R4.08 - TD 1
Travailler dans un conteneur

Exercice 1 - Stéganographie

Introduction

Pour les besoins de ce 1er exercice, vous allez devoir compiler un programme de **stéganographie** dont le code source est sur Github. On va y aller par étape.

La stéganographie est un procédé permettant de dissimuler discrètement le contenu d'un fichier (le message) au sein d'un autre fichier (le support).

Le support est souvent une image, une vidéo, une séquence sonore, ou plus rarement un autre texte car l'intérêt de cette manipulation est que le fichier final, qui contient le message caché, semble totalement identique à son original. Ceci est possible sur des média qui acceptent une légère altération imperceptible à l'œil ou à l'oreille. Les images, sons et vidéos, se prêtent parfaitement à cet exercice.

Dans le cas présent, nous utiliserons une image PNG comme support.

Rendez-vous sur le dépôt du logiciel[1] :

`https://github.com/BigPapoo/steganography`

et regardez les consignes d'installation dans la section **Building**. N'installez rien pour le moment ! Nous allons travailler dans un conteneur et pas sur votre machine hôte.

Volume de travail

Vous allez travailler dans un volume Docker que vous nommerez **`r408`**.

Préparons le terrain à l'aide des commandes suivantes :

[1] Il s'agit d'un fork du projet **`7thSamurai/steganography`**.

```
docker volume rm r408        (pas d'inquiétude si vous avez une erreur ici)
docker volume create r408
```

Si vous avez une erreur indiquant que le volume est in use, ça signifie sans doute que quelqu'un avant vous a laissé un conteneur attaché à son volume r408. Le plus simple est d'arrêter et de supprimer ce conteneur :

```
docker ps -a                 (pour lister tous les conteneurs)
docker container rm <ID_DU_CONTENEUR_BLOQUANT>
```

Pour travailler dans votre volume, le plus simple est de passer par VSC, comme vous avez appris à le faire en BUT 1. Vous avez un document **TD 1 - Rappels VSC + Volumes** pour vous rafraîchir la mémoire. Ne consultez cette documentation que quand vous en aurez besoin (c'est indiqué plus loin).

Compilation

Le code est écrit en C++ (c'est indiqué en introduction du dépôt). Nous avons donc besoin d'un compilateur C++. Il se trouve que **gcc**, le compilateur C de Linux, est aussi un compilateur C++.

Pour cette expérimentation, nous allons donc choisir une image Docker contenant **gcc**. Par chance, il existe une image appelée **gcc** tout simplement, qui fera parfaitement l'affaire pour commencer.

Récupérez l'image **r408-gcc** qui est un clone de l'image **gcc** provenant du Docker Hub et placée sur le Hub de l'IUT pour les besoins de ce TD.

Si vous souhaitez travailler de chez vous, vous pourrez utiliser ces mêmes images en ajoutant **bigpapoo/** devant.

Lancez un conteneur en mode interactif (et sans **--rm**) sur une base d'image **r408-gcc**, en prenant soin de mapper le volume **r408** (option **-v** du **docker run**) vers le dossier **/work** du conteneur.

Voici maintenant les étapes à faire **DANS LE CONTENEUR** pour récupérer et compiler le code source du programme steganography :

- Déplacez-vous dans **/work**
- Clonez le dépôt du Github dans **/work** à l'aide de cette commande :

```
git -c "http.proxy=129.20.239.9:3128" clone https://github.com/BigPapoo/steganography
```

Pour chez vous, vous pourrez omettre le **`-c "http.proxy=..."`** qui précède la commande **`clone`**, car ça ne concerne que la situation particulière de l'IUT.

Une fois le dépôt correctement cloné dans votre conteneur, placez-vous dans le dossier **`steganography`** qui vient d'être créé par **`git clone`**, et suivez les instructions de compilation données dans la section **Building** en haut de page du projet dans Github.

❗ Vous devriez rencontrer un problème avec la commande **`cmake`** qui est introuvable. ❗

Il faut donc l'installer dans votre conteneur. Les composants logiciels (librairies, programmes divers) sous Linux, qu'on appelle aussi des **packages**, s'installent en utilisant la famille de commandes **`apt`**, et notamment **`apt-get`**. Ces packages sont téléchargés depuis des dépôts publics (rien à voir avec les dépôts Docker), et installés automatiquement une fois téléchargés. Lancez ceci (toujours dans le conteneur). Chez vous lancez les commandes **`apt-...`** sans les préfixer du **`http_proxy=...`** :

```
http_proxy=http://129.20.239.9:3128 apt-get update
http_proxy=http://129.20.239.9:3128 apt-get install cmake
```

Le **`apt-get update`** permet de mettre à jour le catalogue des packages disponibles. Ce n'est pas la peine de le faire avant chaque **`apt-get install`**. Il est utile la 1ère fois car le catalogue est certainement vide ou obsolète, puis une fois de temps en temps ou si le **`apt-get install`** ne trouve rien alors qu'il devrait trouver le package que vous souhaitez installer.

Puis :

- Recommencez la compilation là où vous vous étiez arrêté. Il y aura quelques Warnings, mais pour notre test ce sera OK ainsi. 😇
- Si vous avez suivi les instructions à la lettre et que la compilation s'est bien déroulée, le binaire, résultat de la compilation, s'appelle **`steganography`** et se trouve dans le dossier **`build/`**. Déplacez ce binaire dans **`/usr/local/bin`**, qui est un des dossiers dans lesquels le système trouve les commandes disponibles pour tout le monde.

