طراحی و پیاده سازی ابزار Obfuscator برای Mini-C

شمیم صداقت تیموری 40219043

مقدمه :

مبهم سازی فرآیندی است که در آن کد منبع طوری تغییر داده می شود که فهم آن برای انسان سخت تر شود، بدون آن که عملکرد برنامه تغییر کند. این تکنیک معمولا در امنیت نرم افزار و جلوگیری از مهندسی معکوس استفاده شود.

هدف پروژه :

طراحی و پیاده سازی یک ابزار Obfuscation که روی کد های Mini-C اعمال شده و نسخه معادل از نظر عملکرد اما غیرقابل فهم برای انسان تولید کند .

محدوده زبان Mini-C

1. انواع داده پایه : int, char, bool
2. متغیر ها و عملگر ها
3. کنترل جریان :return, for, while, else, if
4. ورودی/ خروجی از طریق : scanf , printf
5. بدون pointer, struct

موارد پیاده سازی شده :

طراحی و پیاده سازی AST برای زبان Mini-C

استفاده از بازدیدکننده (Visitor) برای تولید کد مبهم شده

اعمال سه تکنیک مبهم سازی

تحلیل و مقایسه عملکرد کد اصلی و کد مبهم شده

ابزار و فناوری ها

زبان برنامه نویسی : java

ANTLR برای تولید Parse Tree

Intellij IDEA برای توسعه و دیباگ

GitHub برای مدیریت نسخه ها

نحوه انجام

1. تحلیل نحوی (parsing) : استفاده از گرامر Mini-C برای تولید درخت نحو (Parse Tree) با ANTLR
2. ساخت AST : تعریف کلاس AST و تبدیل Parse Tree به AST با استفاده از ASTBuilderVisitor
3. مبهم سازی : اعمال تکنیک های مختلف در ObfuscatorVisitor

4. تولید خروجی : نوشتن کد مبهم شده در فایل خروجی output.mc
5. مقایسه عملکرد کد اصلی و کد مبهم شده از نظر خروجی، حافظه و زمان performanceComparator

تکنیک های مبهم سازی پیاده سازی شده

در این پروژه، از سه تکنیک مبهم سازی استفاده شده است :

1. تغییر نام متغیر ها و توابع

   تبدیل نام توابع به فرم f0, f1 ,...

   تبدیل نام متغیر ها به فرم v0, v1, ...

2. افزودن کد مرده (Dead code)

   اضافه کردن کدهایی که اجرا نمی شوند ولی ظاهر برنامه را پیچیده تر میکنند مانند : (در هر statement با احتمال 80 درصد کد مرده اضافه می شود)

   1. تعریف متغیر هایی که استفاده نمی شوند به فرم x0, x1, ...
   2. دستورات شرطی unreachable مانند :

```
if (0) {
    int z0 = 23;
}
if ( 1 > 2 ) {
   int y0 = 12;
```

```
}
```

3. افزودن حلقه for که اجرا نمی شوند مانند :

```
for( int i = 0; i < 0; i++){
    int loop = i * 42;
}
```

3 . استفاده از عبارات معادل پیچیده تر

عبارات ساده به فرم معادل ولی پیچیده تر بازنویسی می شوند . مانند :

a = b + 1 → a = b – (-1)
a = b → a = b + 0
if(x) → if(!!x)
a * 2 → a << 1
a / 2 → a << 1

نحوه مقایسه عملکرد کد اصلی و کد مبهم شده :

پس از ایجاد فایل output.mc هر دو فایل ورودی و خروجی را تبدیل به فایل .c میکنیم و اندازه هر کدام را خروجی میدهیم . سپس با استفاده از gcc هر کدام را کامپایل و اجرا میکنیم از این طریق زمان اجرا و خروجی را میتوانیم بررسی کنیم.

نمونه ورودی و خروجی :

Input1.mc

```
int func(int a, int b) {
    int temp = b * 8;
    int result = a + b;
    return result;
}

int main() {
    int x = 3;
    int y = 4;
    int total = func(x, y);
    printf("%d\n", total);
    return 0;
}
```

output1.mc

```
int f0(int v0, int v1) {
    if (0) {
       int z0 = 81;
    }
    int v2 = (v1 << 3);
    int x0 = 70;
    int v3 = (v0 + v1);
    if (0) {
       int z1 = 73;
    }
    return v3;
}
int main() {
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
    int v4 = 3;
    if (1 > 2){
       int y0 = 5;
    }
    int v5 = 4;
    if (0) {
       int z2 = 36;
    }
    int v6 = f0(v4, v5);
    int x1 = 41;
    printf("%d\n", v6);
```

```
    if (1 > 2){
       int y1 = 3;
    }
    return 0;
    if (1 > 2){
       int y2 = 2;
    }
}
```

مقایسه عملکرد

```
obfuscation complete for input1.mc
Size of files :
Original Size : 239
Obfuscated Size : 505
output of Original : 7
output of Obfuscated : 7
Runtime
Original Time : 314ms
Obfuscated Time : 76ms
```

