Analyse de l'application Skype

Victor Lepère
victor.lepere@student.uclouvain.be
61502000

Ygor Lausberg
ygor.lausberg@student.uclouvain.be
16452000

***Abstract*—Ce document rapporte les observations et conclusions émanant de l'analyse de l'application réseau Skype dans le cadre du cours LINFO1341 - Computer networks : information transfer, dispensé par Pr. Olivier Bonaventure.**

I. Introduction

Skype est une application de messagerie instantanée et de visioconférence détenue par Microsoft. Plusieurs de ses fonctionnalités ont été étudiées par le prisme de captures de paquets. Nous commencerons par discuter d'aspects techniques relatifs aux paquets capturés au sein des traces analysées tels que les noms de domaines résolus ou les protocoles de transports utilisés. Nous utiliserons ensuite notre analyse technique pour comprendre les mécanismes fondamentaux employés derrière les différentes fonctionnalités de l'application. Si le contexte semble manquant, le lecteur peut considérer l'observation comme générale (appel audio/vidéo à deux participants ou plus).

II. DNS

A. Protocole de transport

Les requêtes DNS sont conçues pour être résolues rapidement et ne contiennent que les quelques informations nécessaires à la traduction du nom de domaine demandé. Au sein des traces analysées, le protocole privilégié est donc UDP qui permet une transmission rapide des données et plus économe en ressources qu'un protocole tel que TCP. Il est donc idéal pour traiter ces requêtes de petites tailles.

B. Noms de domaines résolus

Dans le contexte d'un appel Skype (avec ou sans vidéo) entre deux utilisateurs connectés à un réseau domestique différent, les noms de domaines suivants sont résolus chez l'appelant au lancement de l'appel[1].

- *api.flightproxy.skype.com* : l'application envoie des requêtes DNS pour connaître les adresses IPv4 et IPv6 de ce nom de domaine qui est en réalité un alias pour *api-flightproxy-skype.trafficmanager.net*, lui-même un alias pour *b-flightproxy-euno-01-skype.cloudapp.net*. L'adresse IPv4 correspondante est 65.52.67.156. Ce nom de domaine ne possède pas de correspondance IPv6 [2]. Dans un autre essai, le dernier alias obtenu était *b-flightproxy-euwe-01-skype.cloudapp.net*. Ainsi, il semble que le premier nom de domaine peut être résolu en une adresse IP qui correspond à un serveur situé dans la région d'Europe du Nord (*euno*) tandis que le second à un serveur d'Europe de l'Ouest (*euwe*). Ceci permettrait de rediriger les utilisateurs vers le serveur hébergé dans leur région afin d'optimiser les performances et la disponibilité des services, en réduisant les temps de latence et en évitant la surcharge des serveurs. Chaque région possède plusieurs serveurs identifiés par un numéro unique. Ici, par exemple, nous utilisons le premier serveur : 01. Ceci est également une manière de réduire la surcharge des serveurs. Cette sous-division de serveurs est aussi utilisée dans les prochains domaines explicités mais nous ne prendrons qu'un seul des serveurs en compte par simplicité.
  De plus, les noms de domaines semblent aussi suggéré qu'il s'agit d'une serveur proxy qui pourrait servir d'intermédiaire entre les participants de l'appel. Cette hypothèse est explorée plus rigoureusement dans la section VI.
  Enfin, le serveur autoritatif pour le nom de domaine est *cloudapp.net*, géré par Microsoft Corporation qui est aussi le propriétaire de l'application.
- *api3.cc.skype.com* : ce nom de domaine est un alias pour *api3-cc-skype.trafficmanager.net* qui est un alias pour *cc-ukso-11-skype.cloudapp.net*. L'adresse IPv4 associée est 52.114.88.102 et le serveur autoritatif est à nouveau *cloudapp.net*.
- *be.relay.skype.com* : il s'agit à nouveau d'un *Canonical Name* pour *beaz.relay.skype.trafficmanager.net*, *CNAME* pour *a-tr-skysc-euno-01.northeurope.cloudapp.azure.com* et la résolution IPv4 donne 20.202.1.192. Le serveur autoritatif est cette fois *northeurope.cloudapp.azure.com* qui est aussi géré par Microsoft.
  Le nom de domaine laisse penser qu'il s'agit d'un serveur relais qui pourrait être utilisé pour acheminer le trafic de données entre les utilisateurs de Skype. Cette hypothèse est explorée davantage dans la section VI.

