Obfuscation

Serge Guelton

Télécom Bretagne

25 Février 2016

Pourquoi ce cours ?

Parce que certains vivent dans un monde...

Dominé par les intérêts économiques

- Digital Right Managment
- Propriété Intellectuelle
- Clefs de chiffrement
- Violation de Licence
- Dissimulation de *malware*
- ...

Pourquoi ce cours ?

Parce que certains vivent dans un monde...

Idyllique

- Interopérabilité
- Keygen
- Recherche de vulnérabilités
- ...

Eviv Bulgroz

Titep Tset

Zenerpmoc-souv ec euq souv zesil ?
Esod Mumixam

Exemples d'utilisation

DropBox

Client codé en Python distribué avec un interpréteur modifié
cf. *Looking inside the (Drop)Box* de Dhiru Kholia et Przemyslaw W egrzyn, WOOT 2013

NVidia

Driver graphique « Open Source » mais livré. . . en code machine !

Skype

- Déchiffrement du code à la volée
- Détection de déboggueur
- Contrôle d'intégrité
- *Time checking*
- Obfuscations : *junk code*, *exceptions redirections*, *indirect calls computations*. . .

cf. *Silver Needle in the Skype* de Philipe Biondi et Fabrice Desclaux, BlackHat 2006

Obfuscation

ne vous o(b)fusquez pas pour si peu

Wikipedia

L'obfuscation, assombrissement, ou obscurcissement est une stratégie de gestion de l'information qui vise à obscurcir le sens qui peut être tiré d'un message. Cette stratégie peut être intentionnelle ou involontaire.

Exemple tiré de Wikipedia

Le recours à l'argot peut également être perçu comme une tentative d'obfuscation, visant à empêcher les non-initiés de comprendre le message échangé entre les interlocuteurs.

Quand la GPL s'en mêle

S'emmêle ?

"Obfuscated "source code" is not real source code and does not count as source code."

– `www.gnu.org/philosophy/free-sw.html`

Obfuscation & Informatique

Mais que veut-on protéger ?

Des données

- Des clefs de chiffrement (cf. crypto boîte blanche)
- Des vidéos attachées au programme
- Des sons/musiques propres au prgramme
- Des constantes caractéristiques du programme

...

Du code

- Un algorithme de chiffrement maison (c'est le mal)
- Un code de vérification de licence
- Des mesures d'anti-debug
- Un algorithme de traitement du signal révolutionnaire

...

Différents niveaux de protection des données

Il ne faut pas que :

1. Une donnée apparaisse dans le binaire final

2. Une donnée apparaisse **en mémoire** lors de l'exécution

3. Une donnée apparaisse **en registre** lors de l'exécution

Différents niveaux de protection des données

Il ne faut pas que :

1. Une donnée apparaisse dans le binaire final
   `Q: comment faire pour rechercher si la chaîne GPL apparait dans un binaire ?`
2. Une donnée apparaisse **en mémoire** lors de l'exécution

3. Une donnée apparaisse **en registre** lors de l'exécution

Différents niveaux de protection des données

Il ne faut pas que :

1. Une donnée apparaisse dans le binaire final
   `Q: comment faire pour rechercher si la chaîne GPL apparait dans un binaire ?`
2. Une donnée apparaisse **en mémoire** lors de l'exécution
   `Q: comment faire pour rechercher si la chaîne GPL est présente en mémoire ? [hint]: gcore + gdb`
3. Une donnée apparaisse **en registre** lors de l'exécution

Différents niveaux de protection des données

Il ne faut pas que :

1. Une donnée apparaisse dans le binaire final
   `Q: comment faire pour rechercher si la chaîne GPL apparait dans un binaire ?`
2. Une donnée apparaisse **en mémoire** lors de l'exécution
   `Q: comment faire pour rechercher si la chaîne GPL est présente en mémoire ? [hint]: gcore + gdb`
3. Une donnée apparaisse **en registre** lors de l'exécution
   `Q: comment faire pour rechercher si un registre contient la valeur 0xDEADBEEF ? [hint]: gdb + watch`

Différents niveaux de protection du code

Résiste[1] à

1. Une analyse statique

2. Une analyse dynamique

Z Big Boss Preuves formelles

[1] Prouve que tu existes

Différents niveaux de protection du code

Résiste[1] à

1. Une analyse statique
   `A: se ramener à un problème difficile, genre analyse d'aliasing`
2. Une analyse dynamique

Z Big Boss Preuves formelles

[1] Prouve que tu existes

Différents niveaux de protection du code

Résiste[1] à

1. Une analyse statique
   `A: se ramener à un problème difficile, genre analyse d'aliasing`
2. Une analyse dynamique
   `A: perturber la trace, la rendre non déterministe ou trop volumineuse`

Z Big Boss Preuves formelles

[1] Prouve que tu existes

Différents niveaux de protection du code

Résiste[1] à

1. Une analyse statique
   `A: se ramener à un problème difficile, genre analyse d'aliasing`
2. Une analyse dynamique
   `A: perturber la trace, la rendre non déterministe ou trop volumineuse`

Z Big Boss Preuves formelles

`A: données trop volumineuses, trop couplées...`

[1] Prouve que tu existes

Soyez une fouine !