Input2.mc

```
int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

int main() {
    int x = 5;
    int y = 9;
    int bigger = max(x, y);
    printf("%d\n", bigger);
    return 0;
}
```

output2.mc

```
int f0(int v0, int v1) {
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
    if ((v0 > v1)) {
        for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
        return v0;
        for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
} else {
        if (1 > 2){
       int y0 = 9;
    }
        return v1;
        for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
}
    if (1 > 2){
       int y1 = 8;
    }
}
int main() {
    if (1 > 2){
       int y2 = 3;
    }
    int v2 = 5;
    if (0) {
       int z0 = 16;
    }
    int v3 = 9;
    int x0 = 75;
    int v4 = f0(v2, v3);
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
    printf("%d\n", v4);
    return 0;
    if (1 > 2){
       int y3 = 0;
    }
}
```

مقایسه عملکرد

```
obfuscation complete for input2.mc
Size of files :
Original Size : 248
Obfuscated Size : 714
output of Original : 9
output of Obfuscated : 9
Runtime
Original Time : 80ms
Obfuscated Time : 80ms
```

Input3.mc

```
int factorial(int n) {
    int result = 1;
    int i;
    for (i = 1; i <= n; i = i + 1) {
        result = result * i;
    }
    return result;
}

int main() {
    int num = 4;
    int fact = factorial(num);
    printf("%d\n", fact);
    return 0;
}
```

output3.mc

```
int f0(int v0) {
    if (0) {
       int z0 = 89;
    }
    int v1 = 1;
    int v2;
    if (1 > 2){
```

```
        int y0 = 1;
    }
    for (v2 = (0 + 1); (v2 <= v0); v2 = (v2 - (-1))) {
        if (0) {
        int z1 = 27;
    }
        v1 = (v1 * v2);
        if (1 > 2){
        int y1 = 8;
    }
}
    int x0 = 4;
    return v1;
    if (0) {
        int z2 = 12;
    }
}
int main() {
    for (int i = 0; i < 0; i++) {
        int loop = i * 42;
    }
    int v3 = 4;
    int x1 = 53;
    int v4 = f0(v3);
    for (int i = 0; i < 0; i++) {
        int loop = i * 42;
    }
    printf("%d\n", v4);
    for (int i = 0; i < 0; i++) {
        int loop = i * 42;
    }
    return 0;
}
```

مقایسه عملکرد :

```
obfuscation complete for input3.mc
Size of files :
Original Size : 285
Obfuscated Size : 628
output of Original : 24
output of Obfuscated : 24
Runtime
Original Time : 76ms
Obfuscated Time : 103ms
```

Input4.mc

```
int square(int n) {
    return n * n;
}

int main() {
    int a = 6;
    int s = square(a);
    printf("%d\n", s);
    return 0;
}
```

output4.mc

```
int f0(int v0) {
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
    return (v0 * v0);
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
}
int main() {
    if (0) {
       int z0 = 0;
    }
    int v1 = 6;
    int v2 = f0(v1);
    if (1 > 2){
       int y0 = 7;
    }
    printf("%d\n", v2);
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
    return 0;
    int x0 = 70;
}
```

مقایسه عملکرد

```
obfuscation complete for input4.mc
Size of files :
Original Size : 160
Obfuscated Size : 406
output of Original : 36
output of Obfuscated : 36
Runtime
Original Time : 81ms
Obfuscated Time : 88ms
```

Input5.mc

```
int sumToN(int n) {
    int sum = 0;
    int i = 1;
    while (i <= n) {
        if (i % 2 == 0) {
            sum = sum + i;
        }
        i = i + 1;
    }
    return sum;
}

int main() {
    int number = 10;
    int result = sumToN(number);
    printf("%d\n", result);
    return 0;
}
```

output5.mc

```
int f0(int v0) {
    if (0) {
       int z0 = 50;
    }
    int v1 = 0;
    int v2 = 1;
    int x0 = 71;
    while ((v2 <= v0)){
        int x1 = 80;
        if (((v2 % 2) == 0)) {
            if (1 > 2){
```

```
       int y0 = 4;
    }
         v1 = (v1 + v2);
         if (1 > 2){
       int y1 = 8;
    }
}
     int x2 = 50;
     v2 = (v2 - (-1));
     if (0) {
       int z1 = 41;
    }
}
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
    return v1;
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
}
int main() {
    if (1 > 2){
       int y2 = 0;
    }
    int v3 = 10;
    if (0) {
       int z2 = 30;
    }
    int v4 = f0(v3);
    if (0) {
       int z3 = 23;
    }
    printf("%d\n", v4);
    if (0) {
       int z4 = 34;
    }
    return 0;
    for (int i = 0; i < 0; i++) {
       int loop = i * 42;
    }
}
```

مقایسه عملکرد

```
obfuscation complete for input5.mc
Size of files :
Original Size : 328
Obfuscated Size : 798
output of Original : 30
output of Obfuscated : 30
Runtime
Original Time : 76ms
Obfuscated Time : 78ms
```

همان طور که از نتایج دیده می شود تمام کد ها بدون تغییر در خروجی، و بدون افزایش چشمگیر runtime مبهم شده است. و به دلیل مبهم شدن کد حجم تمامی فایل ها افزایش داشته است .