Fiche technique

# Stéganographie : techniques

Bénoni Martin 

Degré de difficulté

**Que ce soit par VPN ou en utilisant HTTPs, crypter des informations permet d'éviter que tout le monde ne puisse les lire (à moins de casser le cryptage), mais attire aussi l'attention : un flux AH/ESP par exemple indique forcément l'utilisation de cryptage, ainsi que la convoitise car les informations chiffrées sont généralement sensibles et donc intéressantes.**

La stéganographie a pour avantage de faire transiter des informations (cryptées ou non) sans attirer l'attention car ces informations sont contenues dans d'autres informations classiques (textes, images, trames TCP, codes sources, morceaux de musique, ...).

Sa force est principalement basée sur deux idées simples : nos sens (œil, ouïe) ne sont pas capables de détecter d'infimes changements dans une image ou un son, et à priori nous ne savons pas à l'avance que tel fichier renferme de l'information cachée.

La stéganographie (venant du grec *steganos* – dissimulé et *graphein* – écrire) date de 1499 avec le traité Steganographie publié par TRITHEMIUS et a pris depuis diverses formes au fur et à mesure de son évolution et de l'ingéniosité de ses utilisateurs : encres sympathiques (lait ou l'urine au temps de la Rome Antique, sulfate de cuivre révélé par des vapeurs d'ammoniaque pendant la IInde G.M., partitions musicales (Gaspar Schott en 1650), écrits (code de Barn, lettres en Georges Sand et A. de Musset, ...), micropoints sur des microfils utilisés par les Allemands pendant la Seconde Guerre Mondiale, etc.

La stéganographie moderne cache les données dans les fichiers binaires et les modes de transport de ces informations sont le plus souvent les images (format *bitmap*) ou les fichiers audios. Nous commencerons donc ici par décrire ces deux techniques, puis nous continuerons avec d'autres un peu plus complexes et plus ou moins utilisées afin que vous ayez un aperçu le plus étendu possible des possibilités de la stéganographie de nos jours.

## Cet article explique ...

- Ce que l'on appelle la stéganographie : le fonctionnement, les techniques utilisées et un état des lieux de cette technologie à l'heure actuelle.

## Ce qu'il faut savoir ...

- Aucun pré-requis particulier car l'article est suffisamment généraliste pour convenir à des personnes n'ayant qu'un vernis technologique. Cependant des notions de réseaux sont quand même recommandées pour aborder la deuxième partie (stéganographie sur d'autres supports qu'audio et images).

## La stéganographie sur images

Le principe est simple et connu sous le nom de LSB (*Least Significant Bit*). Une image est formée d'une succession de pixels (comme un texte en français, mais au lieu des caractères latins, nous avons ces pixels). Prenons par exemple une image avec une palette de 256 couleurs (donc chaque couleur est codée sur 8 bits).

Pour passer un message, il suffira de changer le dernier bit (celui le plus à droite, appelé Least Significant Bit). Le changement est imperceptible pour l'œil humain, mais une machine fera clairement la différence entre 0 et 1. Voyons maintenant en pratique comment cela marche ! La Figure 1 présente le codage d'une image plus complexe (16,8 millions de couleurs), soit donc un codage sur 24 bits (par pixel). Les 8 premiers pixels codent le rouge, les 8 suivants le vert, et les 8 derniers le bleu, c'est ce qui est connu sous le nom de RGB – *Red Green Blue* – en anglais. Dans ce cas, nous aurons 3 bits faibles (encadrés en rouge dans la Figure 2) par pixel. La pratique montre que dans ce cas, le changement est encore plus imperceptible pour l'œil. À gauche nous avons les 3 séries codant un pixel quelconque de notre image (l'exemple ici est Rouge à 242, Vert à 147 et Bleu à 19), qui deviennent respectivement (Rouge: 243, Vert : 146 et Bleu : 18) lors du rajout de l'information à cacher (dans notre cas, nous avons rajouté les 3 bits d'information suivants : 1 0 0).

