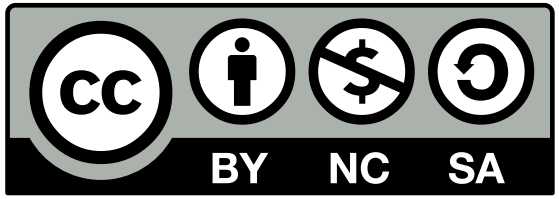
Obfuscation Methods

Dimitrios Yfantidis (UID:3938)
ydimitri@csd.auth.gr

Aristotle University of Thessaloniki,
School of Informatics @Faculty of Sciences

9 May 2023



Introduction

Obfuscation is the concealment of the intended meaning of the communication, making the message unintelligible, by using confusing and ambiguous language.

Like encryption, it contributes to protecting a set of data from malicious entities. However, obfuscation and encryption are two different concepts that should not be confused.

Obfuscation and Encryption

Encryption involves transforming data sequences to different data sequences in order to completely hide the original message. Obfuscation is the concealment of the purpose or meaning of data without changing the data itself.

The goal of obfuscation is not to make it impossible to understand a message, but to cause frustration for anyone who attempts it.

- **Obfuscation:**
  "The President has arrived" $\rightarrow$ "The Eagle has landed"
- **Encryption:**
  "The President has arrived" $\rightarrow$ "buwtixwvfvliaignbvgciu"

Source Code Obfuscation

In software development, obfuscation is the act of writing source or machine code that is difficult for humans or computers to understand.

Like concealment in natural language, it can use redundant and/or difficult to understand (but equivalent) expressions to compose statements.

Source Code Obfuscation

Developers may deliberately obscure their proprietary code to hide its purpose (security through obscurity) or its logic, mainly to prevent both software tampering and reverse engineering, as well as protect their copyrights.

This can be done manually or using an automated tool, with the latter being the preferred technique in the industry.($\pi.\chi$. ProGuard, javascript-obfuscator).

Source Code Obfuscation

Prime Numbers < 1000

```
print(list(filter(None,map(lambda y:y*reduce(lambda x,y:x*y!=0,
map(lambda x,y=y:y%x,range(2,int(pow(y,0.5)+1))),1),range(2,1000)))))
```

The first 10 numbers of the Fibonacci sequence

```
print(list(map(lambda x,f=lambda x,f:(f(x-1,f)+f(x-2,f)) if x>1 else 1:
f(x,f), range(10))))
```

Mandelbrot set

```
print((lambda Ru,Ro,Iu,Io,IM,Sx,Sy:reduce(lambda x,y:x+'\n'+y,map(lambda y,
Iu=Iu,Io=Io,Ru=Ru,Ro=Ro,Sy=Sy,L=lambda yc,Iu=Iu,Io=Io,Ru=Ru,Ro=Ro,i=IM,
Sx=Sx,Sy=Sy:reduce(lambda x,y:x+y,map(lambda x,xc=Ru,yc=yc,Ru=Ru,Ro=Ro,
i=i,Sx=Sx,F=lambda xc,yc,x,y,k,f=lambda xc,yc,x,y,k,f:(k<=0)or (x*x+y*y
>=4.0) or 1+f(xc,yc,x*x-y*y+xc,2.0*x*y+yc,k-1,f):f(xc,yc,x,y,k,f):chr(
64+F(Ru+x*(Ro-Ru)/Sx,yc,0,0,i)),range(Sx))):L(Iu+y*(Io-Iu)/Sy),range(Sy
))))(-2.1, 0.7, -1.2, 1.2, 30, 80, 24))
```

source: Python Programming FAQ

Obfuscation Methods

- Code reordering
- Changing variable names (identifier renaming)
- Introduction of redundant/unnecessary code
- Insertion of jumps (conditional, unconditional)
- Variable reassignment (identifier reassigning)
- String encoding
- Control flow manipulation
- Combination of the above techniques

Obfuscation Methods

Identifier Renaming:

Original Source Code Before Rename Obfuscation

```
private void
CalculatePayroll (SpecialList employee-
Group) {
    while (employeeGroup.HasMore()) {
        employee =
employeeGroup.GetNext(true);
        employee.UpdateSalary();
        Distribute Check(employee);
    }
}
```

Reverse-Engineered Source Code After Rename Obfuscation

```
private void a(a b) {
    while (b.a()) {
        a = b.a(true);
        a.a ();
        a.(a);
    }
}
```

source: "What is Code Obfuscation?"

