

Fuzzgrind : un outil de fuzzing automatique

Gabriel Campana

Sogeti / ESEC

[gabriel.campana\(at\)sogeti.com](mailto:gabriel.campana(at)sogeti.com)



Plan

- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
- 4 Conclusion

Roadmap

- 1 Introduction
 - État de l'art
 - Objectif
 - Concept
- 2 Valgrind et STP
- 3 Implémentation
- 4 Conclusion

Roadmap

- 1 Introduction
 - État de l'art
 - Objectif
 - Concept
- 2 Valgrind et STP
- 3 Implémentation
- 4 Conclusion

État de l'art

- Fuzzing : technique de recherche d'erreurs d'implémentation logicielle par injection de données invalides.
- Multitude de fuzzers.
- Principale méthode de génération de test : grammaire ou modèle.
- Processus sans fin : analyse des spécifications, reverse du protocole, nouveau développement pour chaque nouvelle cible, etc.
- Nouveaux concepts innovants : Autodafé, Flayer, Sage, etc.

Roadmap

- 1 Introduction
 - État de l'art
 - **Objectif**
 - Concept
- 2 Valgrind et STP
- 3 Implémentation
- 4 Conclusion

Objectif

- Rendre le fuzzing **complètement** automatique.
- Donner un programme cible et un fichier en entrée,
- Nouvelles entrées générées automatiquement,
- Attendre les crashes.

Roadmap

- 1 Introduction
 - État de l'art
 - Objectif
 - Concept
- 2 Valgrind et STP
- 3 Implémentation
- 4 Conclusion

Concept

- 1 Exécution symbolique du programme cible sur une entrée donnée,
- 2 Analyse du chemin d'exécution et extraction des conditions de chemin,
- 3 Inversion de chaque contrainte,
- 4 Génération de nouvelles entrées.
- 5 L'algorithme est répété jusqu'à ce que tous les chemins d'exécution soient (idéalement) couverts.

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=6

état symbolique

i0

contraintes

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=6, b=15

état symbolique

i0
i1

contraintes

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=6, b=15

état symbolique


i0

i1

contraintes

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}
```



```
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=6, b=15

état symbolique

i0

i1

contraintes

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=6, b=15, c=9

état symbolique

i0
i1

c=i0+3

contraintes

Exécution symbolique : exemple

```

void main() {
    int a, b;
    FILE *fp = fopen("input", "r");

    fread(&a, sizeof(int), 1, fp);
    fread(&b, sizeof(int), 1, fp);

    f(a, b);
}

void f(int a, int b) {
    int c = a + 3;
    if (c > 42) {
        if (a - b == 7)
            error();
    }
}
    
```



état concret	état symbolique	contraintes
	i0 i1	
a=6, b=15, c=9	c=i0+3	i0+3 <= 42

Exécution symbolique : exemple

```

void main() {
  int a, b;
  FILE *fp = fopen("input", "r");


  fread(&a, sizeof(int), 1, fp);
  fread(&b, sizeof(int), 1, fp);

  f(a, b);
}

void f(int a, int b) {
  int c = a + 3;

  if (c > 42) {
    if (a - b == 7)
      error();
  }
}

```



état concret	état symbolique	contraintes
	i0 i1	
<div style="border: 2px solid black; background-color: #FFD700; padding: 10px; display: inline-block;"> équation : $i0 + 3 > 42$ solution : $i0 = 40$ </div>		
a=6, b=15, c=9	c=i0+3	$i0+3 \leq 42$

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=40

état symbolique

i0

contraintes

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=40, b=15

état symbolique

i0

i1

contraintes

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

a=40, b=15

état symbolique

i0

i1

contraintes

Exécution symbolique : exemple

```
void main() {
    int a, b;
    FILE *fp = fopen("input", "r");

    fread(&a, sizeof(int), 1, fp);
    fread(&b, sizeof(int), 1, fp);

    f(a, b);
}
```



```
void f(int a, int b) {
    int c = a + 3;

    if (c > 42) {
        if (a - b == 7)
            error();
    }
}
```

état concret

a=40, b=15

état symbolique

i0
 i1

contraintes

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```

état concret

état symbolique

contraintes

i0

i1

a=40, b=15, c=43

c=i0+3

Exécution symbolique : exemple

```

void main() {
    int a, b;
    FILE *fp = fopen("input", "r");

    fread(&a, sizeof(int), 1, fp);
    fread(&b, sizeof(int), 1, fp);

    f(a, b);
}


void f(int a, int b) {
    int c = a + 3;
    if (c > 42) {
        if (a - b == 7)
            error();
    }
}
    
```



état concret	état symbolique	contraintes
	i0 i1	
a=40, b=15, c=43	c=i0+3	i0+3 > 42