- *Couleur avant* : 242 (rouge)+ 147 (vert)+ 19 (bleu),
- *Couleur après* : 243 (rouge)+ 146 (vert)+ 18 (bleu).

Ensuite, on fait cela de gauche à droite et de haut en bas dans le texte, pixel par pixel, tant qu'il restera des informations à cacher. Maintenant que nous avons montré la théorie, présentons quelques exemples d'outils permettant de faire cela automatiquement.

**Tableau 1.** *Comparaison technique des quelques produits de stéganographie*

| | Stools 4 | HIDESeek 5.0 | Gift It Up | Invisible Secrets | Hide4PGP |
|---|---|---|---|---|---|
| Stégano Audio | WAV | | | WAV | WAV, VOC |
| Stégano Images | GIF, BMP | | | JPEG, BMP, PNG | BMP |
| Stégano Texte | | | | HTML | |
| Algos de cryptage | IDEA, DES, 3DES, MDC | IDEA | | Blowfish, Twofish, RC4, Cast128, Gost | PGP |
| Systèmes d'exploitation | Windows | Windows | Windows | Windows | Windows |

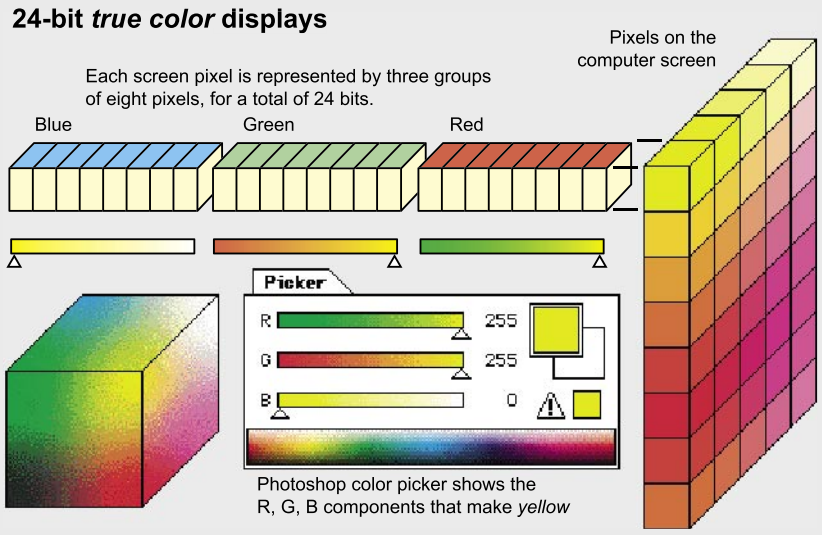


**Figure 1.** *Détail du codage d'une image sur 24 bits (extrait de http://www.webstyleguide.com)*

## Exemples pratiques

Un premier exemple de stéganographie sur image est montré à la Figure 4 avec l'outil Stools4 (il existe aussi Invisible *Secrets* ou *Gift-it-up*). Des outils plus avancés comme *Digital Invisible Ink Toolkit* (outil développé en Java tournant sous Windows/Linux/Mac OS) permettent de sélectionner les pixels (voir Figure 3) dans lesquels cacher les informations à des frontières, cette technique s'appelle filterfirst car nous balayons d'abord l'image pour trouver les pixels intéressants, puis nous enregistrons les données.

## Conclusion

La stéganographie sur des images présente les avantages suivants :

- Les modifications visuelles sont indécelables tant qu'on respecte la *règle des 20-25%* (il ne faut pas que la taille des informations cachées dépasse 20-25% de celle du fichier initial),
- Rapide et facile à mettre en œvre comme l'ont montré les exemples précédents,
- Permet de cacher beaucoup d'informations dans un fichier à peine 4-5 fois plus grand,
- Pas de modification du fichier initial.