Obfuscation Methods

Control flow manipulation:

Original Source Code Before Control Flow Obfuscation

```
public int CompareTo (Object o) {
    int n = occurrences -
    ((WordOccurrence)o) .occurrences ;
    if (n == 0)  {
       n = String.Compare
  (word, ((WordOccurrence)o) .word) ;
    }
    return (n) ;
}
```

Reverse-Engineered Source Code After Control Flow Obfuscation

```
private virtual int _a(Object A+0) {
    int local0 ;
    int local1 ;
    local 10 = this.a - (c)  A_0.a;
    if (local10  != 0) goto i0 ;
    while  (true)  {
       return local1;
    }
    i1: local10 =
System.String.Compare(this.b,  (c)
A_0.b) ;
   goto  i0;
}
```

source: "What is Code Obfuscation?"

Obfuscation from a Scientific POV

- **Black Box Obfuscation:**
  A cryptographic primitive which would allow a computer program to be obfuscated in such a way that it is impossible to determine anything about it except its input and output behavior. It was proven to be impossible, even in theory.

Obfuscation from a Scientific POV

- **Black Box Obfuscation:**
  A cryptographic primitive which would allow a computer program to be obfuscated in such a way that it is impossible to determine anything about it except its input and output behavior. It was proven to be impossible, even in theory.

- **Indistinguishability obfuscation:**
  It is denoted by $i\mathcal{O}$ and has the defining property that hiding any two programs that compute the same mathematical function results in programs that are indistinguishable from each other.

Black Box Obfuscation

Assumption:

- $\vec{\alpha}, \vec{\beta} \in \{0, 1\}^k$, where $\vec{\beta} \neq \vec{0}$
- $C : \{0, 1\}^k \to \{0, 1\}^k$ where

$$C(\vec{x}) = \begin{cases} \vec{\beta} & \text{if } \vec{x} = \vec{\alpha} \\ \vec{0} & \text{if } \vec{x} \neq \vec{\alpha} \end{cases}$$

- $Z : \{0, 1\}^k \to \{0, 1\}^k$ where $Z(\vec{x}) = \vec{0}, \ \forall \vec{x} \in \{0, 1\}^k$
- $D : (\{0, 1\}^k)^{\{0, 1\}^k} \to \{0, 1\}$ where

$$D(\mathcal{X}) = \left\{ \begin{array}{ll} 1 & \text{if } \mathcal{X}(\vec{\alpha}) = \vec{\beta} \\ 0 & \text{if } \mathcal{X}(\vec{\alpha}) \neq \vec{\beta} \end{array} \right| \ time(\mathcal{X}) \leq poly(k)$$

- $\mathcal{BBO}$ is a black box obfuscator where $\mathcal{BBO}(P)$ is P's black box obfuscation for every computer program P.
- $C' \leftarrow \mathcal{BBO}(C)$ and $D' \leftarrow \mathcal{BBO}(D)$

Black Box Obfuscation

It can be proved that there exists a program that cannot be obfuscated by $\mathcal{BBO}$, thus rendering black box obfuscation impossible as a whole.

Black Box Obfuscation

It can be proved that there exists a program that cannot be obfuscated by $\mathcal{BBO}$, thus rendering black box obfuscation impossible as a whole.

Proof by contradiction (part A):
Assuming that the attacker can only access the program C', then it is practically impossible to find the code α, i.e:

$$Pr[C(\vec{x_0}) = \vec{\beta} \mid \vec{x_0} \xleftarrow{\$} \{0, 1\}^k] \;=\; 2^{-k}$$

Which is ≈ 0 for (not so) large k, e.g. $k > 50$. Thus they cannot distinguish C' from Z in polynomial time as $C'(\vec{x_0}) = Z(\vec{x_0}) = \vec{0}$, for every $\vec{x_0} \in \{0, 1\}^k$ within a polynomial time margin.

Black Box Obfuscation

Proof by contradiction (part B):
However, if they have access to D' then they can conclude that $D'(C') = 1$ while $D'(Z) = 0$, so they can determine that the internal implementation of C' is not that of Z. Therefore, they know more about C' beyond its input-output behavior. The programs C and D can be merged into the following program:

$$F(m, \vec{x}) := \begin{cases} C(\vec{x}) & \gamma\iota\alpha\ m = 0 \\ D(C) & \gamma\iota\alpha\ m = 1 \end{cases}$$

or else

$$F(\mathcal{X}, m, \vec{x}) := \begin{cases} \mathcal{X}(\vec{x}) & \gamma\iota\alpha\ m = 0 \\ D(\mathcal{X}) & \gamma\iota\alpha\ m = 1 \end{cases}$$

Thus, based on the aforementioned observations $\nexists\, \mathcal{BBO}(F)$.

