Grenoble INP ENSIMAG

L I G

Travaux d'Études et de Recherches

Test en frelatage (fuzzing) et code binaire

20 Mai 2011

Olivier Benjamin

LIG - Équipe VASCO
Encadrant : Roland Groz

Plan

- Cadre du travail
- Problème posé
- Réflexions
- Expérimentations
- Bilan

Cadre du travail

- LIG, équipe VASCO

  VAlidation de Systèmes, Composants, et Objets Logiciels
- Sofia Bekrar : VASCO, Verimag, VUPEN
- Test en frelatage
- Evaluation de la qualité

Test en frelatage

- Méthode de test pour détecter des vulnérabilités
- Fournir des entrées aléatoires, mal formées
- Très utile dans les tests en boîte noire
- Très bon ratio détection/coût

Test en frelatage

- Problème :

  On teste beaucoup de valeurs au hasard

  Optique d'efficacité : évaluation

  Comment déterminer ce qui a été (bien) testé ?

  Exemple : somme de contrôle

  Microsoft : 100 000 itérations

Problème posé

- Réponse classique : couverture d'instructions

  Ratio instruction exécutées / instructions totales

  Même importance pour toutes les instructions

  Mal adapté :

  Toutes les instructions n'ont pas la même importance

  Le contexte n'est pas pris en compte

- Objectif : trouver d'autres moyens de mesure

Réflexions

- Beaucoup de vulnérabilités viennent du mauvais usage d'une fonction

  exemple utilisé : fonction strcpy de la libc

- Repérer les appels à cette fonction, et vérifier qu'ils ont été testés

- Dans un premier temps : appels seuls
- Critère trop simple : l'appel peut être inoffensif
- Dans un deuxième temps : chaînes d'instructions : regarder les instructions précédentes

- Chaînes d'instructions :

  Quelles instructions sont susceptibles d'être exécutées successivement

  Après le test, lesquelles l'ont effectivement été

Expérimentations

Objectif : Écrire un outil implémentant ces approches et vérifier leur utilité en pratique

L'outil doit être capable de distinguer les parties du programme jugées intéressantes pour cette métrique

Les résultats doivent correspondre à une évaluation des risques faites par un humain

Expérimentations

- Outils : IDA pro + IDApython
- Travail réalisé : écriture d'un script python pour IDA pro répondant au besoin
- Test sur des exemples de programmes simples

Approche simplifiée

Fonctionnement :

- Répertorie les instructions recherchées présentes
- Ajoute un breakpoint
- Rencontre un breakpoint → répertorie l'exécution
- Compare les instructions présentes à celles testées

Premier exemple

```
#include <stdio.h>
#include <stdlib.h>
#include <readline/readline.h>

int main(int argc, char* argv[]){

    if(!strcmp(argv[1], "1")){
        printf("le premier argument vaut 1\n");
    }else if (!strcmp(argv[1],"2")){
        printf("le premier argument vaut 2\n");
    }else if (!strcmp(argv[1],"3")){
        printf("le premier argument vaut 3\n");
    }else if (!strcmp(argv[1],"4")){
        char *p = readline(">>>");
        char buffer[20];
        strcpy(buffer, p);
   }
}
```

Premier exemple

- Fonction recherchée : strcpy
- Un appel déclenché par l'entrée '4'
- Vulnérabilité de type buffer overflow

Résultats

- Si entrée différente de '4'

  Le programme prévient de l'appel dangereux manqué

- Si entrée '4'

  Le programme marque l'appel

- On arrive à détecter l'appel convenablement

Deuxième exemple

```
#include <stdio.h>
#include <stdlib.h>
#include <readline/readline.h>

int main(int argc, char* argv[]){
    if(!strcmp(argv[1], "1")){
        printf("le premier argument vaut 1\n");
                ...
    }else if (!strcmp(argv[1],"4")){
        char *p = readline(">>>") ;
        char buffer[20];
        if(strlen(p)<19){
            strcpy(buffer, p) ;
        }
    }else if (!strcmp(argv[1],"5")){
        char *p = readline(">>>");
        char buffer[20];
        strcpy(buffer, p);
    }
}
```

Résultats

- Le programme détecte le(s) appel(s) non déclenché(s)
- Il détecte ou non l'appel préliminaire à strlen
- On arrive à distinguer les cas pour chaque appel à strcpy

Bilan

- On peut alors définir une mesure numérique

- Il faut pour cela attribuer un poids à chaque chaine d'instructions intéressante (expérience, expertise, études statistiques)

- On peut alors calculer une mesure :

  Stestées / Stotales

  - Stestées somme des poids des instructions testées
  - Stotale somme des poids des instructions totales

Conclusion

- On a proposé deux métriques possibles
- On a implémenté un programme montrant qu'on peut les calculer
- On a testé chacune sur un programme simple, et on a trouvé le résultat cohérent

Perspectives

- Tester ces approches sur plus de programmes, des programmes plus complexes
- Implémenter des méthodes de pattern matching plus élaborées
- Tester avec un environnement de fuzzing
- Automatiser la rétroaction sur l'environnement de fuzzing

MERCI