Ica2Tcp : Un proxy SOCKS pour *Citrix*

Hugo Clout
hugo.clout@synacktiv.com

Synacktiv

Résumé. *Ica2Tcp* est un outil programmé en *C* permettant de proxyfier n'importe quelle connexion TCP au travers d'une connexion "bureau à distance" de *Citrix* (ICA) en utilisant les *canaux virtuels* personnalisés d'ICA. Il est à ICA ce que l'option `-D` est à la commande `ssh` d'*OpenSSH* : il permet d'exposer une interface SOCKS5 sur un port local qui est redirigé à travers la connexion ICA établie jusqu'au serveur distant. Toute commande SOCKS envoyée sur le port local entraîne donc l'établissement d'une connexion depuis le serveur distant vers l'hôte demandé.
Lien de l'outil : `https://github.com/synacktiv/ica2tcp`

1 Introduction

1.1 Définitions

Citrix [1] : Éditeur logiciel proposant notamment des solutions de virtualisation et d'accès à distance.

ICA : Protocole applicatif propriétaire de *Citrix* utilisé par les connexions entre les clients et les serveurs distants dans une infrastructure d'accès à distance (équivalent à RDP chez *Microsoft*).

1.2 Le besoin

Dans une situation où nous contrôlons à distance un serveur *Citrix* par une connexion ICA, nous souhaiterions parfois pouvoir utiliser ce serveur comme pivot pour atteindre le réseau auquel il est connecté depuis notre machine locale. Or, il n'est pas toujours possible d'ouvrir une connexion arbitraire entre ce réseau et le nôtre. Il serait donc intéressant d'exploiter la connexion ICA, autorisée et déjà établie.

Par exemple, certains audits de sécurité sont réalisés à distance, le client fournissant un accès à une de ses applications ou machines *Citrix*. Même si l'on peut contrôler cette machine, elle ne dispose en général pas des nombreux outils nécessaires à la réalisation de l'audit (outils de scan réseau, proxy d'interception, etc.). Nous souhaiterions alors utiliser nos propres outils sur notre machine, en redirigeant leur trafic réseau vers le serveur distant, comme il est par exemple possible de faire avec la commande

```
ssh -D port user@host
```

C'est ce que propose l'outil *Ica2Tcp*.

1.3 État de l'art

Comme mentionné précédemment, des outils similaires existent déjà pour d'autres types de connexions. On peut par exemple citer :

— pour les connexions SSH : l'option `-D` d'*OpenSSH* [12] ;
— pour les connexions RDP : *Rdp2Tcp* de N.Collignon [9], ou *SocksOverRDP* de *Nccgroup* [11].

Néanmoins, au moment du commencement des travaux, il n'existait pas de solution fonctionnelle supportant les connexions *Citrix* et répondant précisément au besoin évoqué plus haut.

2 L'outil

2.1 ICA et les *canaux virtuels*

De nombreux éléments sont nécessaires pour la mise en place d'une infrastructure *Citrix*. Cependant, une fois la session à distance ouverte, nous pouvons nous concentrer sur les éléments suivants :

— une machine locale sur laquelle est installé le logiciel client *Citrix* (*Citrix Workspace App* [3], ou *Citrix Receiver* [2]) ;
— une machine distante sur laquelle est installé le logiciel serveur *Citrix* (*Virtual Delivery Agent* - VDA [8]) ;
— la connexion ICA directe entre ces deux machines.

ICA est un protocole appartenant à la couche présentation du modèle OSI et qui peut fonctionner indépendamment de la couche de transport sous-jacente. Il est basé sur l'utilisation de *canaux virtuels* [4], qui permettent de multiplexer les différents flux de la session (audio, vidéo, clavier, souris, presse-papiers, etc.) dans une seule connexion.

Citrix permet aux développeurs de mettre en place des *canaux virtuels* personnalisés pour permettre le support et l'intégration de nouvelles fonctionnalités dans l'infrastructure *Citrix*.

Un SDK est ainsi mis à disposition pour développer ces *canaux virtuels* en C [6, 7]. Ces derniers sont composés (figure 1) :

— d'un pilote client sous la forme d'une bibliothèque partagée, dont les différentes fonctions sont appelées par le logiciel client *Citrix* ;
— d'un exécutable côté serveur, qui peut utiliser les fonctions de l'API *Citrix* pour la gestion des *canaux virtuels*.

