Isabel Banobre Garcia 23/04/19

Projet : MIPS Stéganographie – Architecture des ordinateurs

Introduction :

Pour ce projet, j’ai commencé en faisant un cahier des charges des différents besoins. J’ai cherché des algorithmes dans d’autres langages pour essayer de bien comprendre le processus à suivre pour pouvoir cacher un texte dans une image BMP. Puis j’ai regardé des exemples types de programme en MIPS. (Gestion des arguments / valeurs de retour / subroutines / stack /array)

1. Encodage

L'encodage se découpe en plusieurs parties :  
-Une partie de gestion d'input de l’utilisateur afin de récupérer les différentes données (menu pour le choix des images / chaîne de caractère à cacher / nom du fichier pour la sauvegarde).   
-Une autre de codage de la string en binaire afin de pouvoir modifier plus facilement les pixels de l’image. En effet, un caractère possède une représentation en ASCII qui permet de le transcrire en binaire. J’ai donc mis chaque caractère sur un octet. Pour ce faire, j’ai fait un shift de l’itération de cet octet (j = 7 -> j = 0) en faisant un shift left de j et en comparant celui-ci avec un ‘and’ 1. Ce qui permet de récupérer chaque bit du caractère un a un.  
-Une troisième partie de gestion d’utilisation d’open / read / close du file descriptor comparable au langage C. Où dedans j’ai fait une lecture du header afin de récupérer les données importantes (par exemple taille du fichier). Pour pouvoir allouer en mémoire un array du fichier total et ainsi pouvoir le modifier.  
-Une de modification des pixels en fonction de la string transcrit en binaire. En ajoutant 1 ou non au pixel en fonction de chaque bit de l’octet du caractère en ascii.  
-Une autre de sauvegarde du fichier et de suppression du \n afin d’avoir un fichier réutilisable par la suite.  
-Enfin une partie de gestion d’erreur.  
  
Afin d’avoir une architecture assez stable, j’ai donc utilisé une partie main dans laquelle je store les registres $s et où je fais des ‘jal’ pour faire des jumps vers des ‘subroutines’ (fonction) dans lesquelles j’envoie oui ou non des arguments initialisés $a. Dans la plupart des cas je retourne des valeurs $v dans la main que je sauvegarde par la suite dans des registres.  
J’ai aussi mis en place des fonctions de contrôle pour avoir une meilleure visibilité sur les erreurs de mes instructions.  
  
  
  
  
  
Pour connaitre la fin de la string dans l’array de pixel, j’ai mis le caractère ‘/’ qui vaut 47 en ascii, il suffit donc de chercher la fin de la string et de la parcourir en chemin inverse pour voir s’il reste des caractères identifiable, jusqu’au début de celle-ci et de convertir les valeurs binaires en ascii.

1. Décodage

Je n’ai guère eu le temps de faire cette partie étant donné que j’étais avec un binôme qui m’a lâché, j’ai donc dû faire le projet seul, mais je pense qu’avec un peu plus de temps j’aurais certainement pu la finir. J’ai seulement pu faire la première partie en affichant les outputs demandés dans le sujet. J’ai perdu énormément de temps sur la compréhension du langage et son utilisation en programme complet

1. Fichiers du dossier

Dans le dossier, il y a les images sources dans lesquelles il n’y aucun changement et un autre dossier avec les images avec un texte caché dans chacune d’entre elle :  
-imageOne : “What do you want ? I'm a hidding text, do you see me ? yes you do”  
-imageTwo: “No puedo ir a la escuala porque tengo una palabras que nadie entendien”