Mais les inconvéneits suivants :

- Assez facile à casser pour des experts,
- Forte sensibilité à la moindre altération (compression, mise en page, rotation, ...).

# La stéganographie sur fichiers audio

Les fichiers audios sont encore une meilleure manière de cacher de l'information grâce à leur taille.

En effet, même au format *MP3* qui est l'un des plus répandus et ayant une des meilleures compressions, un fichier de 3-6 Mo est tout à fait usuel. Si nous respectons toujours la règle d'or (volume des informations à cacher < 20-25% de la taille du fichier initial), cela nous fait autour de 1Mo de données cachées !

Il y a plusieurs techniques de stéganographie sur des fichiers audio :

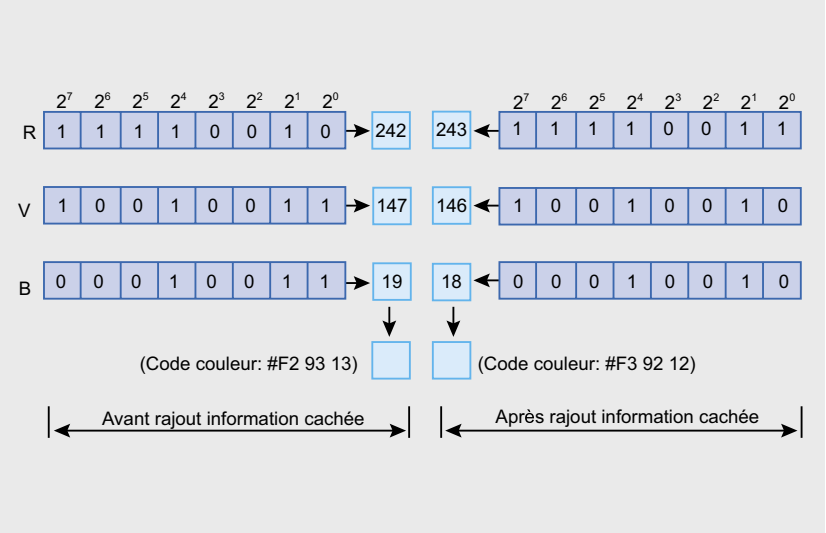


**Figure 2.** *Informations cachées dans une image codée sur 24 bits*

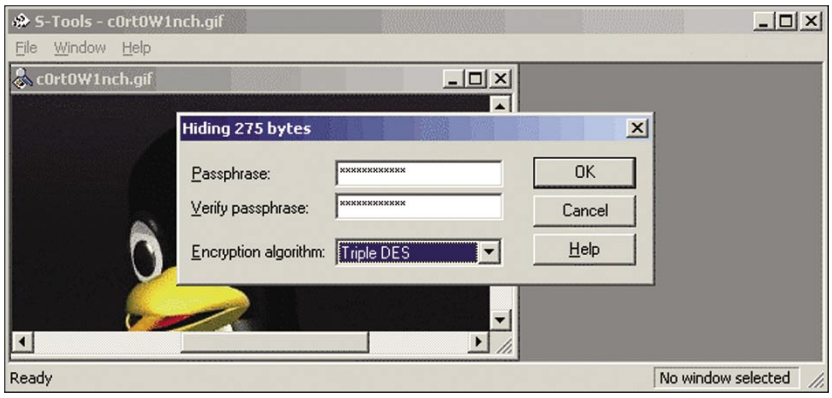


**Figure 3.** *Exemple de stéganographie sur image avec Stools4 (cache de données)*

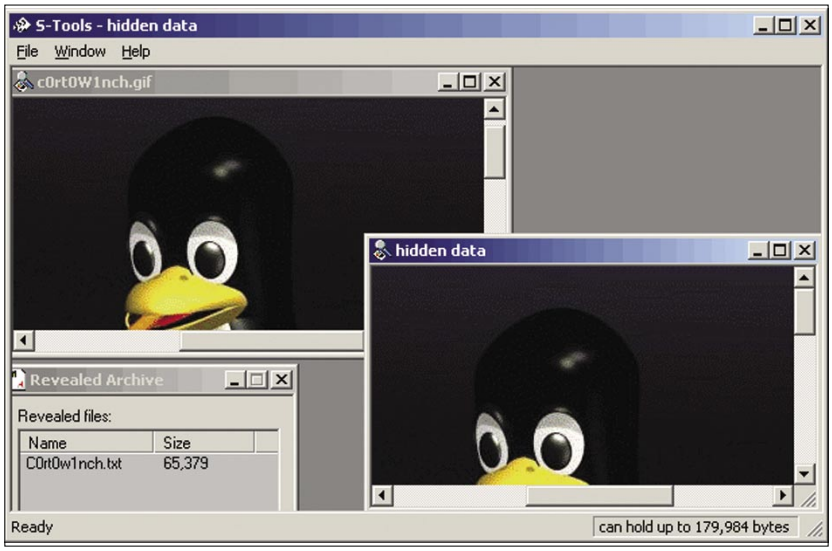


**Figure 4.** *Exemple de stéganographie sur image avec Stools4 (décache de données)*

- LBE (*Low Bit Encoding)* : cela ressemble à la technique LSB pour les images,
- SSE (*Spread Spectrum Encoding* ) : on rajoute du bruit aléatoire (dans lequel sera le message) à la mélodie originale,
