Nerojan Rajaratnam
Quentin Chouleur
5A - UFA

Secure Programming

Sécurisation du code source DynStat.c

Introduction

Ce rapport vise à retracer les différentes modifications apportées au code source DynStat.c dans le cadre de sa sécurisation. Nous allons expliquer chacune des modifications effectuées, en fonction de la ligne, en exprimant leurs motivations et les solutions implémentées. Les logiciels ayant été utilisés dans la realisation de ce projet sont *valgrind* et *flawfinder,* des recherches supplémentaires sur des règles de programmation sécurisé ont été effectuées en plus sur différents sites web.

Au préalable une modification du programme pour être compiler de manière portable sur toutes les platesformes. Nous avons remplacer l'appel de la fonction `strcpy_s` spécifique à *Microsoft Windows* par l'appel de la fonction `strncpy`. Nous avons aussi inclus les fichiers d'en-têtes manquant pour la definition des fonctions utilisées.

Historique des modifications

- Initialisation des toutes les variables locales

- Vérification du nombre d'argument (ligne 20)

  Pour que notre programme s'exécute correctement il faut s'assurer qu'il reçoit le nombre attendu d'argument, dans notre cas nous attendons deux arguments. Dans la version originale sans vérification, un appel sans le bon nombre d'argument faisait planter le programmer.

- Ajout d'un format dans l'appel de la fonction printf (ligne 27)

  La version originale était vulnérable à une attaque de type format string, l'utilisateur pouvant passer des formats en argument et donc voir le contenu de la pile. Nous avons corrigé cette vulnérabilité en utilisant un format dans l'appel de la fonction printf afin de se prémunir d'un argument malicieux.

- Initialisation de l'espace mémoire alloué (ligne 31)

  Nous avons changé la méthode utilisée pour l'allocation dynamique de mémoire et avons opté pour la méthode calloc. Contrairement à malloc, cette méthode initialise l'espace mémoire alloué et permet donc de supprimer une erreur remontée par valgrind "Uninitialised value was created by a heap allocation".

- Vérification du bon déroulement de l'allocation dynamique (ligne 33)

  Nous évaluons la valeur de retour de la fonction d'allocation dynamique de mémoire pour nous assurer qu'elle s'est bien déroulée. Dans la version originale cette vérification n'était pas effectuée et pouvait entrainer l'utilisation d'un buffer pointant vers NULL.

- Modification de la taille utilisée en paramètre de la fonction strncat (ligne 39)

  L'utilisation de sizeof sur un pointeur est une pratique déconseillée pour obtenir la taille de l'espace mémoire associé (cf. http://cwe.mitre.org/data/definitions/467.html). Ayant de plus connaissance de la taille du tampon utilisé nous pouvons directement l'indiquer.

- Verification du respect de la taille maximale (ligne 44 à 52)

  La fonction `strncpy` peut tronquer la chaine de caractères passée en argument si sa taille est supérieure à celle indiquée lors de l'appel de la méthode. Il convient d'en un premier temps de s'assurer que le caractère de fin de chaine est bien présent en l'ajoutant à la fin du tampon et de vérifier ensuite l'exactitude de la copie effectuée. Si la copie a été tronquée, nous avons decider d'arrêter l'exécution du programme.

- Suppression de la fuite mémoire (ligne 71)

  Dans la version originale du programme il existait une fuite mémoire, celle-ci avait, en plus de notre simple analyse, été détectée par *valgrind*. En effet, l'espace mémoire alloué précédemment n'était jamais désalloué car, l'appel de la méthode `free` est présent dans un block jamais exécuté car la condition sur la taille du buffer était toujours vraie. Nous avons donc sorti l'appel à la méthode `free` de ce block pour éviter toute fuite mémoire.

Conclusion

La sécurisation du code source de DynStat nous a permis de mettre en oeuvre plusieurs outils permettant d'analyser d'éventuels problèmes de mémoire et la présence de potentiel vecteurs d'attaque. Le code est beaucoup plus robuste que dans sa version d'origine et ne présente plus d'erreur selon les outils d'analyse.