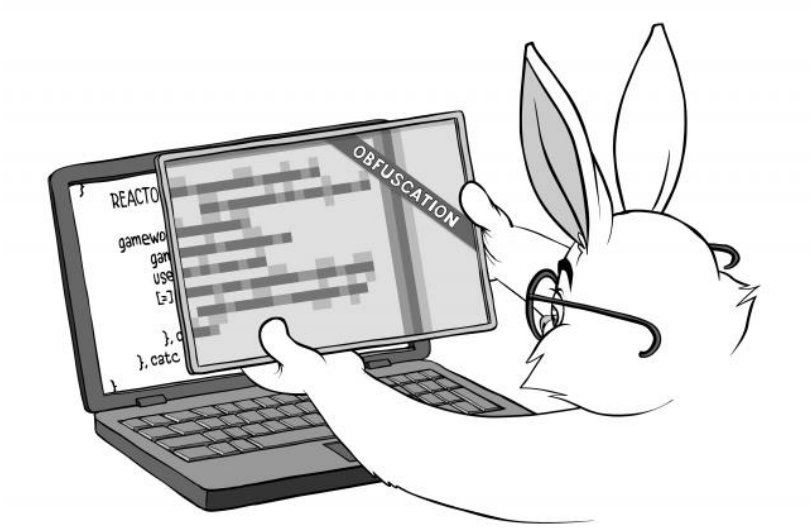
Présentation
Veille Technologique :

Cdoe Obofsucaitn



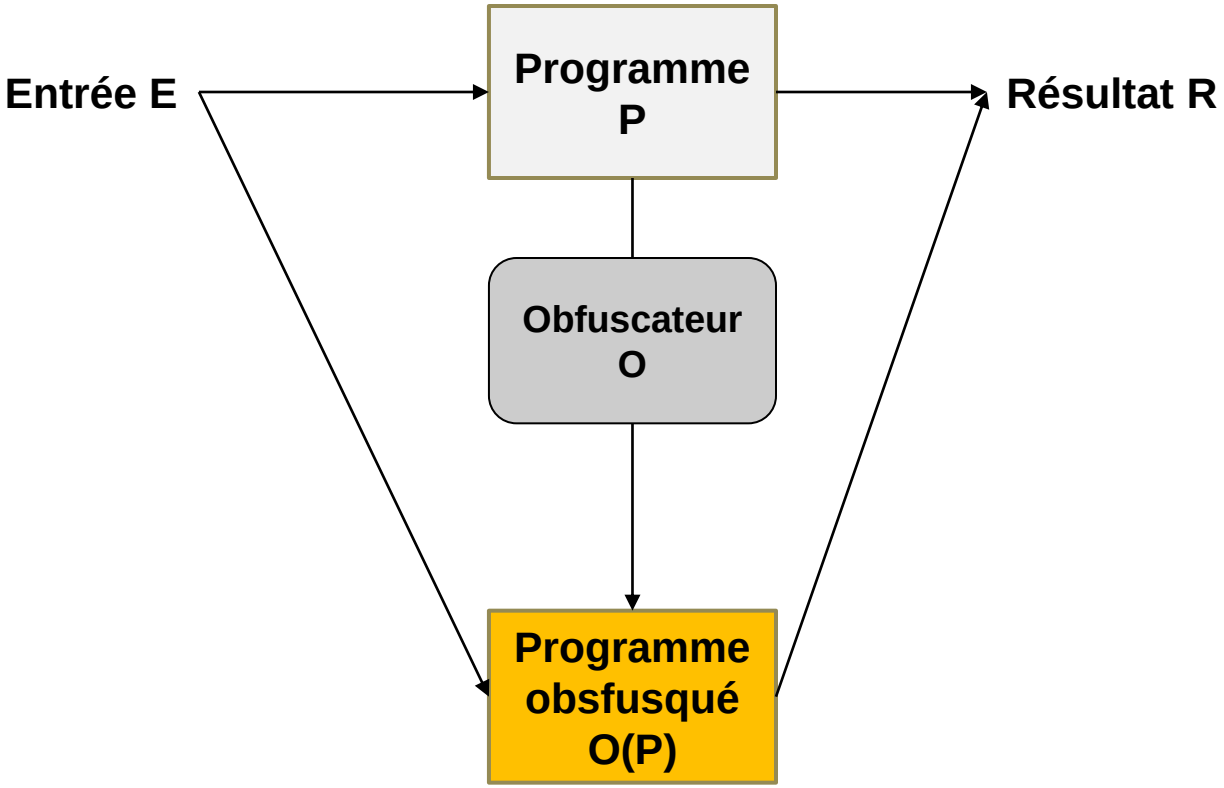
Sommaire

- Introduction
- Problématique
  - Etat actuel de l'obfuscation
  - Recherche de l'obfuscateur idéal
- Avantages et Inconvénients
- Conclusion
- Démonstration