- EDH – *Echo Data Hiding* : il existe souvent dans les mélodies un écho associé au son original. La distance séparant ces deux sons peut être utilisée pour y coder des informations,
- *Masque de perception* : on cache un son derrière un son plus puissant mais de même intensité.

Avec des fichiers audio, nous n'allons plus jouer sur les palettes de couleurs comme avec les images, mais sur les pistes de son qui peuvent s'inverser.

Quelques exemples de logiciels permettant de mettre en œuvre ce genre de techniques sont MP3-stego ou Stools4 (mais qui ne gère que les fichiers au format *wav*). L'avantage certain de la stéganographie sur fichiers audio sur celle sur des images est que la quantité d'informations qui peuvent être cachées y est plus importante grâce à la taille du support, ainsi que les techniques de dissimulation qui sont plus nombreuses.

## La stéganographie sous diverses formes

Nous avons vu précédemment les deux supports les plus classiques de la stéganographie : les images et les fichiers audio.

Mais cette technique étant l'art de cacher de l'information dans d'autres informations, il est possible d'utiliser cette idée dans le domaine des réseaux informatiques, principalement via des canaux cachés ou tunneling sur presque toutes les couches du modèle OSI (IP, TCP, ICMP, HTTP, DNS, ...) comme nous allons le voir ci-dessous.