Compromise

The best Obfuscation possible:
Obfuscator $\mathcal{O}$ is said to be optimal when it guarantees that any information that is not hidden by the obfuscated program is not hidden by any other program of the same size that computes the same function. Thus obfuscation is (literally) optimal.

This is achieved by the alternative version of black box obfuscation, which is **Indistinguishability Obfuscation**.

Indistinguishability Obfuscation

Let $i\mathcal{O}$ be some uniform probabilistic polynomial-time algorithm. Then $i\mathcal{O}$ is called an Indistinguishability Obfuscator if and only if it satisfies the conditions of **Completeness** (or **Functionality**) and **Non-Distinguishability**, i.e:

Indistinguishability Obfuscation

Let $i\mathcal{O}$ be some uniform probabilistic polynomial-time algorithm. Then $i\mathcal{O}$ is called an Indistinguishability Obfuscator if and only if it satisfies the conditions of **Completeness** (or **Functionality**) and **Non-Distinguishability**, i.e:

1. For any Boolean circuit $C(\vec{x}) : \{0, 1\}^n \rightarrow \{0, 1\}$we have:

$$Pr[C'(\vec{x}) = C(\vec{x}) : C' \leftarrow i\mathcal{O}(C)] = 1$$

Indistinguishability Obfuscation

Let $i\mathcal{O}$ be some uniform probabilistic polynomial-time algorithm. Then $i\mathcal{O}$ is called an Indistinguishability Obfuscator if and only if it satisfies the conditions of **Completeness** (or **Functionality**) and **Non-Distinguishability**, i.e:

1. For any Boolean circuit $C(\vec{x}) : \{0,\ 1\}^n \rightarrow \{0,\ 1\}$we have:

$$Pr[C'(\vec{x}) = C(\vec{x}) : C' \leftarrow i\mathcal{O}(C)] = 1$$

2. For every pair of circuits $C_0,\ C_1$ of the same size k and for every adversary, probabilistic, polynomial-time algorithm, $\mathcal{A}$ adversary:

$$Pr[\mathcal{A}(i\mathcal{O}(C_0)) = 1] \ \simeq \ Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1]$$

In other words, $\mathcal{A}$ cannot distinguish whether the initial computer program is C_0 or C_1.

Existence of $i\mathcal{O}$

Whether it is possible to have a $i\mathcal{O}$obfuscator or not depends on the answers to other questions in computer science that yet remain open, such as the existence of **one-way functions** and the answer to **P vs NP**. All current implementations are based on heuristics.

Existence of $i\mathcal{O}$

Whether it is possible to have a $i\mathcal{O}$obfuscator or not depends on the answers to other questions in computer science that yet remain open, such as the existence of **one-way functions** and the answer to **P vs NP**. All current implementations are based on heuristics.

The "five worlds" of Russell Impagliazzo:

1. **Algorithmica:** P $=$ NP, $i\mathcal{O}$ exists
2. **Heuristica:** NP problems are relatively easy, $i\mathcal{O}$ doesn't exist
3. **Pessiland:** BPP $\neq$ NP, one-way functions do not exist, $i\mathcal{O}$ doesn't exist.
4. **Minicrypt:** one-way functions exist, Public Key Cryptography is not safe, $i\mathcal{O}$ doesn't exist.
5. **Cryptomania:** Public Key Cryptography is safe, $i\mathcal{O}$ doesn't exist.
6. (extra) **Obfustopia:** $i\mathcal{O}$ exists.

References

1. Wikipedia
   "Black-Box obfuscation"
   "Indistinguishability obfuscation"
2. Wikibooks
   "Technical Legislation for Computer Engineers/Technical means to protect software from illegal copying - piracy (GR)"
3. Aayush Jain; Huijia Lin; Amit Sahai (2021)
   "Indistinguishability Obfuscation from Well-Founded Assumptions"
4. Tim Robinson (2022)
   "Obfuscation Vs. Encryption (Easily Explained)"
5. Chandan Kumar Behera; D. Lalitha Bhaskari (2015)
   "Different Obfuscation Techniques for Code Protection"

Ρεφερενζες

6. Boaz Barak; Oded Goldreich; Russell Impagliazzo; Steven Rudich; Amit Sahai; Salil Vadhan; Ke Yang (2010)
   "On the (Im)possibility of Obfuscating Programs"
7. Valentine Kabanets; Shawn Andrews (2011)
   "based on 'A Personal View of Average Case Complexity' by R. Impagliazzo, 1995"
8. Shafi Goldwasser; Guy N. Rothblum (2014)
   "On Best-Possible Obfuscation"