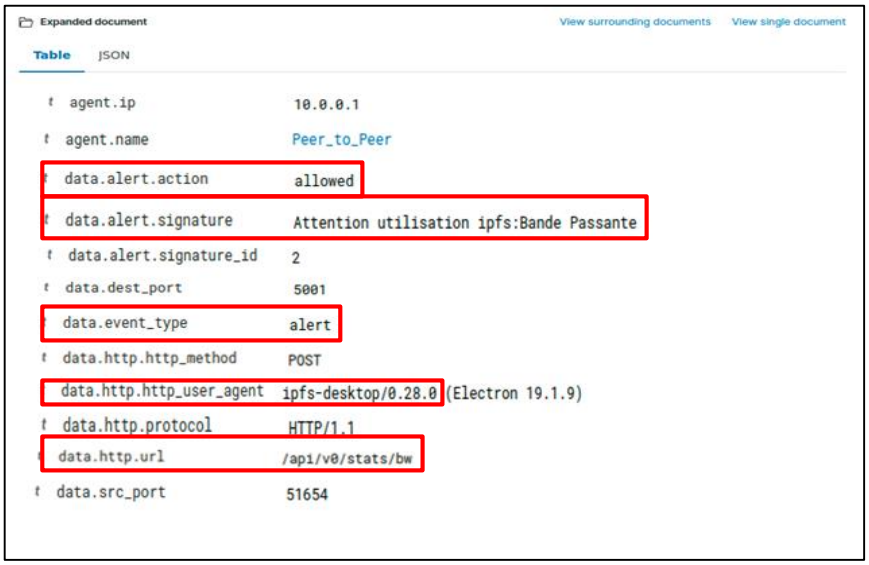


Figure 4.19 : Détail de l'alerte "Bande Passante".

Cette capture nous montre clairement la signature d'alerte « Attention utilisation IPFS : Bande Passante », User-Agent utilisé : IPFS Desktop 0.28, l'action d'alerte « allowed ». Cela nous permet de dire que ces alertes correspondent au trafic IPFS.

- Dans la figure ci-dessous, on montre la même alerte ***« Bande Passante »*** qui a été également bloqué.

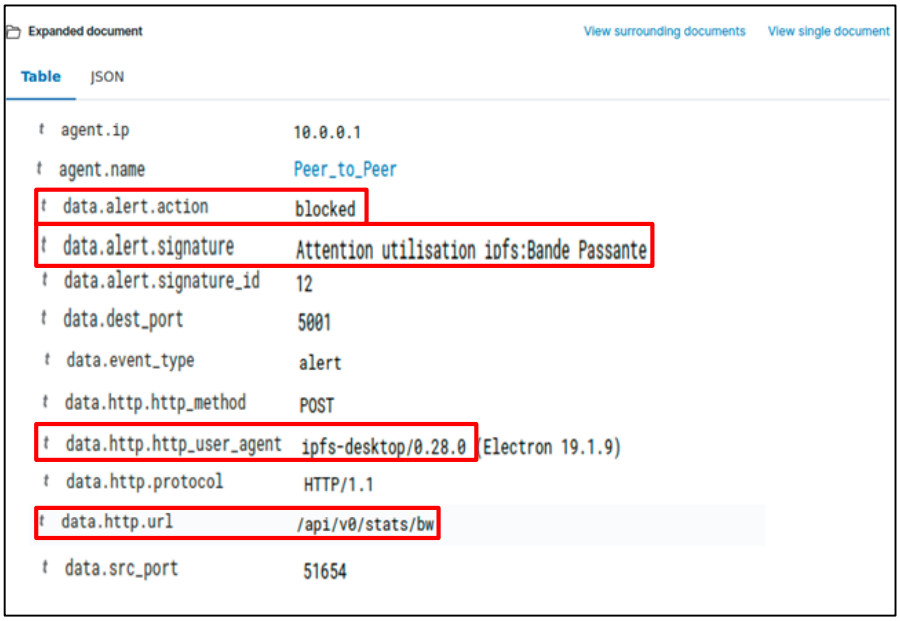


Figure 4.20 : L'alerte Bande passante "drop".