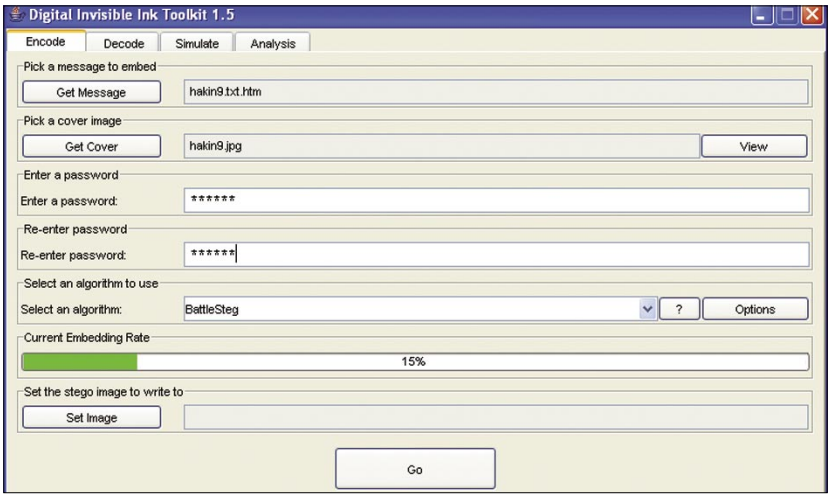


**Figure 5.** *L'outil DIIT en action (1)*

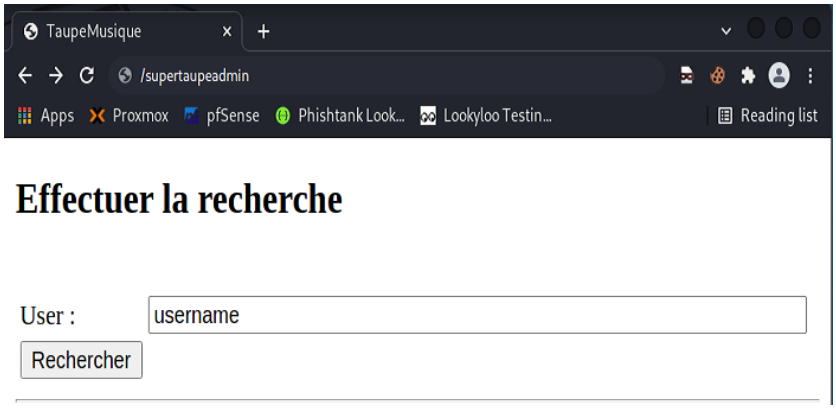


**Figure 6.** *L'outil DIIT en action (2)*

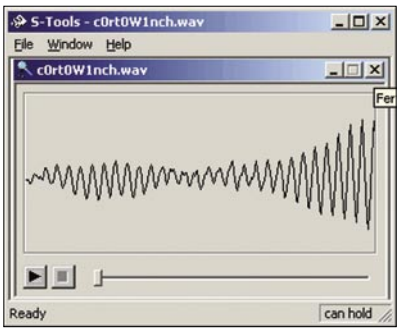


**Figure 7.** *Exemple de stéganographie sur fichier audio avec Stools4 (choix du fichier audio)*

### Canaux cachés IP

Dans ce cas, on utilise généralement le champ Identification (16 bits) pour cacher les informations (Figure 10).

Comme un caractère ASCII est codé sur 8 bits maximum (7 pour les caractères ASCII non étendus) alors que le champ ID de IP est sur 16 bits, il faut multiplier les codes ASCII par 256 avant de les transformer en binaire, ce qui nous donnera un code final en binaire sous 16 bits.

### Canaux cachés TCP

Dans ce cas, on peut utiliser principalement les champs :

- Numéro de Séquence (le principal inconvénient est de n'avoir que 32 bits de disponibles par connexion TCP mais par contre c'est indétectable car le choix du ISN est alétoire par définition),
- Numéro d'accusé de réception,
- Ou même le Bourrage (pour ce champ par contre, c'est très facilement détectable car le bourrage est très souvent une série de bits à 0 ...

Cependant, comme ces paquets transitent sur le réseau, ils peuvent subir des altérations dans leur périple : altération du au réseau, champ(s) modifié(s) par un routeur/ firewall/ ... . sans compter que la modification des bits de bourrage est très facilement repérable, étant très souvent tous à 0 !

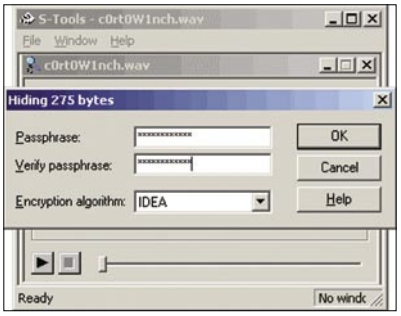


**Figure 8.** *Exemple de stéganographie sur fichier audio avec Stools4 (cache de données)*

## Canaux cachés ICMP

Dans ce cas, on peut surtout utiliser le champ Numéro de Séquence (le principal inconvénient est de n'avoir que 32 bits de disponibles).