Exécution symbolique : exemple

```
void main() {  
    int a, b;  
    FILE *fp = fopen("input", "r");  
  
    fread(&a, sizeof(int), 1, fp);  
    fread(&b, sizeof(int), 1, fp);  
  
    f(a, b);  
}  
  
void f(int a, int b) {  
    int c = a + 3;  
  
    if (c > 42) {  
        if (a - b == 7)  
            error();  
    }  
}
```



état concret

a=40, b=15, c=43

état symbolique

i0
i1

c=i0+3

contraintes

i0+3 > 42
i0 - i1 != 7

Exécution symbolique : exemple

```

void main() {
    int a, b;
    FILE *fp = fopen("input", "r");

    fread(&a, sizeof(int), 1, fp);
    fread(&b, sizeof(int), 1, fp);

    f(a, b);
}

void f(int a, int b) {
    int c = a + 3;

    if (c > 42) {
        if (a - b == 7)
            error();
    }
}
    
```



état concret	état symbolique	contraintes
	i0 i1	
<div style="border: 2px solid black; background-color: #FFD700; padding: 10px;"> équation : $i_0 + 3 > 42 \ \&\& \ i_0 - i_1 == 7$ solution : $i_0 = 40, \ i_1 = 33$ </div>		
a=40, b=15, c=43	c=i0+3	$i_0+3 > 42$ $i_0 - i_1 != 7$

Exécution symbolique : exemple

```

void main() {
    int a, b;
    FILE *fp = fopen("input", "r");

    fread(&a, sizeof(int), 1, fp);
    fread(&b, sizeof(int), 1, fp);

    f(a, b);
}

void f(int a, int b) {
    int c = a + 3;

    if (c > 42) {
        if (a - b == 7)
            error();
    }
}
    
```



état concret	état symbolique	contraintes
	i0 i1	
	c=i0+3	i0+3 > 42 i0 - i1 == 7
a=40, b=33, c=43		

Roadmap

- 1 Introduction
- 2 Valgrind et STP
 - Valgrind
 - STP
- 3 Implémentation
- 4 Conclusion

Roadmap

- 1 Introduction
- 2 Valgrind et STP
 - Valgrind
 - STP
- 3 Implémentation
- 4 Conclusion

Valgrind

- Framework d'instrumentation binaire dynamique.
- Utilisation du cœur de Valgrind pour créer des plugins d'analyse binaire dynamique (exemple : *Memcheck*).
- Interpréteur de code machine.

Fonctionnement simplifié

- Lancement de Valgrind et du programme cible dans un même processus.
- ① Analyse du programme cible par *basic block*,
- ② Désassemblage et traduction en représentation intermédiaire,
- ③ Instrumentation de la représentation intermédiaire par le plugin,
- ④ Conversion en code machine,
- ⑤ Allocation des registres,
- ⑥ Génération du code final.

Représentation intermédiaire et instrumentation

```
0x4000A99: movl %eax,%ecx
```

```
#----- IMark(0x4000A99, 2) -----
```

```
PUT(4) = GET:I32(0) # copie d'EAX dans ECX
```

```
0x4000A9B: leal 0x2C(%ebx), %esi
```

```
#----- IMark(0x4000A9B, 6) -----
```

```
PUT(60) = 0x4000A9B:I32 # mise à jour d'EIP
```

```
t0 = Add32(GET:I32(12),0x2C:I32) # addition du contenu d'EBX avec 0x2C
```

```
PUT(24) = t0 # copie du résultat dans ESI
```

Roadmap

- 1 Introduction
- 2 Valgrind et STP
 - Valgrind
 - STP
- 3 Implémentation
- 4 Conclusion

STP – A Fast Prover

- Solveur de contraintes générées par des programmes d'analyse statique.
- Utilisé dans l'article « *Automatic Patch-Based Exploit Generation* ».
- Particulièrement rapide.
- Prend en entrée une requête composée d'une ou plusieurs contraintes.
- La sortie indique si la requête est satisfiable ou non.
- Exhibition d'un contre-exemple.

Exemple

```
# cat file.c
...
char x, y;
if (x * y == 16)
...
```

```
# cat file.stp
x : BITVECTOR(8);
y : BITVECTOR(8);
QUERY(NOT(BVMULT(8, x, y) = 0h10));
```

```
# stp -p file.stp
Invalid.
ASSERT( y = 0hex05 );
ASSERT( x = 0hexD0 );
```

Roadmap

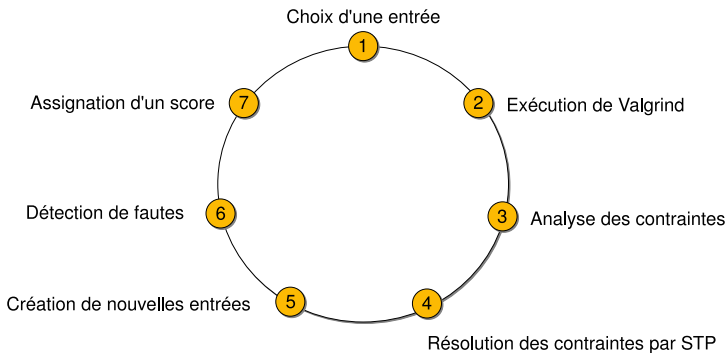
- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
 - Plugin Valgrind
 - Analyse des contraintes
 - Résolution des contraintes
 - Détection de fautes
 - Calcul des scores
- 4 Conclusion

Vue d'ensemble

Utilisation par Fuzzgrind :

- Valgrind : recherche des conditions de chemins.
- STP : résolution des contraintes.
- Scripts python pour lier le tout.