Ou une hermine, pour être couleur locale

Transformez le code suivant pour cachez la structure de boucle :

```
unsigned mysterie(uint8_t x) {
    unsigned res = 0;
    for(int i = 0; i < 8; ++i)
        if(x & (1 << i))
            ++res;
    return res;
```

Conseils

- Regardez l'assembleur généré
- Changez l'ordre des instructions
- Changez l'algorithme
- Changez le paradigme

Désarmer le *reverser*

t'es comme select et reset...

Attaquer les outils :

1. Le décompilateur

2. Le débuggeur

3. Les VMs

Désarmer le *reverser*

t'es comme select et reset...

Attaquer les outils :

1. Le décompilateur
   `Q: Expliquez (avec dis.dis) le comportement de la séquence de bytecode python suivant :`
   `0x91, 0x01, 0x00, 0x79, 0xf7, 0x32, 0xa0, 0xed, 0x8a`
2. Le débuggeur
3. Les VMs

Désarmer le *reverser*

t'es comme select et reset...

Attaquer les outils :

1. Le décompilateur
   `Q: Expliquez (avec dis.dis) le comportement de la séquence de bytecode python suivant :`
   `0x91, 0x01, 0x00, 0x79, 0xf7, 0x32, 0xa0, 0xed, 0x8a`
2. Le débuggeur
   `Q: Lancez un processus dans gdb puis essayez de vous attacher dessus avec un autre gdb...`

3. Les VMs

Désarmer le *reverser*

t'es comme select et reset...

Attaquer les outils :

1. Le décompilateur
   `Q: Expliquez (avec dis.dis) le comportement de la séquence de bytecode python suivant :`
   `0x91, 0x01, 0x00, 0x79, 0xf7, 0x32, 0xa0, 0xed, 0x8a`
2. Le débuggeur
   `Q: Lancez un processus dans gdb puis essayez de vous attacher dessus avec un autre gdb... puis man ptrace`
3. Les VMs

Désarmer le *reverser*

t'es comme select et reset...

Attaquer les outils :

1. Le décompilateur
   `Q: Expliquez (avec dis.dis) le comportement de la séquence de bytecode python suivant :`
   `0x91, 0x01, 0x00, 0x79, 0xf7, 0x32, 0xa0, 0xed, 0x8a`
2. Le débuggeur
   `Q: Lancez un processus dans gdb puis essayez de vous attacher dessus avec un autre gdb... puis man ptrace`
3. Les VMs
   `Q: Comment détecter qu'on est dans une VM ? time attack, inspect drivers, vbox processes etc`

Jeu : je construis mon anti-debug moi-même

Étape 1

Choisir un PNG qui vous plaît

Étape 2

Modifier le premier octet

Étape 3

Essayer d'ouvrir le PNG

Étape 4

Écrire un programme qui appelle une visionneuse d'image en patchant à la volée le premier octet du PNG

Lavoisier et l'obfuscation

Rien ne se perd, rien ne se crée, tout se transforme

Pas de repas gratuit

L'obfuscation impacte :

- le temps d'exécution
- la taille du binaire
- la consommation mémoire
- la structure du programme (introspection...)

Compromis à trouver lors de l'obfuscation !

Propr(i)étés d'une obfuscation

Conservatif

Le code transformé doit avoir le même comportement (observable ?)

Furtivité

Rendre l'obfuscation difficile à déceler

Cas pratique : Calcul de la distribution de données

Furtivité

Il est rare que des données pertinentes aient une distribution aléatoire. C'est par contre le cas pour des données chiffrées ! Une bonne façon de détecter des données chiffrées serait donc de calculer la distribution des octets

Ex-Air6

Utilisez `readelf -x .rodata` ou `objdump -s -j .rodata` pour extraire les données en lecture seule d'un binaire, et calculez en l'histogramme (en vrai, il faudrait utiliser un truc comme le test du χ^2)

Qualité de l'obfuscation

Mesures empiriques

On peut regarder certaines métriques statiques :

- Single Line Of Code
- Complexité cyclomatique
- Dispersion
- Niveau d'imbrication moyen / maximal...