L'ensemble des noms de domaines énumérés ci-dessus appartiennent à Microsoft. Lorsque l'un des participants raccroche et que l'appel prend fin, les noms de domaines ci-dessous sont résolus.

- *azscus1-client-s.gateway.messenger.live.com* : il s'agit d'un alias pour *azscus1-client-s.msnmessenger.msn.com.akadns.netcname*, alias pour

[1] Les DNS résolus chez l'appelé sont identiques.

[2] Ceci a été vérifié en aval au moyen de commandes telles que `nslookup`. Cette remarque s'applique d'ailleurs aussi aux autres noms de domaines énumérés.

ip.azscus1-client-s.msnmessenger.msn.com.akadns.net.
La correpondance IPv4 est 40.74.219.49. Le serveur autoritatif est *akadns.net* qui n'est pas géré par Microsoft mais par Akamai Technologies, une société américaine spécialisée dans la mise à disposition de serveurs de cache pour les entreprises.
Ce domaine est utilisé comme passerelle par Skype pour faciliter la communication entre les utilisateurs.

- *ic3.events.data.microsoft.com* : *CNAME* de *teams-events-data.trafficmanager.net* qui est à son tour un *CNAME* de *onedscolprdneu01.northeurope.cloudapp.azure.com*. L'adresse IPv4 du serveur est 20.50.73.9 et le serveur autoritatif est *northeurope.cloudapp.azure.com*, géré par Microsoft. Ce domaine permet à l'application de collecter des informations sur la qualité des appels et les performances des services.
- *consumer.entitlement.skype.com* : il s'agit d'un alias pour *sconsentit9.trafficmanager.net*, alias pour *sconsentit9.trafficmanager.net* avec comme adresse IPv4 associée 40.114.211.99 et comme serveur autoritatif *cloudapp.net*. Ce domaine est utilisé pour gérer les droits des utilisateurs (crédit, abonnements, ...).

Pour un appel utilisant la vidéo conférence ou dans des conditions WIFI différentes, les noms de domaines résolus sont identiques.

C. Adresses

Indépendamment du contexte, le *Opcode* des requêtes DNS capturées est systématiquement à 0. Les requêtes sont donc de type standard, i.e. un client envoie un nom et le serveur renvoie les données correspondantes. Il n'y a ainsi aucune requête demandant une actualisation de ces données.
La figure 1 représente la proportion des familles d'adresses IP demandées au sein des requêtes DNS. Ces proportions sont en réalités des proportions moyennes obtenues en effectuant les mesures sur l'ensemble des traces réseau dont nous disposons. Ce calcul a été rendu possible grâce à l'utilisation de *pyshark*.

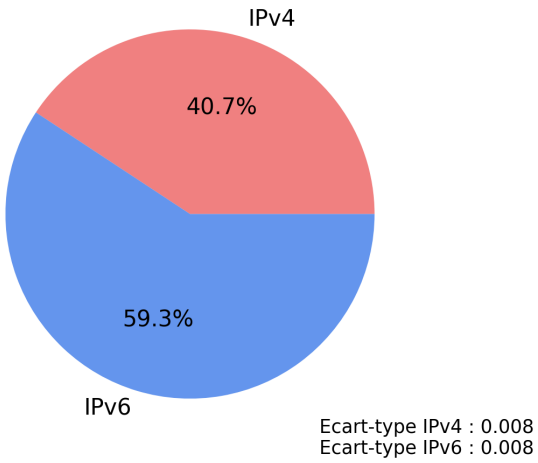

Fig. 1: Type d'adresses IP demandées

L'application semble donc légèrement préférer la version 6. Toutefois, comme vu précedemment, la plupart des noms de domaines résolus n'ont pas de corrrespondance IPv6. Dès lors, ce sont bien les adresses IPv4 qui sont utilisées pour établir une connexion entre le client Skype et les serveurs cloud de Microsoft. Cette connexion est principalement établie grâce à l'utilisation du protocole TCP et sécurisée avec TLS.

