LINFO1341 : Projet 1
Analyse d'application réseau

LECHAT Jérôme
50351800
jerome.lechat@student.uclouvain.be

DELSART Mathis
31302100
mathis.delsart@student.uclouvain.be

I. Introduction

Dans ce rapport, nous commencerons par présenter une analyse des requêtes DNS, qui constituent souvent la première étape de la communication entre une application et les serveurs distants. Nous poursuivrons ensuite avec une analyse des différentes couches réseau et transport utilisées par l'application, afin de mieux appréhender le flux de données à travers le réseau. Par la suite, nous aborderons la question du chiffrement et de la sécurité mis en place par l'application OneDrive pour protéger les données des utilisateurs lors de leur transfert sur le réseau. Nous examinerons les protocoles de chiffrement utilisés et les mesures de sécurité mises en oeuvre pour garantir la confidentialité et l'intégrité des données. Enfin, nous approfondirons notre analyse en nous concentrant sur des parties spécifiques de l'application OneDrive, afin de comprendre en détail son fonctionnement interne et ses interactions avec les différents composants du système.

II. Nos outils

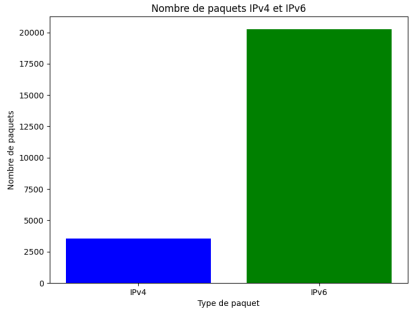
Nous avons utilisé Wireshark pour tracer les paquets entrants et sortants de notre réseau. Les captures ont été réalisées sur différents systèmes d'exploitation tels que MacOS, Linux, Windows 10 et Windows 11, ainsi que dans différents scénarios de réseau (Wi-Fi, Ethernet). À l'aide de scripts personnalisés, nous avons effectué une analyse efficace des paquets pour extraire un maximum de données pertinentes. Tous les scripts d'analyse et de création de graphiques, ainsi que les captures et vidéos utilisées, sont disponibles dans notre dépôt GitHub, accessible via le lien donné en référence. Un tutoriel détaillé est également disponible dans le fichier README.md du dépôt, offrant une documentation complète sur l'utilisation des scripts, la manipulation des données capturées mais également une collection de filtres Wireshark qui nous ont été utiles tout au long du développement de ce projet.

III. Domain Name System

La résolution des noms de domaine se produit principalement lors de l'accès à l'application web, du rafraîchissement de la page ou lors de l'inscription/connexion à l'application. Il se peut également, dans de rare cas, que les résolutions se réitèrent en cas d'expiration du cache DNS. Cette information nous est donné via l'index **TTL** pour **T**ime **T**o **L**ive. Nous avons observé des requêtes en direction de *onedrive.live.com* lors de l'accès au site. Lors du processus de connexion, nous retrouvons des requêtes en direction de *login.live.com*, ainsi que d'autres domaines tels que *browser.pipe.aria.microsoft.com* et *onedscolprdcus03.centralus.cloudapp.azure.com*.

Les serveurs autoritatifs, possédant le type **SOA** pour **S**tart **O**f a zone of **A**uthority, sont variés pour les noms de domaine résolus, comprenant des serveurs appartenant à Microsoft tels que *spov-msedge.net*, ainsi que d'autres services Microsoft comme *trafficmanager.net* et *cloudapp.azure.com*, faisant partie intégrante du service Azure.

La résolution des noms de domaine a révélé des chaînes de réponses **CNAME** (association de sous-domaines à un domaine principal), indiquant un processus complexe utilisant plusieurs redirections avant d'atteindre le nom de domaine final. Par exemple, la chaîne de réponse CNAME de onedrive.live.com est la suivante : *onedrive.live.com* → *web.fe.1drv.com* → *odc-web-geo.onedrive.akadns.net* → *odc-web-brs.onedrive.akadns.net* → *odwepl.trafficmanager.net.dual-spov-006.spov-msedge.net* → *dual-spov-006.spov-msedge.net*
Cela signifie que *dual-spov-006.spov-msedge.net* est probablement le nom de domaine le plus proche de l'adresse IP finale qui sera utilisée pour établir la connexion. A l'aide de ces redirections CNAME, de l'analyse de multiples frames et du site *whois.com*, nous avons pu conclure que la majorité des noms de domaine résolus sont liés à Microsoft et à ses services tiers tel qu'Azure, mais nous avons également identifié l'intervention d'une société extérieure, Akamai, avec le domaine *akadns.net*.