- Le Diagramme de la figure (4.20), nous montre le taux d'alerte le plus élevé (afficher le fichier) :

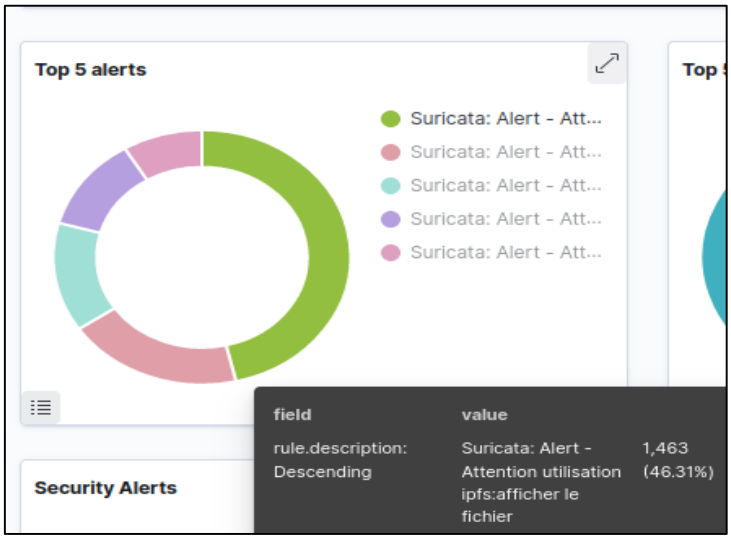


Figure 4.21 : Diagramme d'alerte "IPFS".

❖ **Constatation**

- ✓ On constate que Suricata a réussi à détecter et à bloquer l'utilisation de l'application IPFS Desktop.
- ✓ Les Alertes sont lancés en temps réel.
- ✓ On remarque aussi que le taux d'alerte généré par la règle « afficher le ficher » est plus élevé **(46.31%)** et cela est totalement normal parce que cette règle génère une alerte sur chaque requête envoyée lorsque on veut afficher le contenu d'un fichier.
- ✓ Si on bloque les trois premières règles (bande passante, config-ipfs, Peer ID), on remarque l'absence des autres règles, car celles-ci sont conçues pour bloquer entièrement le fonctionnement de l'application IPFS Desktop.

4.10 Discussion

- L'ensemble des règles élaborées ont été testées dans un environnement de laboratoire, pour cette raison le nombre de faux positifs est quasiment nul.
- Le trafic Peer to Peer (BitTorrent et IPFS) a été détecté et bloqué.
- Pour améliorer encore notre solution sur un environnement réel (entreprise ou campus universitaire) on propose de mettre en place :
  - **Un serveur annuaire** qui bloque complètement les utilisateurs suspects lorsqu'ils utilisent des applications Peer to Peer. Chaque utilisateur est associé à un profil spécifique à travers ce serveur afin de pouvoir afficher et identifier les utilisateurs du réseau P2P par leurs propres comptes.

4.11 Conclusion

Ce chapitre nous a permis de mettre en évidence notre approche proposée basée sur la détection par signatures, on a réussi à détecter et bloquer l'utilisation des réseaux P2P en temps réel à l'aide des règles présentées précédemment, ce qui confirme la fiabilité de celle-ci.

L'approche a permis non seulement la détection à travers les alertes lancées mais aussi blocage des paquets suspects, la mise en place de nos règles dans l'IDS/IPS suricata avec l'interface graphique Wazuh a considérablement facilité le suivi temps réel du trafic dans le réseau.

Conclusion générale

BitTorrent et IPFS sont des protocoles de partage de fichiers populaires et largement utilisés dans le domaine P2P. Ils offrent une méthode efficace pour la distribution de fichiers à un grand nombre d'utilisateurs. Cependant, il est important de reconnaître les inconvénients associés à ces applications, tels que les risques des attaques (Dos/DDos…) et de logiciels malveillants. Par conséquent, ils permettent de contourner les mesures de sécurité imposées par les réseaux, en particulier les réseaux d'entreprise, compromettant ainsi la sécurité de leurs systèmes d'information.

Notre travail s'est principalement concentré sur la sécurisation d'un réseau d'entreprise contre l'utilisation des protocoles Peer to Peer "IPFS" et "BitTorrent". Dans ce contexte, on a proposé une solution visant à détecter et limiter l'utilisation de ces protocoles au sein d'un réseau local. Cette détection s'est déroulée en plusieurs étapes:

Dans la première étape, on a capturé les paquets qui transitaient dans le réseau d'entreprise en utilisant le logiciel "Wireshark". Cela nous a permis d'extraire les empreintes numériques spécifiques à ces deux applications P2P, à savoir BitTorrent et IPFS Desktop.

Ensuite, dans la deuxième étape, les empreintes ont été implémentées dans le système de détection et de prévention d'intrusion (IDS/IPS) Suricata avec l'interface "Wazuh". Cela a permis de générer des alertes qui ont signalé et bloqué les menaces.