III. Couche réseau

A. NATs

Lorsqu'un utilisateur de Skype souhaite établir une communication avec un autre utilisateur afin de communiquer avec ce dernier, une connexion entre les deux doit être initiée. Cependant, la présence de NATs dans les réseaux privés peut masquer les adresses IP et les ports des clients. Ainsi, les NATs rendent possible l'utilisation d'adresses locales privées dans les réseaux locaux afin de pallier à l'épuisement des adresses IPv4 publiques.
Lorsque IPv4 est utilisé, l'application utilise le protocole *Session Traversal Utilities for NAT* (STUN) pour traverser les NATs. La version du protocole est celle définie dans RFC-5389 [3] et RFC-8489 [5]. Cette dernière rend obsolète plusieurs méthodes de la version initiale de STUN présentée dans RFC-3489 [2] (voir [6]) dont l'algorithme de caractérisation du comportement NAT (*symmetric*, *full-cone*, *restricted-cone*, *post-restricted-cone*) décrit dans la ressource suggérée dans l'énoncé du projet [7].
Au sein des traces analysées, la détection des NATs a été effectuée comme suit. L'application envoie d'abord une requête initiale au serveur STUN situé dans la partie public du NAT depuis l'adresse IP locale et observe si la réponse employe une adresse et un numéro de port publics différents (correspondant à l'attribut MAPPED-ADDRESS). Ce fut le cas dans nos traces, ce qui confirme la présence de NATs (pour deux participants situés sur des réseaux domestiques distincts).

IV. Couche transport

Outre DNS, d'autres protocoles ont été observés utilisant UDP pour échanger des paquets. Le premier est le protocole STUN discuté précédemment dans III-A. Les demandes de découvertes d'adresses IP publiques étant des échanges légers et spontanés, il n'est pas suprenant que ce protocole ait recourt à UDP.
Le protocole RTCP a aussi été identifé dans le trafic UDP. Ce protocole fournit des métadonnées statistiques sur la qualité du service (QoS). Plus précisément, il permet à l'application Skype de surveiller la qualité d'une session d'appel au moyen de plusieurs indicateurs tels que le nombre de paquets et d'octets transmis ou le *RTP Timestamp* utile à la synchronisation audio-video. Ces indicateurs permettent donc à l'application de détecter les pertes de paquets et les delais à solutionner par la suite.
Enfin, le dernier protocole detecté employant UDP est l'*Internet Control Message Protocol* (ICMP). Il s'agit d'un protocole de la couche réseau qui permet aux périphériques de diagnostiquer les problèmes éventuels sur le réseau. Dans

les traces analysées, les messages d'erreurs générés par ICMP permettaient principalement à l'application de déterminer si les données atteignaient ou non leur destination en temps voulu.