Le type dominant des requêtes DNS semble être le type **AAAA**. Cependant, les réponses à ces requêtes sont quant

à elles toujours de type CNAME. L'**IPv6** est la version IP demandée par l'application. En effet, le rapport $\frac{type(AAAA)}{type(A)}$ est très élevée comme illustré à la figure précédente, ce qui révèle une préférence pour IPv6 par rapport à **IPv4**.

Aucun record additionnel n'a été trouvé lors de l'analyse des requêtes DNS. Il n'y a également eu aucun comportement DNS innatendu au cours de l'analyse.

IV. Couche réseau

Nous avons observé très peu de paquet envoyé via IPv4. L'unique raison est que **NAT** (**N**etwork **A**dress **T**ranslation) pose des problèmes potentiels avec les adresses IPv4. En effet, la version 4 du protocole Internet utilise un espace d'adressage limité, avec seulement environ 4 milliards d'adresses disponibles. Avec la croissance exponentielle d'Internet et l'expansion du nombre d'appareils connectés, cette quantité d'adresses IPv4 disponibles est devenue insuffisante. Cependant, lorsque IPv4 est utilisé, l'application utilise la technique de **UPnP** (**U**niversal **P**lug and **P**lay) pour traverser les NAT. UPnP permet à des périphériques de se connecter aisément et de simplifier la mise en oeuvre de réseaux domestiques.

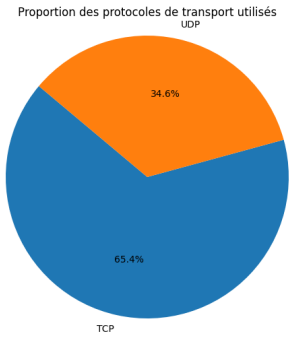
Les adresses accédées par ces paquets IPv4 sont dédiées uniquement au processus de connexion de l'application et pour la synchronisation des données. Une tendance spécifique émerge des adresses IP utilisées, qui pointent toutes vers des serveurs Microsoft ou vers la plateforme de cloud computing de Microsoft nommée Azure, confirmant ainsi la nature étroite de la collaboration avec Microsoft (est le développeur et le propriétaire de OneDrive et héberge les données OneDrive). Elles proviennent de serveurs variés. Nous avons observé des adresses IP venant des USA, de France et d'un peu partoût dans le monde. Par exemple, *20.199.120.182:443* est l'une des adresses IP utilisée provenant de France. Nous avons aussi remarqué que toutes les adresses IP pointent vers le *port 443* qui permet une connexion HTTPS sécurisée et la transmission de données chiffrées.

V. Couche transport

Pour chaque fonctionnalité testée, seuls trois ou quatre protocoles de transport ont été observés : **TCP** (**T**ransmission **C**ontrol **P**rotocol), **UDP** (**U**ser **D**atgram **P**rotocol), **HTTP/2** (**H**yper**T**ext **T**ransfert **P**rotocol) et **QUIC** (**Q**uick **U**DP **I**nternet **C**onnections).

Les données capturées indiquent l'utilisation principalement du protocole TCP pour établir des connexions réseau avec les serveurs de OneDrive. TCP est choisi en raison de sa capacité à assurer un transfert fiable des données, garantissant ainsi l'intégrité des échanges entre le client et le serveur. De plus, une analyse approfondie des paquets révèle l'utilisation du protocole TLS, ce qui indique que les communications sont sécurisées via le chiffrement. Cette combinaison de TCP et de TLS dans l'écosystème de OneDrive assure à la fois la fiabilité et la confidentialité des transferts de données, ce qui se traduit par une expérience utilisateur sécurisée et sans faille. Contrairement à TCP, UDP est un protocole non fiable et sans connexion. Cela signifie qu'il ne garantit ni la livraison des données, ni l'ordre dans lequel elles sont reçues. UDP est donc largement préféré dans les applications où la latence est critique, comme les jeux en ligne, la diffusion vidéo en direct (streaming), la voix sur IP (VoIP) et les applications de visioconférence. Ces applications peuvent tolérer une certaine perte de données et privilégient une transmission plus rapide à une fiabilité absolue. De plus, grâce à sa simplicité, UDP présente une faible surcharge de protocole comparativement à TCP. Cela le rend adapté aux applications où la surcharge de contrôle de connexion de TCP n'est pas nécessaire.