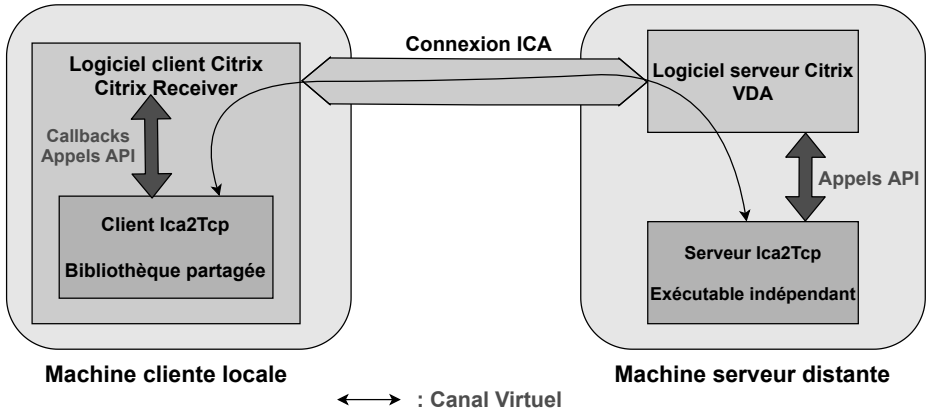

Fig. 1. Architecture d'une connexion ICA et d'un canal virtuel

2.2 Fonctionnement de l'outil

Mise en place d'un canal virtuel Côté client comme serveur, une partie du code permet la mise en place du *canal virtuel*. Les spécifications sont fournies dans la documentation du SDK : le pilote client doit obligatoirement définir certaines fonctions qui sont ensuite appelées par le client *Citrix* à différents moments du cycle de vie du *canal virtuel* (ouverture, configuration, fermeture, réception de données, etc.), et il peut envoyer des données par un appel à une fonction de l'API *Citrix*. C'est la partie serveur d'*Ica2Tcp* qui initie l'ouverture du *canal virtuel*. Là aussi, l'API *Citrix* expose des fonctions permettant l'envoi et la réception de données, ainsi que la gestion du *canal virtuel*.

Une fois que ce dernier est ouvert, on dispose d'une connexion garantissant la remise et l'ordre d'arrivée des paquets envoyés, et ce quel que soit le protocole de la couche transport sous-jacent.

Le reste du code expose une interface SOCKS sur la machine cliente, et gère le contrôle de flux (qui n'est pas nativement géré par ICA) ainsi que le multiplexage de plusieurs connexions dans le canal virtuel.

Interface SOCKS Une partie de la *RFC-1928 - SOCKS Protocol Version 5* [10] est actuellement implémentée par *Ica2Tcp* :

- protocol version : 0x05-Version 5 ;
- method selection : 0x00-No authentication required ;
- commands : 0x01-CONNECT ;
- address type : 0x01-IPv4, 0x03-Domain Name, 0x04-IPv6.

Les connexions TCP vers les destinataires des commandes SOCKS sont établies depuis le serveur *Citrix* distant puis relayées jusqu'aux clients par le canal virtuel.

Multiplexage L'outil fonctionne avec une logique asynchrone et un seul fil d'exécution. C'est une petite machine à états qui implémente un mircro protocole et utilise des fonctions de rappel, des sockets non bloquants, et des utilitaires comme `epoll` pour gérer le multiplexage de toutes les connexions SOCKS ouvertes dans le canal virtuel.

Contrôle du flux Enfin, comme le contrôle de flux n'est pas implémenté nativement par ICA dans les canaux virtuels, un système de mise en file et de fenêtrage similaire à celui de TCP a été mis en place afin de s'assurer que les paquets sont transmis à un rythme permettant leur bon traitement et évitant toute perte d'information.

3 Utilisation et démo

3.1 Pré-requis

À l'heure actuelle, le client *Ica2Tcp* n'existe que pour *Linux*. Une version *Windows* est néanmoins prévue. La partie serveur est quant à elle nécessairement sous *Windows* car le VDA *Citrix* pour *Linux* est bien moins fréquemment rencontré et le SDK pour celui ci est très récent (octobre 2021). Les deux composants sont compilables depuis *Linux*, avec *GCC* et *MinGW*, les instructions précises étant disponibles avec les sources de l'outil sur *Github* (voir lien dans l'abstract).

3.2 Exécution

Pour utiliser l'outil, il faut au préalable copier le pilote client dans les dossiers d'installation du logiciel client *Citrix* et demander son chargement dans un fichier de configuration. De plus amples explications sont aussi disponibles sur *Github*.

Il suffit ensuite de déposer l'exécutable sur le serveur *Citrix* et de l'exécuter depuis le contexte d'une session ICA ouverte (figure 2) car l'ouverture du *canal virtuel* nécessite l'existence d'une connexion ICA.

Dès lors le port 33556 (par défaut, mais paramétrable) est en écoute sur 127.0.0.1 (également paramétrable) sur la machine cliente et prêt à recevoir des requêtes SOCKS5. Il suffit donc de configurer ce port comme hôte SOCKS dans nos outils, ou, si ceux-ci ne proposent pas de support natif pour ce type de proxy, d'utiliser un outil comme *TSocks* [14] ou *Proxychains* [13].