V. Chiffrement et sécurité

Lors d'un appel audio Skype classique, le *DO bit* est mis à 0 dans l'ensemble des requêtes envoyées aux serveurs DNS capturées. Ainsi, le client indique aux serveurs qu'il n'est pas en mesure de traiter les types d'enregistrement DNS relatifs à DNSSEC (*RSSIG*, *DNSKEY*, ...). Dès lors, bien que le *Authenticate Data* (AD) *bit* soit défini à 1 dans les réponses des serveurs, l'utilisation du DNS demeure insécurisée, car le client n'est pas capable de valider les signatures numériques via DNSSEC.
Par ailleurs, la version de TLS systématiquement négociée et utilisée par le client est TLS 1.2. Cette dernière est employée pour sécuriser les protocoles de transport TCP nécessaire à l'établissement des connexions vers les services cloud de Skype (II-B).
Le certificat envoyé par le serveur TLS (*api.flightproxy.skype.com*) pour prouver son identité est sans surprise émis par Microsoft Corporation. Sa date de validité expire le 2023-11-01 00:04:26 (UTC). Sa durée de vie est d'une annéee puisque sa validité a commencé aux mêmes heure et date l'année précédente. De plus, la suite de chiffrement choisie dans le *handshake* est TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.
L'algorithme d'échange des clés est donc basé sur le chiffrement *Elliptic Curve Diffie-Hellman Ephemeral* (ECDHE) avec des signatures *Rivest Shamir Adleman* (RSA) pour authentifier l'échange des clés. Le protocole d'échange de clés utilisé assure donc la propriété de *Perfect Forward Secrecy* (PFS) qui garantit que même si la clé privée du serveur est compromise, les clés utilisées pour les sessions précédentes ne seront pas compromises. La suite indique aussi l'utilisation du chiffrement *Advanced Encryption Standard with 256bit key in Galois/Counter mode* (AES 256 GCM) pour la confidentialité (cryptage) des données échangées entre le client et le serveur. Enfin, l'algorithme de hashage employé est le *Secure Hash Algorithm 384* (SHA384) pour authentifier les messages et assurer l'intégrité des données. Cette suite de chiffrement est toujours celle négotiée entre les clients Skype et les serveurs de Microsoft listés dans II-B.
Le trafic UDP quant à lui ne semble pas chiffré ni sécurisé dans l'ensemble des traces de réseau parcourues.

VI. Application

Cette section décrit concrètement comment Skype initialise et maintient une communication audio/vidéo entre les participants d'un appel. Dans la section II-B, nous avions évoqué la potentielle utilisation d'un serveur relais *be.relay.skype.com* pour acheminer les données entre les utilisateurs. En pratique, aucun paquet n'est envoyé ou reçu vers ce destinataire lors d'un appel. L'astuce est ailleurs et il faut à nouveau se pencher vers le protocole STUN présenté dans III-A.

Le trafic STUN déployé lors d'un appel Skype ne se limite pas qu'à la détection des NATs. En effet, lorsque deux clients Skype sont chacuns derrière un routeur NAT, il peut être difficile d'établir une connexion directe entre eux. Certaines techniques dites de *hole punching* permettent de trouver un chemin de communications direct entre les hôtes, mais peuvent échouer en fonction du type de NAT rencontré (e.g. symétrique). Il est alors nécessaire de passer par un service de relais pour rendre possible l'échange des paquets entre les participants de l'appel. Ce service est fourni ici par un protocole nommé *Traversal Using Relays around NAT* (TURN) et défini dans [4]. Ce protocole a la particularité de permettre à un client de communiquer avec plusieurs homologues en utilisant qu'une seule adresse relais. Le protocole TURN est étroitement lié au protocole STUN puisqu'il est en réalité une extension de ce dernier. Ainsi, les paquets TURN utilisent la plupart du temps le *header* des paquets STUN ainsi qu'un formattage identique. À cela s'ajoute des méthodes et attributs additionnels qui rendent possible la communication entre un hôte et des pairs. Cette communication s'effectue en pratique en passant par le serveur TURN, situé dans la partie publique de l'Internet (Fig. 2).

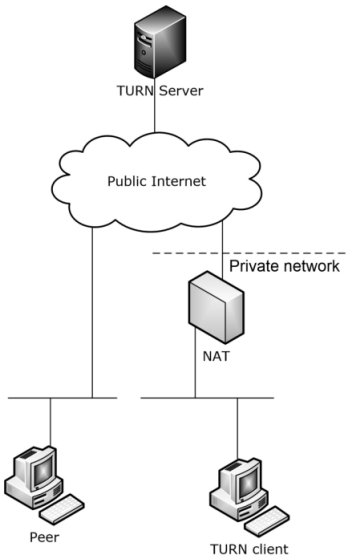

Fig. 2: © 2022 Microsoft Corporation

En ce qui concerne notre application, Microsoft utilise une légère variation du protocole TURN qui reprend son fonctionnement global tout en ajoutant de nouveaux attributs et en redéfinissant le comportement de certaines procédures. Ce protocole est connu sous le nom de MS-TURN et est donc une extension du protocole TURN. La référence [8] redirige vers la documentation officielle disponible sur le site de Microsoft (cette extension n'est pas exclusivement utilisée par Skype, mais aussi par d'autres services de la compagnie).
Pour analyser en profondeur le fonctionnement de MS-TURN, nous allons considérer un scénario simple d'une conversation audio entre deux utilisateurs Skype. Nous l'étendrons ensuite aux autres cas. Lorsque l'appelant lance l'appel et que la sonnerie retentit chez l'appelé, un message de type