En résumé, ***pour OneDrive***, TCP assure un transfert de fichier fiable entre le client et le serveur (et vice-versa). De plus, il permet de réguler le flux de données entre les deux parties, ce qui prévient toute surcharge due à un volume important de données à traiter simultanément. En parallèle, l'utilisation d'UDP permet un transfert de données en temps réel, facilitant notamment la modification simultanée de fichiers en ligne en assurant la synchronisation en temps réel, aussi connue sous le nom de ***multicast***. Nous pouvons observer la proportion de ces deux protocoles sur la figure suivante.



QUIC est un protocole de transport basé sur UDP : il combine les avantages de TCP et UDP en offrant la fiabilité de TCP avec les performances de UDP. Ce protocole est utilisé pour des applications web, notamment par le protocole HTTP/3. La version QUIC utilisée ici est la **version 1** (**0x00000001**) et elle peut facilement être retrouvée lors d'une initialisation et/ou d'un handshake.

De plus, grâce à l'utilisation du fichier **SSLKEYLOGFILE**, nous avons pu décrypter le trafic TLS, révélant l'utilisation du protocole HTTP/2. HTTP/2 est la version la plus récente du protocole HTTP, conçue pour améliorer les performances et l'efficacité des transferts de données sur le web. Il introduit des fonctionnalités telles que la multiplexation des flux, la compression des en-têtes et le push de serveur, ce qui permet d'accélérer le chargement des pages web et d'optimiser l'utilisation des ressources réseau. Ainsi, l'intégration de HTTP/2 dans l'infrastructure de OneDrive garantit des performances accrues et une meilleure expérience utilisateur lors de l'accès aux fichiers et aux données stockés dans le cloud.

Pour ce qui est des multiples connexions vers un même

domaine, nous avons observé plusieurs requêtes consécutives de type *server hello* dès l'accès au site OneDrive, suggérant donc l'établissement de plusieurs connexions vers un même nom de domaine. Cette observation conduit à plusieurs explications plausibles:

- *Optimisation de la performance:*

En initiant plusieurs connexions simultanées, il est possible d'améliorer le chargement des ressources et d'accélérer l'affichage de la page pour l'utilisateur.

- *Gestion de la charge:*

En répartissant la charge sur plusieurs connexions, le serveur peut mieux gérer le trafic et répondre de manière plus efficace aux demandes des utilisateurs, améliorant ainsi l'expérience globale de navigation.

Lors de l'analyse des paquets ***QUIC***, plusieurs extensions de négociation peuvent être observées. Voici les différentes extensions observées:

- ***SNI** (**S**erver **N**ame **I**ndication)*

Cette extension permet au client de spécifier le nom de domaine du serveur auquel il souhaite se connecter lors de l'initialisation de la connexion TLS, facilitant ainsi la sélection du certificat approprié par le serveur.

- ***ALPN** (**A**pplication-**L**ayer **P**rotocol **N**egotiation)*

ALPN permet au client et au serveur de négocier le protocole de la couche application à utiliser après l'établissement de la connexion TLS, ce qui optimise les performances en sélectionnant le protocole le plus approprié, tel que HTTP/3 dans ce contexte.

- *Supported Versions*

Cette extension indique les versions de TLS que le client supporte, permettant au serveur de sélectionner la version la plus sécurisée pour la communication.

- *Key Share*

Utilisée pour négocier les paramètres d'échange de clés pour l'établissement d'une session TLS. Cette extension permet au client et au serveur de convenir des paramètres cryptographiques pour sécuriser la communication tel que les groupes de clés et les méthodes d'échange.

- *Signature Algorithms*

Spécifie les algorithmes de signature pris en charge par le client pour authentifier les messages TLS. Cette extension permet au serveur de sélectionner un algorithme compatible avec celui du client pour garantir l'intégrité des données échangées.

- ***PSK** Key Exchange Modes*

Spécifie les modes d'échange de clés pré-partagées (PSK) pris en charge par le client pour sécuriser la connexion, permettant ainsi d'améliorer l'efficacité et la sécurité de la communication.

