Stack overflow: exploitation des failles de sécurité dans les programmes multi-threads

Sid Touati*

2020

Lieu

Laboratoire INRIA Sophia Antipolis, France. Equipe-projet KAIROS.

Prérequis espérés

Programmation (C, système, multi-threads, assembleur), architecture des processeurs, compilation, Linux.

Présentation générale du domaine

La sécurité informatique est un thème assez large, avec des contours flous: protection des données contre des lectures non désirées, protection des transmissions contre les interceptions, protection des systèmes d'exploitation contre les intrusions, protections des logiciels contre une mauvaise utilisation, protection contre la modification de fichiers ou des données en mémoire, virus, chevaux de Troie, malware, hameçonnage, etc. Le terme "sécurité" est utilisé actuellement dans un tas de domaines informatiques, à des niveaux divers d'abstraction: niveaux haut (protocoles, interfaces homme-machine), niveaux intermédiaires (réseaux, middeware), niveaux bas (code binaire, code assembleur, couches basses des systèmes d'exploitation).

Dans ce TER, nous nous focalisons sur un cas connu et précis en informatique bas niveau: il s'agit du débordement de pile d'une fonction (*stack overflow*). Lorsqu'un programme contient un bug qui corrompt la pile d'exécution d'une fonction, il est possible à un hacker de dérouter le code du programme pour faire exécuter un code malicieux. Prenons un exemple en C:

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer,str);
}
void main() {
  char large_string[256];
  int i;
  for( i = 0; i < 255; i++)
    large_string[i] = 'A';
  foo(large_string);
}
```

*Professeur à l'université Côte d'Azur. Sid.Touati@inria.fr

Dans cet exemple, la taille du buffer alloué à l'intérieur de la fonction `foo` n'est pas suffisante pour contenir la chaine de caractères donnée en paramètre. En cas d'exécution du programme, la pile de la fonction `foo` débordera sur la zone mémoire qui lui sera allouée, provoquant dans ce cas un plantage du programme. Un compilateur C classique ne fait pas une analyse avancée du programme pour détecter ce type de bugs. Un programmeur quelconque ne songera pas à activer des analyses statiques de codes C pour détecter ces erreurs. Quelle serait la conséquence au delà d'un crash du programme ? C'est là que le bas blesse, car la conséquence d'un tel bug n'est pas qu'un simple un plantage d'exécution.

Pour le comprendre, il faut se rappeler de l'organisation bas niveau de la mémoire d'une fonction qui s'exécute en natif sur un processeur: la pile d'une fonction contient non seulement les données locales à la fonction, mais aussi l'adresse de l'instruction de retour de la fonction. En d'autres termes, lorsque la fonction `foo` finit de s'exécuter, le flot d'exécution du programme se branchera vers une instruction dont l'adresse est stockée sur la pile. Ainsi, si un programmeur malveillant se débrouille bien, il peut introduire dans la pile un code binaire malveillant (sous forme de données locales à la fonction), et aussi un branchement vers ce code malveillant.

Ce type de bugs de débordement de pile existent dans beaucoup de logiciels, créant des failles de sécurité un peu partout. Toutes ces failles ne sont pas forcément connues ni exploitées, les hackers essayent généralement de s'attaquer à des logiciels répandus ou critiques: les commandes shell d'un système, les services des systèmes d'exploitation, les serveurs (web, bases de données), les logiciels libres, etc.

La meilleure façon d'éviter ces failles est de prouver que les codes soient corrects. Bien entendu, cela n'est pas toujours possible, et la majorité des logiciels diffusés ne sont ni prouvés ni analysés contre ce type de failles. Des compilateurs proposent des parades au détriment des performances du code. Certaines de ces parades sont elles aussi contournées par les hackers, ce qui renvoie au dilemme habituel en sécurité: l'éternel combat entre le glaive et le bouclier.

Objectifs du TER

Ce TER a un objectif essentiellement pédagogique. Il offre à l'étudiant une première expérience de spécialisation dans le domaine de la sécurité bas niveau, ou de compilation selon la sensibilité du candidat. Le sujet ici est d'exploiter la faille provoquée par un débordement de pile, et étudier les mécanismes de parades. D'autres types de failles existent, mais ne seront pas étudiées ici. L'étudiant devra étudier quatre grandes tâches:

1. Étude bibliographique du domaine (articles scientifiques et documents en ligne): attaques et parades.
2. Reproduire des attaques informatiques exploitant la faille de débordement de pile sur des exemples de programmes multi-threads s'exécutant sous Linux.
3. Tester les parades des compilateurs contre les attaques de type débordement de pile.
4. Réfléchir à de nouvelles méthodes de compilation pour prévenir ces attaques.

Références bibliographiques pour commencer

- Smashing the Stack For Fun and Profit. Aleph One.
- `https://travisf.net/smashing-the-stack-today`