Allocate request est envoyé au serveur TURN. Cette requête est authentifiée par le serveur qui fournit en réponse une adresse IPv4 publique allouée (et un port) via un message de type *Allocate response*. Cette adresse, ce port et le protocole de transport choisi pour échanger les données (UDP ou TCP) constituent la *transport address*. Le serveur communique aussi l'adresse allouée de l'appelé (XOR-PEER-ADDRESS) à l'appelant au sein d'un message dit de *Data Indication* [3]. L'appelant peut alors envoyer un message de type *Send request* au serveur TURN avec comme valeur associée à l'attribut DESTINATION-ADDRESS l'adresse allouée de l'appelé. D'une part, cette requête permet d'indiquer que toutes les données envoyées sur la *transport address* devront être relayées vers la destination spécifiée. D'autre part, des permissions sont accordées de telle sorte que les données arrivant sur la *transport address* envoyées par l'appelé soient relayées vers l'appelant. Une fois que l'appelant a décroché, il peut récupérer son adresse allouée (*Allocate request*) et établir une connexion avec l'appelant (*Send request*). Toutefois, la *transport address* est toujours considérée comme l'adresse de relais principale entre les deux utilisateurs vers laquelle la plupart de paquets sont envoyés.
Les participants peuvent dès lors transmettre leur voix (numérisée via un codec) en envoyant des paquets vers la *transport address* selon le protocole de transport utilisé. MS-TURN supporte aussi bien UDP que TCP mais TCP requiert un *framing* supplémentaire. De plus, le transport TCP est automatiquement converti en UDP par le serveur TURN lors du relais. UDP est donc privilégié par Skype. De plus, ce protocole permet d'éviter la latence lors de l'appel, au risque de potentiellement perdre des données lors de l'échange (compensable facilement si la perte est sur un laps de temps court). Si un utilisateur raccroche et met fin à la transmission, une requête *Allocate Request* avec un *lifetime* nul est envoyé au serveur TURN qui désalloue alors l'adresse publique précédemment accordée.

L'utilisation de la transmission vidéo dans un appel Skype n'altère pas le mécanisme présenté ci-dessus. Par contre, le volume des données envoyées via UDP au serveur augmente considérablement. La figure 3 compare le nombre de bytes capturés du point de vue de l'appelant au sein de deux appels : l'un utilisant la transmission audio uniquement, l'autre utilisant la transmission audio et vidéo. Ces derniers ont été effectués dans les mêmes conditions (l'appelé décroche vers 10 secondes et l'appelant raccroche vers 50 secondes). On observe que le trafic de données est près de 30 fois plus important lorsque la vidéo est utilisée. A noter qu'outre le trafic STUN et UDP, les participants établissent aussi en parallèle pour chaque appel des connexions par TCP encryptées via TLS aux noms de domaines listés dans II-B. Parmi les plus contactés, on retrouve les serveurs correspondants aux adresses 20.101.67.88[4] (*api.flightproxy.skype.com*) et 40.74.219.49 (*azscus1-client-s.gateway.messenger.live.com*). Le premier est un serveur proxy permettant de contourner les pare-feux et les restrictions de réseau pour garantir une communication fluide entre les clients et les serveurs de Skype. Le second gère tous les aspects liés à la messagerie de l'utilisateur (statut en ligne, contacts, ...).