- *Record Size Limit*

Spécifie la taille maximale des enregistrements TLS acceptée par le client. Cette extension limite la taille des messages échangés pour des raisons de sécurité ou de contraintes réseau. Cela évite notamment les attaques par déni de service.

- *QUIC Transport Parameters*

Spécifie divers paramètres de transport QUIC, tels que la taille maximale des données, le délai d'attente maximal, etc. Cette extension permet de configurer et d'optimiser les performances du protocole QUIC pour la communication entre le client et le serveur.

- *Encrypted Client Hello*

Contient un Hello client chiffré qui est une fonctionnalité de TLS 1.3. Cette extension permet au client d'envoyer des informations sensibles de manière sécurisée dès le début de la négociation TLS, améliorant ainsi la confidentialité et la sécurité de la communication.

Lors de l'analyse du trafic réseau associé à l'application OneDrive, en plus des protocoles QUIC (HTTP/3) et DNS qui utilisent la couche de transport UDP, nous avons également identifié la présence de protocoles tels que **mDNS** (**M**ulticast **DNS**) et **SSDP** (**S**imple **S**ervice **D**iscovery **P**rotocol). Bien que ces protocoles ne soient pas directement liés à l'application OneDrive elle-même, ils jouent des rôles spécifiques dans le contexte des réseaux locaux.

En particulier, mDNS est utilisé pour la résolution de noms de domaine au sein des réseaux locaux, facilitant ainsi la découverte et la connexion entre les appareils sans nécessiter de serveur DNS central. D'autre part, SSDP est utilisé pour la découverte de services sur un réseau local, principalement pour les appareils prenant en charge UPnP (Universal Plug and Play), permettant ainsi l'annonce et la découverte d'autres appareils et services disponibles sur le réseau local. La table I résume l'utilisation des protocoles au sein de l'application OneDrive.

TABLE I
UTILISATION DE CHAQUE PROTOCOLE

Protocole utilisé	**Utilisation (exemple)**
TCP	Connexion sécurisée à l'application (handshake)
QUIC (UDP)	Modification de fichier en temps réel par exemple
TLS	Transmission de données chiffrées (utilisé avec QUIC et TCP)
HTTP/2	Multiplexage / Compression d'en-tête / Serveur Push
mDNS	Résolution des noms de domaine au sein des réseaux locaux
SSDP	Découverte de services sur un réseau local

VI. CHIFFREMENT ET SÉCURITÉ

Le DNS ne nous semble pas sécurisé à première vue, nous avons observé que nos requêtes DNS n'utilsent **ni le port 443** (faisant référence à ***DNS-over-HTTPS***) **ni le port 883** (faisant référence à ***DNS-over-TLS***), mais utilisent plutôt des ports "standard" comme le **port 53**, qui est le port par défaut pour les requêtes DNS et le **port 55163** qui n'est autre qu'un port aléatoire pour les requêtes sortantes.

Lors de l'établissement de la connexion vers OneDrive, une requête TLS *client hello* est envoyée au serveur onedrive.live.com. Cette requête contient plusieurs extensions

(notamment *server_name*, *extended_master_secret*, *renegociation_info*, *supported_groups*, ...), mais l'extension pertinente pour déterminer la version TLS utilisée est ***supported_version***. Dans cette extension, le client indique les versions TLS qu'il prend en charge (dans notre analyse, TLSv1.2 et TLSv1.3). Le serveur répond ensuite avec un message *server hello*, indiquant la version TLS choisie parmi celles proposées par le client. Dans notre cas, la version choisie est **TLSv1.2**.

Dans les paquets sous TLS, notamment lors des échanges des messages *client hello* et *server hello*, les certificats peuvent être observés. Ces certificats sont délivrés par **Microsoft Azure RSA TLS Issuing CA 07** et sont signés numériquement avec l'algorithme de hachage **SHA-384** (**sha384WithRSAEncryption**). Leur validité est généralement d'environ **1 an**, comme indiqué par les champs *notBefore* et *notAfter* dans la couche TLS du paquet.