Introduction

- Code Obfuscation:
  - “The action of making something obscure, unclear, or unintelligible”
  - Obscurçir la lecture du programme sans modifier le comportement
  - Limiter/ralentir le vol de code
- Plus subtil que des changements de variables
  - Dépend du langage source
  - International Obfuscated C Code Contest

Introduction



Introduction

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

main()
{
    char*O,l[999]="'`acgo\177~|xp .-\0R^8)NJ6%K4O+A2M(*0ID57$3G1FBL";
    while(O=fgets(l+45,954,stdin)){
        *l=O[strlen(O)[O-1]=0,strspn(O,l+11)];
        while(*O)switch((*l&&isalnum(*O))-!*l){
        case-1:{char*I=(O+=strspn(O,l+12)+1)-2,O=34;
            while(*I&3&&(O=(O-16<<1)+*I---'-')<80);
            putchar(O&93?*I&8||!(I=memchr(l,O,44))?'?':I-l+47:32);
            break;
        case 1: ;}*l=(*O&31)[l-15+(*O>61)*32];
            while(putchar(45+*l%2),(*l=*l+32>>1)>35);
        case 0: putchar((++O,32));}
    putchar(10);}
}
```

Introduction

```
[[]+1/!1][1^1][1>>1]+({}+[])[1<<1^11>>1]+([]+!!-[])[1<<1]+[/~/+{}][+!1][-~1<<1]+/\[[^1]+\]/[([]+![])
[1<<1<<1]+(/|/[(1+{})[1+11>>>1]+[[]+{}][+!1][1]+([]+1/[])[1<<1>>1]+([1<1]+[])[1+11>>>1+1]+[[!!1]+1]
[+[]][1-1]+([]+!!/!/)[1|1]+(/1/[1]+[])[!1%1]+(-{}+{})[-1+1e1-1]+(1+[!!1])[1]+([]+1+{})[1<<1]+[!!/!!/+[]]
[+[]][1&1]]+/=/)[1e1+(1<<1|1)+(([]+/-/[(!!1+[])[1>>1]+(!!1+[])[1<<1^1]+(!1+[])[1|1<<1]+(!!1+[])[1^1]])
[1^1]==+!1)]+(!![]+{})[1|1<<1]+[1+{}+1][!1+!1][(11>>1)+1]](([]+/-/[(!!1+[])[1>>1]+(!!1+[])[1<<1^1]+(!1+[])
[1|1<<1]+(!!1+[])[1^1]]))[1&.1][11>>>1]+([,][~1]+[])[1-~1]+[[]+{}][!1.1%1[11111.1%11.1*111e11|!1]+(/1/+1/[1<1]
[1%1])[1^11]+[[],[]+{}][1<<1>>>1][1||1]+(/[<+>]/[1&1|1]+[1.1])[1/11.1&1.11]
```

Introduction

Lettre « I » de la chaine de caractère « Infinity »

