Using Evolutionary Algorithms to obfuscate code

Benoît Bertholon[1], Sébastien Varrette[2] et Pascal Bouvry[2]

[1] Security and Trust (SnT) interdisciplinary center,
[2] Computer Science and Communication (CSC) Research Unit
University of Luxembourg, Luxembourg

uni.lu UNIVERSITÉ DU LUXEMBOURG

SnT securityandtrust.lu

Fonds National de la Recherche Luxembourg

Outline

1. Context et Motivations
2. Obfuscation
3. Evolutionary algorithms to obfuscate code

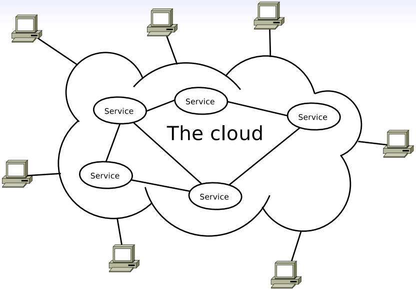
Outline

1 Context et Motivations

2 Obfuscation

3 Evolutionary algorithms to obfuscate code

Context: Cloud Computing paradigm



Basic idea of the Cloud Computing paradigm

- to outsource computing services,
- to use a service without knowing the infrastructure,
- different types of Cloud: SaaS, PaaS, IaaS.

Context: Cloud Computing paradigm

Security issues in Cloud Computing

- Confidentiality of the user's data...
- Potential disclosure of the user algorithms.

How to hide information in the software

→ Obfuscation

Outline

1. Context et Motivations

2. **Obfuscation**

3. Evolutionary algorithms to obfuscate code

Obfuscation

Before

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

int main(int argc, char * argv[])
{
  printf("Hello␣world\n");
}
```

Obfuscation

After

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include <signal.h>
#define m(b)a=b;z=*a;while(*++a){y=*a;*a=z;z=y;}
#define h(u)G=u<<3;printf("\e[%uq",l[u])
#define c(n,s)case n:s;continue
 char x[]="((((((((((((((((((((((",w[]=
 "\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b";char r[]={92,124,47},l[]={2,3,1
 ,0};char*T[]={"␣␣|","␣␣|","%\\|/%","␣%%%",""};char d=1,p=40,o=40,k=0,*a,y,z,g=
 -1,G,X,**P=&T[4],f=0;unsigned int s=0;void u(int i){int n;printf(
 "\233;%uH\233L%c\233;%uH%c\233;%uH%s\23322;%uH@\23323;%uH␣\n",*x-*w,r[d],*x+*w
 ,r[d],X,*P,p+=k,o);if(abs(p-x[21])>=w[21])exit(0);if(g!=G){struct itimerval t=
 {0,0,0,0};g+=((g<G)<<1)-1;t.it_interval.tv_usec=t.it_value.tv_usec=72000/((g>>
 3)+1);setitimer(0,&t,0);f&&printf("\e[10;%u]",g+24);}f&&putchar(7);s+=(9-w[21]
 )*((g>>3)+1);o=p;m(x);m(w);(n=rand())&255||--*w||++*w;if(!(**P&&P++||n&7936)){
 while(abs((X=rand()%76)-*x+2)-*w<6);++X;P=T;}(n=rand()&31)<3&&(d=n);!d&&--*x<=
 *w&&(++*x,++d)||d==2&&++*x+*w>79&&(--*x,--d);signal(i,u);}void e(){signal(14,
 SIG_IGN);printf("\e[0q\ecScore:␣%u\n",s);system("stty␣echo␣-cbreak");}int main
 (int C,char**V){atexit(e);(C<2||*V[1]!=113)&&(f=(C=*(int*)getenv("TERM"))==(
 int)0x756E696C||C==(int)0x6C696E75);srand(getpid());system("stty␣-echo␣cbreak"
 );h(0);u(14);for(;;)switch(getchar()){case 113:return 0;case 91:case 98:c(44,k
 =-1);case 32:case 110:c(46,k=0);case 93:case 109:c(47,k=1);c(49,h(0));c(50,h(1
 ));c(51,h(2));c(52,h(3));}}
```