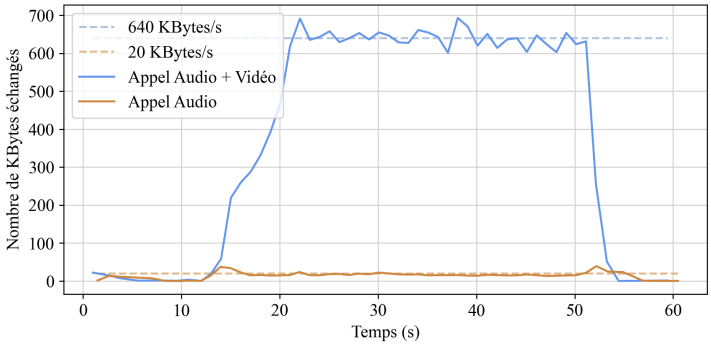

Fig. 3: Comparatif du trafic.

Qu'en est-il si plus de deux participants prennent part à la visioconférence ? Chaque session TURN utilise la version *MULTIPLEXED TURN over UDP and TCP* de MS-TURN. MULTIPLEXED TURN est une extension du protocole TURN qui permet à un serveur TURN de multiplexer le trafic de plusieurs clients TURN sur le même port UDP ou TCP, correspondant à la *transport address*. En pratique, cette extension est déjà utilisée même si la conversation n'implique que deux participants. Lors d'un appel de groupe, les mécanismes utilisés sont identiques à ceux d'un appel simple à l'exception que la *transport address* est cette fois une adresse publique fournie et détenue par Microsoft. Les participants envoie alors chacun une *Send request* à cette adresse via le serveur TURN pour s'y connecter et y transmettre les paquets contenant leur transmission audio ou vidéo.

Le protocole TURN n'est pas systématiquement utilisé au sein des fonctionnalités de l'application. Si les utilisateurs se trouvent sur le même réseau WiFi, alors il n'est pas nécessaire et l'échange se fait directement à travers le réseau local et les adresses privés.

Pareillement, le protocole TCP combiné à TLS est d'application pour le service de messageries instantanées afin de se connecter aux services cloud de Skype. Il assure que les messages sont livrés en entier et sans erreurs de manière sécurisée, à défaut d'être exposé à une plus grande latence (qui est évidemment moins cruciale dans ce contexte).

ANNEXES

Le dépot git recueillant l'ensemble des traces de réseau et des scripts qui ont servi à la rédaction de ce document est accessible à l'adresse https://github.com/Lopiuy/LINFO1341_P1.

[3]Les noms utilisés ici correspondent à ceux de la documentation officielle [8]. Ils peuvent différer de ceux assigner par Wireshark.

[4]Cette adresse est différente de celle rapportée dans II-B car il s'agit de l'adresse associée à la région d'Europe de l'Ouest.

REFERENCES

[1] O. Bonaventure, ”Computer Networking : Principles, Protocols and Practice”, 3rd edition, Université Catholique de Louvain, 2011-2021, https://beta.computer-networking.info/syllabus/default/index.html#.

[2] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, ”STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, RFC 3489, DOI 10.17487/RFC3489, March 2003, https://www.rfc-editor.org/info/rfc3489.

[3] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, ”Session Traversal Utilities for NAT (STUN)”, RFC 5389, DOI 10.17487/RFC5389, October 2008, https://www.rfc-editor.org/info/rfc5389.

[4] Mahy, R., Matthews, P., and J. Rosenberg, ”Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)”, RFC 5766, DOI 10.17487/RFC5766, April 2010, https://www.rfc-editor.org/info/rfc5766.

[5] Petit-Huguenin, M., Salgueiro, G., Rosenberg, J., Wing, D., Mahy, R., and P. Matthews, ”Session Traversal Utilities for NAT (STUN)”, RFC 8489, DOI 10.17487/RFC8489, February 2020, https://www.rfc-editor.org/info/rfc8489.

[6] Wikipedia contributors, ”STUN,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=STUN&oldid=1140524920 (accessed April 1, 2023).

[7] https://www.cisco.com/c/dam/en_us/about/ac123/ac147/archived_issues/ipj_7-3/ipj_7-3.pdf

[8] https://learn.microsoft.com/en-us/openspecs/office_protocols/ms-turn/9e434b27-eb13-4249-b031-2d15c3835c8b