Mesures sur le terrain

Rien ne vaut un bon CTF !

Illustration – protection de code (1)

CFG flattening

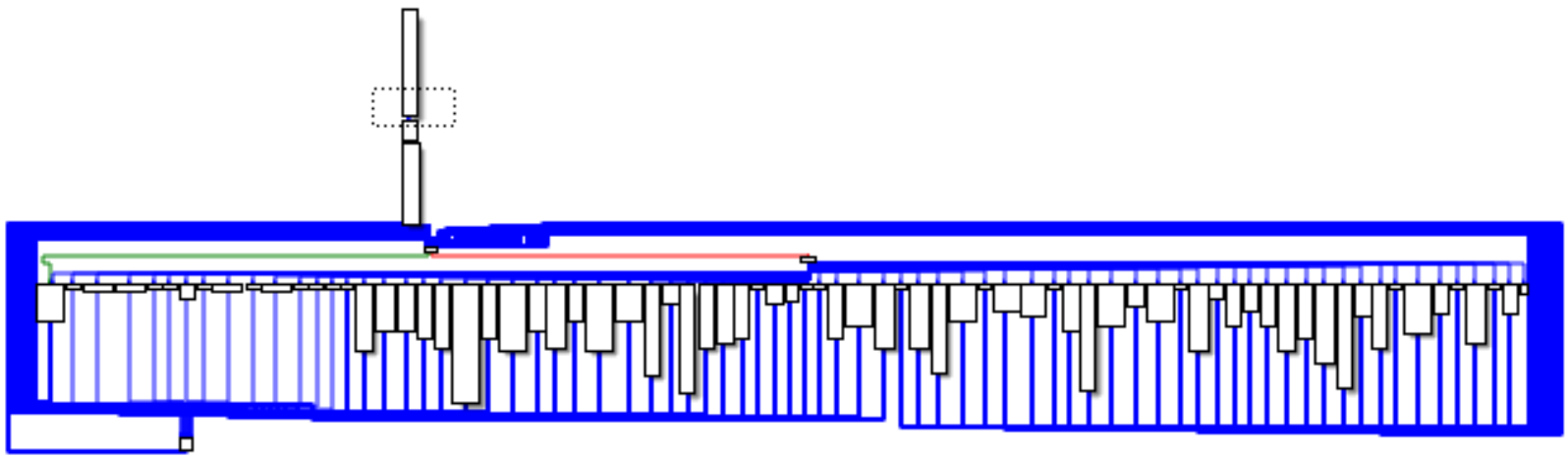
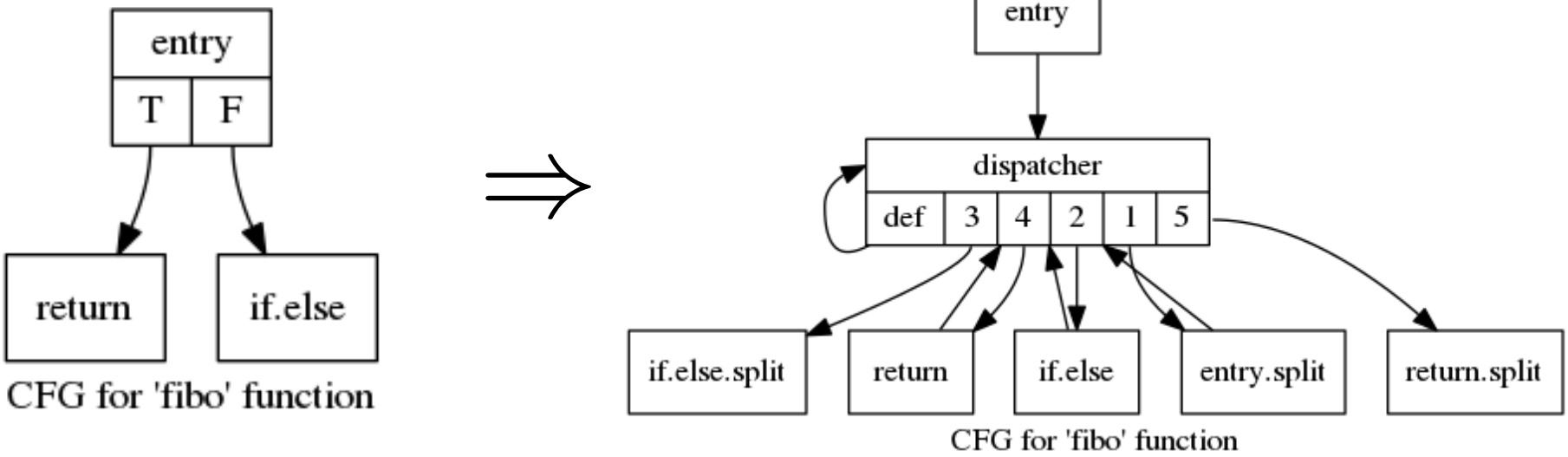