BrainFuck

Hello World

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]
>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.
```

Syntax

"<" pointer move left

">" pointer move right

"+" add 1 to the data at the position of the pointer

"-" sub 1 to the data at the position of the pointer

"[" begin of the while statement

"]" end of the while statement

"." output the value of the pointer

"," input the value of the pointer

Definitions

Definition (Obfuscation)

Transformation of a program P into a obfuscated program P' with the following attributes:

- P' has the same behavior that P.
- P' should be harder to understand.

2 Levels

- Source to Source Obfuscation
- binary Obfuscation

Definitions

Definition (Resilience)

The Transformation Resilience is the addition of the two measures:

- the programmer effort.
- deobfuscator effort.

Definition (Cost)

The Transformation Cost is the extra execution time and space of P' compared to P

Obfuscation is impossible

Barak et al. [Barak01]

- Obfuscation proven impossible
  - ↪ Virtual Black-box impossible.
- End of the story?

Time limited Black-Box [Hohl98]

- Guarantee the black box property for a limited time.
- Same as in RSA or ECC

Metrics 1/2

Metrics

1. Program Length: number of operators & operands in P[Halstead77].
2. Cyclomatic Complexity: number of predicates in F [McCabe76] .
3. Nesting Complexity: nesting level of conditionals in F [Harrison81] .
4. Data Flow Complexity: number of inter-basic block variable references in P [Oviedo80] .
5. Fan-in/out Complexity: number of formal parameters to F, and number of global data structures read or updated by F [Henry81] .

Metrics 2/2

Metrics

6. Data Structure Complexity: number of dimension and type in an array [Munson93] .
7. OO (Object Oriented) Metric:
   - number of methods in C
   - the distance from the root of C
   - the number of direct subclasses of C
   - the number of other classes to which C is couple
   - the number of methods that can be executed in response to a message sent to an object of C [Chidamber94] .

Some transformation's examples

Some transformation's examples

- μ_1 Program Length → Insert Dead code
- μ_2 Cyclomatic Complexity → Parallelize code
- μ_3 Nesting Complexity → Extend loop condition
- μ_4 Data Flow Complexity → Change variable lifttime
- μ_5 Fan-in/out Complexity → Interleaving methodes
- μ_6 Data Structure Complexity → Split Array
- μ_7 OO (Object Oriented) → Insert Bogus Classes

Some examples

μ_3 Nesting Complexity → Extend loop condition

```
i=1;
while(i<100){
   ...
   i++;
}
```

→

```
i=1;j=100
while((i<100) &&
      (j*j*(j+1) * (j+1) * %4 == 0)){
   ...
   i++;
}
```

μ_4 Data Flow Complexity → Change variable lifetime

```
void f(...){
   int i ; .... i ....;
}
void g(...){
   int k ; .... k ....;
}
```

→

```
int C:
void f(...){
     .... C ....;
}
void g(...){
     .... C ....;
}
```

Some examples

μ_5 Fan-in/out Complexity → Interleaving methodes

```
void f1(int a, int b ) {S1;}
void f2(int a, int b ) {S2;}
int main(){
  int a , b ,c ;
  ..
  f1(a,b);
  f2(a,c);
}
```

→

```
void f1(int a, int b , int v) {
  if(v == p)
    {S1;}
  else
    {S2;}
}
int main(){
  int a , b ,c ;
  ..
  f(a,b,v=p);
  f(a,c,v=q);
}
```

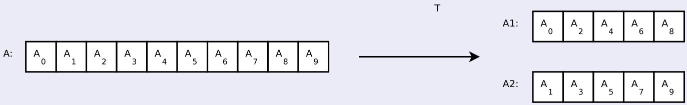
Some examples

μ_6 Data Structure Complexity → Split Array