Il est aussi possible d'utiliser des outils comme PTunnel (*Ping Tunnel*) qui permettent d'encapsuler des sessions TCP dans du flux ICMP (les très connus ICMP Echo & Request), ce qui est utile par exemple pour accéder à sa messagerie via un proxy en passant à travers un firewall bloquant tous les flux SMTP/POP/IMAP. Plus de détails sur cet outil sont disponibles sur la page d'accueil du site *http://www.cs.uit.no/~daniels/PingTunnel/*.

## Canaux cachés HTTP

Il est possible de cacher des informations dans des flux HTTP (principalement en-tête et corps des données), qui sont sant doutes parmi les plus utilisés, les moins filtrés et cependant les plus nécessaires dans une entreprise.

Dans l'en-tête, il est par exemple possible de cacher les informations directement dans des champs standard de HTTP comme User-Agent, de les coder selon l'alternance des balises (voir l'exemple ci-après de la stagénographie sur des fichiers HTML), ou encore de les dissimuler dans de faux champs. Cette méthode n'est cependant pas utilisable pour véhiculer de larges quantitées de données.

## Canaux cachés DNS

L'outil dns2tcp écrit par O. DEMBOUR & N. COLLIGNON, et disponible sur *http://www.hsc.fr/ressources/outils/dns2tcp/* permet d'encapsuler des sessions TCP dans des paquets DNS.

## Fichiers HTML

L'outil Deogol (téléchargeable sur *http://wandership.ca/projects/deogol/*) permet de cacher des informations dans des fichiers HTML, ce qui est fait par la manière de placer les attributs dans les balises HTML (jusqu'à 300 octets dans un container de 16Ko).

## Manipulations / substitutions de mots

Cette forme de stéganographie permet de cacher un message à l'intérieur d'un autre message.

Le moyen le plus commun est d'utiliser le spamming : un message spam passe en général directement à la poubelle, et étant donné le volume important à l'heure actuelle de spam sur Internet, il est d'autant plus

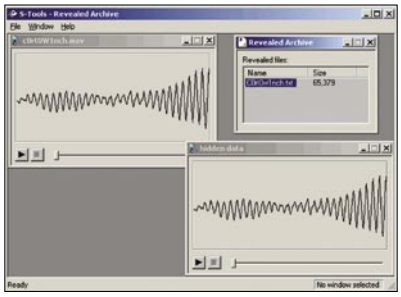

**Figure 9.** *Exemple de stéganographie sur fichier audio avec Stools4 (décache de données)*

facile d'y glisser quelques faux spams contenant des informations cachées. Cependant ce genre de messages a la facheuse tendance d'être souvent dépourvus de sens quand on les lit en entier, les rendant donc assez détectables par un œil averti.

## Rajout de données (EOF, en-têtes, ...)

Les données cachées consitent en un fichier image rajouté juste sous le marqueur EOF d'un fichier. Il est très