```
[[]+1/!1][1^1][1>>1]+({}+[])[1<<1^11>>1]+([]+!!-[])[1<<1]+[/~/+{}][+!1][-~1<<1]+/\[[^1]+\]/[([]+![])
[1<<1<<1]+(/|/[(1+{})[1+11>>>1]+[[]+{}][+!1][1]+([]+1/[])[1<<1>>1]+([1<1]+[])[1+11>>>1+1]+[[!!1]+1]
[+[]][1-1]+([]+!!/!/)[1|1]+(/1/[1]+[])[!1%1]+(-{}+{})[-1+1e1-1]+(1+[!!1])[1]+([]+1+{})[1<<1]+[!!/!!/+[]]
[+[]][1&1]]+/=/)[1e1+(1<<1|1)+(([]+/-/[(!!1+[])[1>>1]+(!!1+[])[1<<1^1]+(!1+[])[1|1<<1]+(!!1+[])[1^1]])
[1^1]==+!1)]+(!![]+{})[1|1<<1]+[1+{}+1][!1+!1][(11>>1)+1]](([]+/-/[(!!1+[])[1>>1]+(!!1+[])[1<<1^1]+(!1+[])
[1|1<<1]+(!!1+[])[1^1]]))[1&.1][11>>>1]+([,][~1]+[])[1-~1]+[[]+{}][!1.1%1[11111.1%11.1*111e11|!1]+(/1/+1/[1<1]
[1%1])[1^11]+[[],[]+{}][1<<1>>>1][1||1]+(/[<+>]/[1&1|1]+[1.1])[1/11.1&1.11]
```

Message complet « I love you »

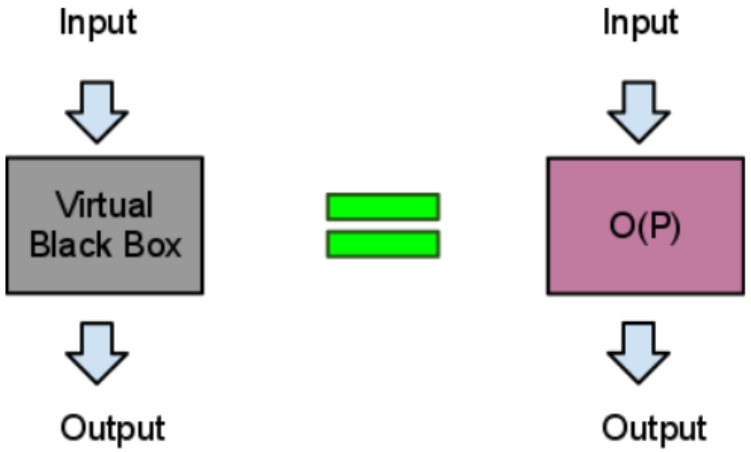
⇒**Impressionnant, mais pas assez sécurisé, universel, ou automatique.**

Problématique – Etat actuel de l'obfuscation

- Objectif : Créer une méthode d'obfuscation garantissant :
  - La sécurité absolue du code
  - L'équivalence de résultat entre P et O(P)
- Renouveau de l'obfuscation grâce à la cryptographie
- Chercheurs Satoshi Hada et Boaz Barak
  - Blackbox Obfuscation
  - Indistinguishable Obfuscation

Problématique – Etat actuel de l'obfuscation

- Blackbox Obfuscation
  - "Anything that can be efficiently computed from O(P) can be efficiently computed given oracle access to P."
  - Aucune information sur le programme obfusqué (oracle access)
  - Même résultat que le programme non obfusqué



- Problème : Tout n'est pas obfusquable

Problématique – Etat actuel de l'obfuscation

- Indistinguishable Obfuscation
  - Programme = Circuit électronique contenant des combinaisons de composants (ET, OU, NOT)
  - R un résultat, C1 et C2 des circuits
  - C1 et C2 n'ont pas le même code source mais font la même chose

    ⇒ Fonctionnellement équivalents
  - O(C1) et O(C2) renvoient le même résultat pour une entrée E

    ⇒ Informatiquement indistinguables

Problématique – Etat actuel de l'obfuscation

- Indistinguishable Obfuscation Exemple :

- C1 = ab+ac
- C2 = a(b+c)

- Même fonctionnalité, et pour O(C1) et O(C2), même résultat

- Problème : Code trop long, consomme trop de ressources

Problématique – Recherche de l'obfuscateur idéal

- Similaire au cryptage :
  - Programme incompréhensible sans la "clé"
  - Entrée et sortie en clair
- Cela créé une piste pour une réalisation de l'iO
- iO bénéfique pour le cryptage (algo de génération de clé)
- Encore théorique, mais de gros progrès sont faits

Avantages et Inconvénients

- Avantages :
  - Réduit la taille du code
  - Limite le vol de code
  - Utilisé dans d'autres domaines (ex : cryptage)
  - Versions basiques d'obfuscation

- Inconvénients :
  - Lourd et cher à mettre en place
  - Retour en arrière à prévoir
  - Renforce les résistances des virus/malwares
  - Code visiblement louche

Conclusion

- Encore beaucoup de théorie
- Provoque un grand intêret à cause de sa finalité
- Solutions basiques existantes et utilisables

⇒ Plus de sécurité = Plus de recherche pour la contrer