Lors de l'échange des messages *client hello* et *server hello* dans le cadre du protocole TLS, plusieurs méthodes de chiffrement sont proposées et sélectionnées pour sécuriser la communication. Initialement, le client envoie au serveur la liste des algorithmes de chiffrement qu'il prend en charge, tels que ***TLS_AES_128_GCM_SHA256***, ***TLS_CHACHA20_POLY1305_SHA256***, ***TLS_AES_256_GCM_SHA384***, ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256***, etc. En réponse, le serveur spécifie les méthodes de chiffrement qu'il a choisies parmi les propositions du client. Par exemple, le serveur peut sélectionner ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384***.
Cet algorithme utilise l'échange de clé **ECDHE (Elliptic Curve Diffie-Hellman)** lors de l'établissement de la session, puis chiffre en utilisant **AES256 en mode GCM**, et enfin applique le hachage en **SHA256** pour assurer l'intégrité des données échangées.

Lors de l'analyse du trafic UDP, il a été observé que dans certains cas, ce trafic semble être chiffré grâce au protocole QUIC (Quick UDP Internet Connections). QUIC, qui repose sur UDP, intègre un mécanisme de chiffrement pour garantir un échange sécurisé entre le serveur et le client.

VII. Application

Lors du transfert de nouveaux fichiers, nous avons remarqué que le volume de données échangées est généralement plus élevé que lors de la modification de fichiers existants. Cela s'explique par la nécessité de transférer l'intégralité du fichier lors d'un nouveau transfert. De même, lorsqu'un fichier est modifié par plusieurs utilisateurs, la synchronisation automatique entre eux est plus rapide lors de la modification d'un fichier existant que lors de l'ajout d'un nouveau fichier, car seule la partie modifiée du fichier doit être transférée.

Lorsque deux utilisateurs se trouvent sur le même réseau Wi-Fi ou Ethernet, nous avons constaté une réduction de la latence et une augmentation des vitesses de transfert. Cela est dû à l'évitement des serveurs relais, car l'application est capable de détecter la proximité des utilisateurs sur le même réseau local. En effet, nous avons observé une diminution du nombre de requêtes DNS et une amélioration de la vitesse de résolution de ces dernières. Cette amélioration provient très probablement du protocole mDNS utilisé, permettant la résolution des noms de domaine directement au sein du réseau local, sans nécessiter d'infrastructure supplémentaire par rapport à DNS (voir section V).

Lorsque les utilisateurs accèdent à leurs fichiers via l'application OneDrive, leurs requêtes sont généralement dirigées vers les clusters de serveurs Azure. Cette architecture permet une communication directe entre l'application OneDrive et les serveurs de stockage, ce qui favorise des performances optimales et une meilleure intégrité des données. Bien que des serveurs intermédiaires ou des relais puissent être utilisés dans certaines situations, notamment dans les réseaux où la communication directe est restreinte, l'architecture principale de OneDrive repose sur l'utilisation de clusters de serveurs Azure pour la gestion du stockage et des données.

Le volume de données échangées par OneDrive dépend de la taille et du nombre de fichiers lors du transfert ou de la modification. Les interactions entre utilisateurs sur le même réseau ajoutent également des échanges, selon la taille moyenne des fichiers et la fréquence des interactions. Bien que variable, mesurer ces échanges par minute permet de comparer l'impact des fonctionnalités sur le volume total des données échangées. La table II montre les résultats que nous avons obtenu pour les 4 fonctionnalités de base.

TABLE II
Volume de données échangées par fonctionnalité

Fonctionnalité	Volume de données échangées
Connexion de l'application	33.15 Mo/seconde
Déconnexion de l'application	6.19 Mo/seconde
Ouverture d'un fichier	15.58 Mo/minute
Création d'un fichier	15.58 Mo/minute

VIII. Conclusion

Cette analyse approfondie de l'application OneDrive marque la conclusion du projet 1 d'analyse de réseau dans le cadre du cours ***LINFO1341 - Computer Networks***. En explorant les divers aspects de la manière dont OneDrive échange des données en fonction des fonctionnalités utilisées et des interactions entre utilisateurs, nous avons obtenu un aperçu approfondi de son fonctionnement dans un environnement réseau.

References

[1] NAT: https://www.it-connect.fr/chapitres/nat-translation-dadresse/
[2] mDNS: https://en.wikipedia.org/wiki/Multicast_DNS
[3] SSDP: https://stormwall.network/knowledge-base/protocol/ssdp
[4] Whois: https://who.is/
[5] Lien Github (Traces + Scripts): https://github.com/GGisOnline/Network-Projects
[6] Azure: https://learn.microsoft.com/fr-fr/azure/cloud-adoption-framework/get-started/what-is-azure