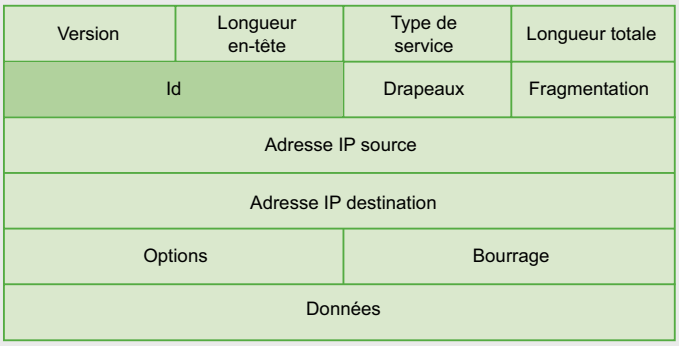


**Figure 10.** *Champs susceptibles de cacher des informations dans un en-tête IP*

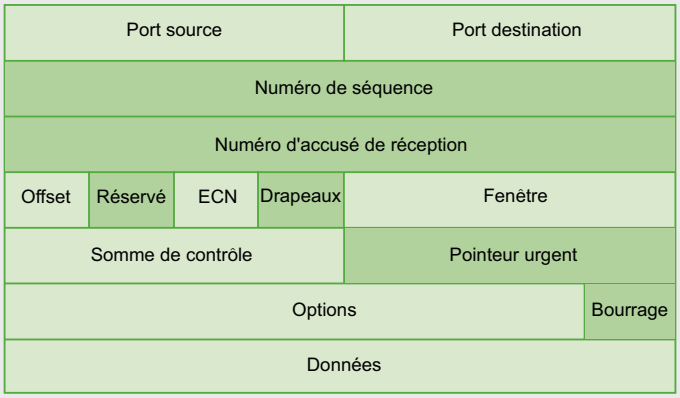


**Figure 11.** *Champs susceptibles de cacher des informations dans un en-tête TCP*

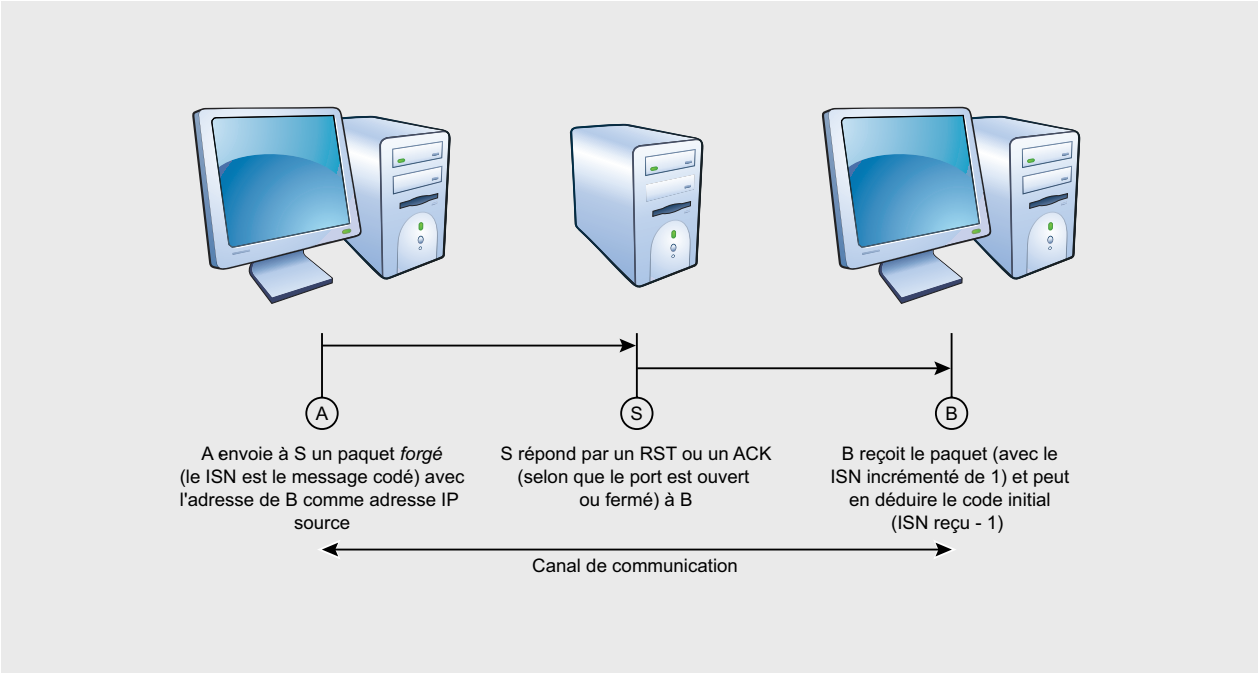


**Figure 12.** *Transmission de données stéganographiées par rebond TCP*

facile de rajouter des informations de cette manière, mais cela augmente cependant la taille du fichier, inscrit des données là où ne devrait pas en avoir (sous le marqueur EOF) et les données sont facilement visibles en ouvrant le fichier avec un éditeur de texte comme UltraEdit.