Vue d'ensemble



Roadmap

- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
 - Plugin Valgrind
 - Analyse des contraintes
 - Résolution des contraintes
 - Détection de fautes
 - Calcul des scores
- 4 Conclusion

2) Plugin Valgrind : marquage des données

- Sources supportées : fichier ou entrée standard.
- Marquage des données issues de la source suivie (*taint tracing*).
- Association d'une contrainte à chaque donnée marquée.
- Affichage des contraintes sur chaque condition rencontrée.

2) Plugin Valgrind : surveillance des appels systèmes

- Marquage initial des données :
- `open` : suivi du descripteur de fichier,
- `read`, `mmap` : adresses des données marquées,
- `close` : stoppe le suivi du descripteur de fichier.
- Donnée marquée : contrainte dépendant du numéro i de l'octet du fichier lu ou `mmapé`.

2) Plugin Valgrind : propagation et affichage

- Analyse de la représentation intermédiaire.
- Instruction ne dépend pas d'une donnée marquée \implies ignorée,
- Opération sur une donnée marquée \implies
 - Marquage du temporaire associé,
 - Association de la contrainte liée au temporaire.

`t0 = Add32(GET:I32(12), 0x2C:I32)`

- Condition dépendant d'une donnée marquée \implies affichage de la contrainte associée.

`0x08048e0d: CmpEQ32(8Uto32(LD1e:I8(input(0))),0x0:I32) => 0`

Roadmap

- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
 - Plugin Valgrind
 - **Analyse des contraintes**
 - Résolution des contraintes
 - Détection de fautes
 - Calcul des scores
- 4 Conclusion

3) Analyse des contraintes

- Traduction de la représentation intermédiaire de Valgrind vers le langage de STP.
- Optimisation et négation des contraintes.

```
0x08048e0d: CmpEQ32(8Uto32(LD1e:I8(input(0))),0x0:I32) => 0
```

```
x0 : BITVECTOR(8);  
QUERY(  
  NOT((  
    IF (((0h000000@x0) = 0h00000000)) THEN  
      (0b1)  
    ELSE  
      (0b0)  
    ENDIF)  
  = 0b1));
```

Roadmap

- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
 - Plugin Valgrind
 - Analyse des contraintes
 - **Résolution des contraintes**
 - Détection de fautes
 - Calcul des scores
- 4 Conclusion

4) Résolution des contraintes, 5) Nouvelles entrées

- Résolution des contraintes par STP.

```
./stp/stp -p /tmp/example.stp  
Invalid.  
ASSERT( x0 = 0hex00 );
```

- Si insatisfiable, assignation de nouvelles valeurs,
- Création de nouvelles entrées.

Roadmap

- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
 - Plugin Valgrind
 - Analyse des contraintes
 - Résolution des contraintes
 - **Détection de fautes**
 - Calcul des scores
- 4 Conclusion

6) Détection de fautes

- Ptrace,
- Signaux SIGSEGV, SIGKILL, SIGABRT.
- Crackmes, tests : recherche de motifs dans la sortie.

Roadmap

- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
 - Plugin Valgrind
 - Analyse des contraintes
 - Résolution des contraintes
 - Détection de fautes
 - **Calcul des scores**
- 4 Conclusion

7) Calcul des scores

- Plugin Lackey de Valgrind,
- Nombre de basic blocks exécutés.

Démo

Roadmap

- 1 Introduction
- 2 Valgrind et STP
- 3 Implémentation
- 4 Conclusion

Résultats

- Fuzzing complètement automatique.
- Nouvelles vulnérabilités dans `readelf` et `swfextract`.
- Vulnérabilités dans la version 3.8.2 de la bibliothèque `libtiff` en quelques minutes, à l'origine du déblocage de l'iPhone et de la PSP.
- Résolution de crackmes simples.
- Adapté au fuzzing de bibliothèques.

Améliorations envisageables

- Remplacement de Valgrind par PIN ou DynamoRio.
- Suivi d'entrées réseau.
- Outil de calcul de score.
- Parallélisation.
- Correction de bugs...
- Fuzzgrind est sous licence GPL : contribuez !
- <http://www.security-labs.org/fuzzgrind>

Merci de votre attention.

Des questions ?