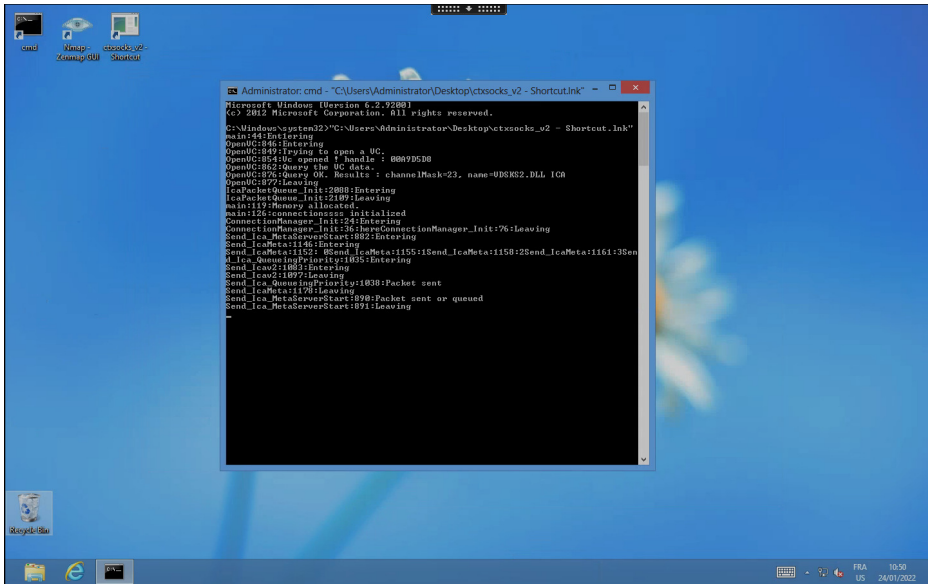

Fig. 2. Lancement de l'exécutable serveur.

4 Détection et prévention

4.1 Virtual Channel Allow Lists

Le filtrage réseau ne constitue pas une méthode efficace pour empêcher l'utilisation d' *Ica2Tcp*. En effet, tous les *canaux virtuels* étant transportés par la même connexion TCP, il n'est pas possible de les bloquer sans bloquer toute la connexion *ICA*.

En revanche, um mécanisme de *Citrix*, les *virtual channel allow lists* [5], peut empêcher d'utiliser *Ica2Tcp*. Cette option est disponible depuis juin 2020 (*Citrix Virtual Apps and Desktops 2006*) et est activée par défaut depuis octobre 2021 (*CVAD 2109*). Lorsqu'elle est active, l'utilisation des *canaux virtuels* est limitée à ceux de Citrix, et aux canaux personnalisés explicitement autorisés dans une liste. Cette liste peut se définir par *GPO* ou à travers le *Citrix Studio*, et contient le nom de chaque canal approuvé ainsi qu'une liste d'exécutables autorisés à l'ouvrir.

4.2 Journalisation

L'apparition des *virtual channel allow lists* est accompagné de la journalisation de plusieurs événements concernant les *canaux virtuels*. Ainsi, les tentatives d'ouverture (réussies ou non) de canaux personnalisés

laisseront des tracces dans les journaux des serveurs sur lesquels elles ont été effectuées, facilitant ainsi la détection de l'utilisation d'*Ica2Tcp*.

Références

1. Citrix. `https://www.citrix.com/`.
2. Citrix. Citrix Receiver. `https://www.citrix.com/en-gb/downloads/citrix-receiver/`.
3. Citrix. Citrix Workspace App. `https://www.citrix.com/fr-fr/downloads/workspace-app/`.
4. Citrix. ICA Virtual Channels. `https://docs.citrix.com/en-us/citrix-virtual-apps-desktops/technical-overview/virtual-channels.html`.
5. Citrix. Virtual Channel allow list. `https://docs.citrix.com/en-us/citrix-virtual-apps-desktops/policies/reference/ica-policy-settings/virtual-channel-allow-list-policy-settings.html`.
6. Citrix. Virtual Channel SDK for Linux. `https://developer-docs.citrix.com/projects/workspace-app-for-linux-virtual-channel-sdk/en/latest/`.
7. Citrix. Virtual Channel SDK for Windows. `https://developer-docs.citrix.com/projects/workspace-app-for-windows-virtual-channel-sdk/en/latest/`.
8. Citrix. Virtual Delivery Agent. `https://docs.citrix.com/fr-fr/citrix-virtual-apps-desktops/install-configure/install-vdas.html`.
9. Nicolas Collignon. Rdp2Tcp. `http://rdp2tcp.sourceforge.net/`.
10. Marcus D. Leech. SOCKS Protocol Version 5. RFC 1928, March 1996.
11. Nccgroup. SocksOverRDP. `https://github.com/nccgroup/SocksOverRDP`.
12. OpenSSH. `https://www.openssh.com/`.
13. Proxychains. `https://github.com/haad/proxychains`.
14. TSocks. `http://tsocks.sourceforge.net/`.