Avec des fichiers FAT (cependant plus vraiment utilisés de nos jours) sous Windows, on peut cacher des fichiers en les placant dans un répertoire ayant comme nom le caractère ASCII 255.

Avec des fichiers NTFS (c'est le plus utilisé de nos jours sur les plateformes Windows), on peut utiliser la fonctionalité *Alternate Data Streams* qui permet d'attacher un fichier A à un fichier B ... sans changer la taille apparente de A, et sans que B apparaisse dans l'explorateur de documents ou dans tout autre outil standard de NT.

Heureusement que des fichiers joints avec ADS ne peuvent pas être envoyés par email, ou utilisés implicitement par FTP ou WinZIP. Le virus W2K/Stream (ne fonctionne qu'avec les Windows 2000) utilisait cette fonctionalité pour se cacher ...

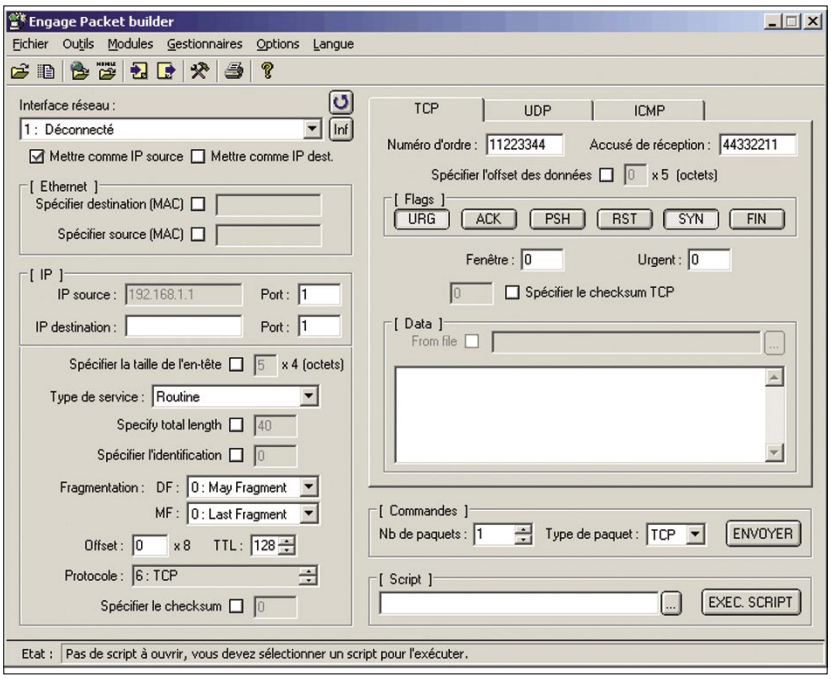


**Figure 13.** *Création manuelle d'un paquet TCP stéganographié*

## Conclusion

La stéganographie reste un vaste champ de recherches de nos jours, même si les techniques les plus utilisées et les plus connues utilisent sans conteste les fichiers images et les fichiers audio pour les non-informaticiens, tandis que les canaux cachés restent plus difficiles à mettre en place et ne sont pas à la portée des néophytes. ●

## À propos de l'auteur

L'auteur a travaillé dans le domaine de la sécurité et des réseaux depuis plus de 5 ans, en France et à l'étranger, pour des multinationales des domaines bancaire, télécoms et industrie. Il est actuellement chef de projet pour la société ArcelorMittal.