  Notez que vous avez le droit de placer des commandes dans **`/usr/local/bin`** car vous êtes **`root`** dans votre conteneur et que c'est l'arborescence de votre conteneur, pas celle de l'hôte. Ce ne serait pas possible sur votre machine hôte, votre poste de travail de l'IUT, car vous n'êtes pas **`root`** sur l'hôte !

  Comme pour toutes les commandes du système, maintenant que **`steganography`** est dans **`/usr/local/bin`**, vous pouvez le lancer directement de n'importe où et sans préciser de chemin d'accès.

Tests

DANS LE CONTENEUR :

- Placez-vous dans le dossier **`/work/steganography`**
- Lancez un encodage :

```
steganography encode -i data/orig.png -e data/jekyll_and_hyde.zip -o output.png
```

DANS VSC (qui permet donc de regarder dans le volume attaché au conteneur) :

- Attachez-vous à votre conteneur (comme indiqué dans le document de rappel **TD 1 - Rappels VSC + Volumes**) et placez-vous dans **`/work/steganography`**.
- Regardez et comparez visuellement **`data/orig.png`** et **`out.png`**. Rien ne laisse supposer que **`out.png`** contient un fichier zip (le **`data/jekyll_and_hyde.zip`**) !

DANS LE CONTENEUR :

- Lancez un décodage de **`out.png`** :

```
steganography decode -i output.png -o out.zip
```

DANS L'HÔTE :

Depuis VSC récupérez le fichier **`out.zip`** du volume **`r408`** vers votre machine hôte, puis dézippez **`out.zip`** et vérifiez que vous avez bien extrait des données cachées dans l'image **`output.png`**.

Exercice 2 - Image personnalisée

Introduction

Nous allons maintenant transformer l'installation faite dans l'exercice 1 en une image personnalisée, autonome et prête à l'emploi pour servir à créer de nouveaux conteneurs avec une commande **`steganography`** immédiatement opérationnelle !

Méthode "de bourrin"

Il est possible de transformer un conteneur en une image.

Ce ne peut être qu'une solution ponctuelle et exceptionnelle car elle présente l'inconvénient d'embarquer tout ce qui a été modifié dans le conteneur et certainement tout un tas de choses inutiles qui resteraient de vos manipulations dans le conteneur entre sa création et le moment où vous en faites une image.

Nous allons voir, plus loin, une autre solution plus propre et plus pérenne de faire une image à partir d'une autre image à laquelle on apporte des modifications.

Pour créer une image d'un conteneur, **DANS L'HÔTE** :

- Arrêter le conteneur. C'est conseillé mais si vous avez créé votre conteneur avec une option **`--rm`**, ça pose le problème que vous allez perdre votre conteneur et que vous ne pourrez donc plus en faire une image.

  Dans ce cas, vous devez le laisser en fonction, même si cette solution présente des inconvénients, et notamment celui que le système étant en fonction, vous allez aussi embarquer des choses (fichiers/dossiers) liées à son activité en cours.
- Lancez ensuite un :

```
docker container commit ID_ou_nom_conteneur stegano:latest
```

- Vous devez avoir créé (ou écrasé si déjà existante) une image **`stegano`** taguée en version **`latest`**. Vérifiez-le en consultant la liste des images présentes sur votre machine.

Nous allons maintenant tester cette nouvelle image :

- Recréez, en mode interactif et, cette fois-ci, **sans monter de volume** (ce point est très important) un nouveau conteneur à partir de cette image et vérifiez que vous avez bien accès à la commande **`steganography`** (essayez de l'exécuter).
- Avez-vous un dossier **`/work`** dans ce conteneur ?

Les volumes ne font pas partie des conteneurs, ce sont des dossiers de l'hôte qui sont partagés dans le conteneur. De ce fait, la création d'une image à partir d'un conteneur n'embarque pas les volumes ! Vous avez bien un binaire **`steganography`** (dans **`/usr/local/bin`** qui est un dossier du conteneur) mais vous n'avez plus le code source qui était dans un volume monté dans le conteneur d'origine. CQFD.

DANS L'HÔTE :

- Arrêtez votre conteneur
- Supprimez votre volume **`r408`**
- Recréez-en un nouveau avec le même nom **`r408`**
- Relancez un nouveau conteneur sur la base de votre nouvelle image **`stegano`** cette fois-ci, en mappant encore le volume **`r408`** sur **`/work`** dans le conteneur.

- A l'aide de VSC, vérifiez que ce volume **`r408`** (monté sur **`/work`**) est bien vide (puisqu'il a été recréé après sa suppression). Si ce n'est pas le cas, vous avez raté une étape !
- Dans le dossier **`/work`**, glissez-déposez un PNG quelconque pour servir de source au "support" stéganographique.
- Dans le dossier **`/work`**, créez un fichier **`secret.txt`** avec le contenu que vous voulez.

DANS LE CONTENEUR :

- Refaites un test d'encodage et de décodage en utilisant **`/work/secret.txt`** (à la place de **`data/jekyll_and_hyde.zip`** qui n'existe plus pour ce test) et le chemin vers votre images (à la place de **`data/orig.zip`** qui n'existe plus) et vérifiez dans l'hôte que vous avez bien une image produite.

Méthode "du pro qui maîtrise son sujet" : Dockerfile

On va construire ça ensemble, au tableau…