```
int i;
...
A[i]
```

→

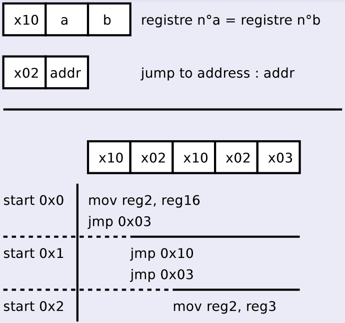
```
int i;
...
if( (i%2) )
    A2[i/2];
else
    A1[i/2];
```


T
A1:
A_0 A_2 A_4 A_6 A_8
A:
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9
A2:
A_1 A_3 A_5 A_7 A_9

Low Level Obfuscation [Linn03]

Idea

- Assembler instruction doesn't have the same length.



x10 | a | b registre n°a = registre n°b

x02 | addr jump to address : addr

x10 | x02 | x10 | x02 | x03

start 0x0 mov reg2, reg16
jmp 0x03

start 0x1 jmp 0x10
jmp 0x03

start 0x2 mov reg2, reg3

Collberg Obfuscation Algorithm

Main Obfuscation Algorithm[Collberg97]

- While not done(Program):
  - Select Code
  - Select transformation
  - Apply the transformation to the selected code.
- End While

Outline

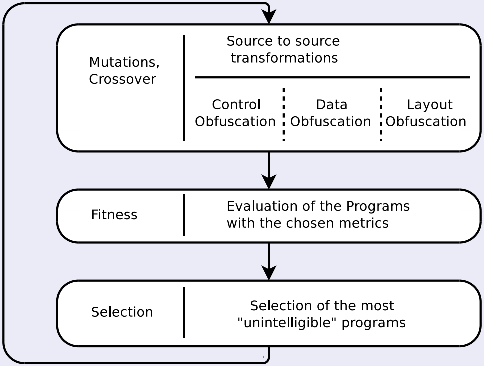
1. Context et Motivations

2. Obfuscation

3. Evolutionary algorithms to obfuscate code

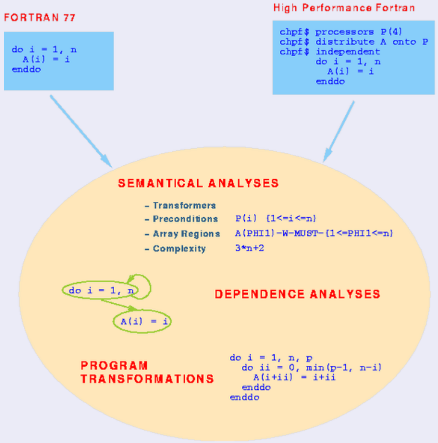
Using Evolutionary Algorithm for Obfuscation purposes

Idea


Mutations, Crossover
Source to source transformations
Control Obfuscation
Data Obfuscation
Layout Obfuscation
Fitness
Evaluation of the Programs with the chosen metrics
Selection
Selection of the most "unintelligible" programs

PIPS: Automatic Parallelizer and Code Transformation Framework [http://cri.ensmp.fr/pips/]

PIPS as a source2source compiler



FORTRAN 77

```
do i = 1, n
  A(i) = i
enddo
```

High Performance Fortran

```
chpf$ processors P(4)
chpf$ distribute A onto P
chpf$ independent
      do i = 1, n
        A(i) = i
      enddo
```

SEMANTICAL ANALYSES

- Transformers
- Preconditions P(i) {1<=i<=n}
- Array Regions A(PHI1)-W-MUST-{1<=PHI1<=n}
- Complexity 3*n+2

DEPENDENCE ANALYSES

PROGRAM TRANSFORMATIONS

```
do i = 1, n, p
  do ii = 0, min(p-1, n-i)
    A(i+ii) = i+ii
  enddo
enddo
```

PIPS: Automatic Parallelizer and Code Transformation Framework [http://cri.ensmp.fr/pips/]

PIPS as a source2source compiler

- Transformation of C code,
- Using Existing Transformation and evaluate them.
- PYPS: Python binding.

Issues

- Implementation of the transformations.
- Definition of a valid metric.

Work in Progress

Work in Progress

- Definition of a representative metric.
- Extend Collberg's work.
- EA as a tool to obfuscate code.
    - Multi-objective Optimizations based on $\mu_1 ... \mu_7$

Merci pour votre attention...

Questions?