Enfin, la dernière phase a consisté à tester la solution réalisée au niveau de notre laboratoire pour valider le bon fonctionnement de nos règles. Les tests ont été très concluants, car on a pu détecter et bloquer toutes les utilisations des applications BitTorrent et IPFS Desktop.

En conclusion, on peut affirmer qu'on a atteint notre objectif. Pour améliorer encore la détection du réseau P2P dans une entreprise, on recommande les points suivants :

- Mettre en œuvre un serveur d'annuaire.
- Mise à jour fréquente de la base des signatures de NIDS.
- Imposer des règles strictes et sévères pour l'utilisation de ces applications.
- La vigilance à travers la sensibilisation et les formations.

Bibliographie

[1] António Godinho , José Rosado, Filipe Sá , Filipe Caldeira et Filipe Cardoso, «Torrent Poisoning Protection with a Reverse Proxy Server,» 2022-12-30.

[2] anukriti16, «geeksforgeeks.org,» : https://www.geeksforgeeks.org/what-is-p2ppeer-to-peer-process/, accédé le 24/06/2023.

[3] M. GHARZOULI, «Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer,» Le 25 Septembre 2011.

[4] H. Y. E. K. L et John Buford, P2P Networking and Applications, 2009.

[5] M. BELKHIRA et A. KENNCHE , «VERS UN SYSTEME DE RECOMMANDATION P2P,» Année Universitaire 2015/2016 .

[6] Nathalie Budan, Benoit Tedschi et Stéphane Vaubourg, «Nouvelles Technologies Réseau,» 2003.

[7] O. Lausecker et L. Viennot, «Interstices - Les réseaux de pair à pair,» 03 05 2016 : https://interstices.info/les-reseaux-de-pair-a-pair/, accédé le 14/05/2023.

[8] GHEMBAZA Hayat et KAID SLIMANE Imane, «Cryptographie symétrique des messages dans un réseau P2P sur JXTA,» 03/ 07/2017.

[9] ABDELOUAHAB FORTAS, «réputation dans les systèmes paire à paire,» 06/03/2014.

[10] Gaetan de Menten, «BitTorrent le chainon manquant des protocoles peer to peer,» 17 juin 2004.

[11] Lan Yu, «A reputation systme for BitTorrent peer-to-peer file-sharing network,» 2006.

[12] Brami Frank et Delattre Arnaud, «Protocol BitTorrent,» 2005-2006.

[13] Jere Keltamäki, «BITTORRENT PROTOCOL,» 2010.

[14] A. Manoj Athreya, Ashwin A. Kumar, S. M. Nagarajat, H. L. Gururaj, V. Ravi Kumar, D. N. Sachin et K. R. Rakesh, «Peer-to-Peer Distributed Storage Using InterPlanetary File System,» janvier 2021.

[15] «IPFS : l'hébergement de données en pair-à-pair,» [En ligne]. Available: https://cryptoms.fr. accédé le 18/05/23.

[16] Juan Benet, «IPFS - Content Addressed, Versioned, P2P File System,» 14 Jul 2014.

[17] Alexandru-Gabriel Cristeaa, Lenuta Alboaie, Andrei Panu et Vlad Radulescu, «Offline but still connected with IPFS based communication,» 2020.

[18] Fléix Freitag, «IPFS as a foundation for anonymous file storage,» 01/2020.

[19] Sebastian Henningsen, Martin Florian , Sebastian Rust et Björn Scheuermann, «Mapping the Interplanetary Filesystem,» 18 février 2020.

[20] Achmad Muhaimin Aziz , Adityas Widjajarto et Avon Budiyono, «Analysis and Implementation of Nodes Communication Between InterPlanetary File System (IPFS) in Smart Contract Ethereum,» 2019.

[21] «IPFS comparisons,» : https://docs.ipfs.tech, accédé le 01/06/2023.

[22] Ubuntu : https://www.ubuntu-fr.org, accédé le 03/06/2023.

[23] BitTorrent : https://www.bittorrent.com/, accédé le 10/06/2023.

[24] IPFS Desktop : https://github.com, accédé le 29/05/23.

[25] Wireshark : https://www.wireshark.org, accédé le 24/05/2023.

[26] Nicolas TOURRETTE , «Mise en place d'un IDS,» Le 31 janvier 2020.

[27] «Qu'est-ce qu'un Système de prévention des intrusions (IPS) ?,» : https://www.forcepoint.com .

[28] «Comprendre les signatures Suricata,» : https://getdoc.wiki, accédé le 19/06/23.

[29] Wazuh : https://wazuh.com, accédé le 14/06/23.