Technique d'attaque : exploitation logicielle

Print format bug et automatisation

Pour EPITA, Spécialisation APPINGI2

2018

Luc Delsalle
Julien Sterckeman

Introduction

Vous devrez, dans le premier exercice, rechercher et exploiter une faille de type *string format bug*. Il vous suffit pour cela de suivre pas à pas la méthode vue en cours.

Le second exercice consiste à trouver une faille dans un programme dont vous n'avez pas les sources et à en faire un exploit utilisable avec ce magnifique outil qu'est Metasploit.

C'est un peu la synthèse de ce qu'il faut faire afin d'être l'auteur d'un exploit (n'oubliez pas qu'il est illégal d'utiliser un exploit contre un serveur qui ne vous appartient pas, ainsi que de distribuer un exploit, mais ce n'est pas une raison pour ne pas faire l'exercice).

Rendu

Vous devez rendre une tarball `login_x-3.tar.bz2` (en remplaçant `login_x` par votre login) par mail à `julien.sterckeman@gmail.com` et `lde.teaching@protonmail.com` avant le dimanche 27 mai 2018 à 20h00 avec les balises `[TATT][APPINGI2][RENDU][3]`.

Votre tarball doit extraire les fichiers demandés dans un répertoire `login_x-3` :

```
$ tar xjf sterck_j-3.tar.bz2
$ ls sterck_j-3
1sfb_exploit.txt
2sfb_exploit.c
3ftp_fuzzer.py
4ftp_exploit.rb
```

Votre rendu doit montrer que vous avez bien compris le cours. Il faut justifier les commandes et les arguments que vous avez utilisés (ne pas simplement les recopier sans comprendre), afficher des lignes pertinentes de désassemblage ou de commandes shell pour expliquer les valeurs ou adresses utilisées dans votre démarche.

Exercices

String format bug (4 points)

Vous trouverez les sources d'un serveur FTP à l'adresse :

`https://tatt.abcdefgh.xyz/app/files/sfb_sterck_j-ftpd.tar.bz2`

L'administrateur système ne veut pas qu'il tourne en permanence, mais il accepte que vous le lanciez de temps en temps. Il est donc en setuid root, afin de pouvoir écouter sur le port 21. Mais avec vos yeux avertis, vous avez découvert une faille de type string format dans les sources et vous comptez bien l'exploiter (même si c'est mal). Vous devrez donc détourner ce programme afin qu'il lance un shell root.

Préparatifs (3.5 points)

Dans cette première partie, vous devrez rendre un fichier `1sfb_exploit.txt` qui contient les commandes (et leur sortie) que vous avez utilisées pour :

1. trouver dans le code source la ligne vulnérable (1 point) ;

2. trouver le numéro de l'argument de la fonction exploitable (de la famille de la fonction `printf`) correspondant au buffer dans la pile 0.5 point) ;
3. obtenir l'adresse que vous voulez modifier avec cette faille (0.5 point) ;
4. calculer l'adresse du shellcode dans l'environnement (en désactivant la randomization de l'espace d'adressage, cf. partie 5.1.2 du PDF du cours) (0.5 point) ;
5. construire la chaîne de format pour exploiter le vulnérabilité (1 point).

Coder l'exploit (0.5 point)

Vous devrez rendre un fichier `2sfb_exploit.c` qui, une fois compilé et lancé, crée un fichier auxiliaire nécessaire à l'exploitation et exécute le serveur FTP en l'exploitant (il faut que le serveur lancé par votre exploit exécute un shell). Votre exploit ne doit PAS calculer lui même les valeurs de la chaîne de format, il faut que vous l'indiquiez en dur dans le code.

Vous pouvez désactiver la randomization de l'adressage de la pile pour placer votre shellcode dans l'environnement.

Fuzzer et Metasploit (5 points)

Le binaire, que vous connaissez déjà, mais corrigé du *string format bug*, se trouve à l'adresse :

`https://tatt.abcdefgh.xyz/app/files/my_ftpd`

Le fichier `users.txt` se trouve dans le même répertoire.

Trouver la vulnérabilité (3 points)

Vous devez tout d'abord créer un fuzzer pour tester le serveur FTP. Vous devez rendre un fichier `3ftp_fuzzer.c`, `3ftp_fuzzer.pl`, `3ftp_fuzzer.py` ou `3ftp_fuzzer.rb`. Votre fuzzer doit se connecter à un serveur distant déjà lancé, il ne doit pas lancer lui même le serveur. Nous supposons que vous connaissez un login et un mot de passe valide pour vous connecter (ces deux informations doivent correspondre à deux variables dans votre fuzzer).

Votre fuzzer doit tester **toutes** les commandes FTP à la recherche de vulnérabilités (1 point), en effectuant **plusieurs** tests **pertinents** pour chaque commande (1 point). Votre fuzzer doit pouvoir être utilisé pour tester d'autres serveurs que celui de l'exercice et doit afficher de façon claire la commande et les paramètres qui ont fait planter le serveur FTP (0.5 point).

Dans un commentaire au début de votre source (0.5 point), vous devez :
— expliquer en français les tests effectués par votre fuzzer (construction automatisée des chaînes) ;
— afficher des extraits de la sortie fournie par votre fuzzer lorsqu'on l'utilise sur le serveur FTP vulnérable de l'exercice. Vous ne devez montrer que des parties pertinentes (un test négatif et un test positif) en utilisant `[...]` pour limiter la taille du résultat.

Coder l'exploit (2 points)

Une fois que vous avez trouvé la vulnérabilité, il est temps de trouver un moyen de l'exploiter et d'en faire un plugin pour Metasploit.

Votre exploit doit fonctionner même avec la randomization de la pile. Pour trouver des adresses utilisables pour l'exploitation, il faudra utiliser `gdb` afin d'analyser l'état des registres et de la pile.

Puisque le serveur effectue un `fork` pour chaque client, n'oubliez pas de spécifier la commande `set follow-fork-mode child` dans le débogueur. Vous pouvez utiliser un *cyclic pattern* généré avec Metasploit, comme vu en TP, afin de déterminer la taille exacte du buffer.

Une fois l'adresse trouvée, il ne reste plus qu'à créer un exploit pour Metasploit. Il suffit pour cela de copier l'exemple du cours (qui est également un exploit pour un serveur FTP) et de modifier les champs adéquats de la méthode `initialize` (1 point) et le code de la méthode `exploit` (0.5 point). Il est conseillé de tester le module créé, pour s'assurer que tous les champs ont bien été modifiés (c'est de plus l'occasion de jouer avec Metasploit). Le shellcode de test pourra être de type *bindshell*, puisque vous exploitez un programme à distance. Vous devez rendre le fichier `4ftp_exploit.rb` qui correspond au module Metasploit. Ce fichier doit contenir un commentaire avec le choix de l'adresse de retour (code correspondant et contexte au moment de l'exécution) (0.5 point).