Illustration – protection de code (1)

CFG flattening



Principes

+ Structure du code implicite
+ Ajout de code inutile
- Exécution ralentie
- Augmentation de la taille du binaire

Illustration – protection de code (2)

Mixed Boolean-Arithmetic

```
int f(int x) {
    x = (0xe5*x + 0xF7) % 0x100;
    v1 = 0x0;
    v2 = 0xFE;
    v0 = (x&0xFF + ( v1 << 8)&0xFFFFFFFF);
    v3 = (((((v0*0xFFFFFE26)+0x55)&v2)+(v0*0xED)+0xD6)&0xFF&0xFF + ( v1 << 8)&0xFFFFFFFF);
    v4 = ((((((- (v3*0x2))+0xFF)&v2)+v3)*0xE587A503)+0xB717A54D);
    v5 = (((((v4*0xAD17DB56)+0x60BA9824)&0xFFFFFF46)*0xA57C144B)+(v4*0xE09C02E7)+0xB5ED2776);
    v7 = ((((v5*0xC463D53A)+0x3C8878AF)&0xCC44B4F4)+(v5*0x1DCE1563)+0xFB99692E);
    v6 = (v7&0x94);
    v8 = ((((v6+v6+(- (v7&0xFF&0xFF + ( v1 << 8)&0xFFFFFFFF)))*0x67000000)+0xD000000) >> 0x18);
    result = ((v8*0xFFFFB22D)+(((v8*0xAE)|0x22)*0xE5)+0xC2)&0xFF & 0xFFFFFFFF;
    result = (0xed*(result-0xF7)) % 0x100;

    return result;
}
```

Illustration – protection de code (2)

Mixed Boolean-Arithmetic

$a \oplus b$

$$2 \times (a \vee b) - ((a \oplus b) + 2 \times (a \wedge b))$$

```
%0 = xor i32 %b, %a
```

$\Longrightarrow$

```
%0 = or i32 %b, %a
%1 = mul i32 2, %0
%2 = xor i32 %b, %a
%3 = and i32 %b, %a
%4 = mul i32 2, %3
%5 = add i32 %2, %4
%6 = sub i32 %1, %5
```

Principes

+ Opérateur masqué
+ Résiste aux solveurs modernes

- Exécution ralentie
- Augmentation de la taille du binaire

Illustration – protection de données (0)

Transformation des données en mémoire

Étape 0

Encoder les données statiques :

$$\forall i \in [0, n-1], \text{data}[i] \rightarrow \text{data}[i] + C$$

Étape 1

À chaque lecture :

$$x = \text{data}[i] \rightarrow x = \text{data}[i] - C$$

Étape 2

À chaque écriture :

$$\text{data}[i] = x \rightarrow \text{data}[i] = x + C$$

Illustration – protection de données (1)

Transformation des données en mémoire

Données « obfusquées » en mémoire, mais pas dans les calculs intermédiaires (chiffrement homomorphe difficile. . . mais partiel possible !)

Principes

+ Pas de données claires sur le disque
+ Pas de données claires en RAM !
- Ralentissement de l'exécution
- Difficile à prouver

Générateur de Machines virtuelles

Soit :

$\sigma \in \Sigma$ Un état mémoire

$\mathcal{L}_0$ Un langage

$\mathcal{L}_1$ Un autre langage

$f : \mathcal{L}_0 \to \mathcal{L}_1$ Un convertisseur d'expression

$\mathrm{VM} : \mathcal{L}_1 \to \Sigma \to \Sigma$ un interpreteur de $\mathcal{L}_1$ écrit en $\mathcal{L}_0$

On peut alors executer une expression $\mathrm{expr}_0 : \Sigma \to \Sigma$ écrite dans $\mathcal{L}_0$ en utilisant:

$$\mathrm{expr}_0(\sigma) = \mathrm{VM}(f(\mathrm{expr}_0), \sigma)$$

Propriétés d'une exécution sur une VM

+ Protection du code
+ Protection ddes données internes
+ Composable
  - Potentiellement très lourd
  - Tout ou rien : le *reverse* de la VM donne tout le code

Dans la nature sauvage : Interpréteur Python chargeant du code natif qui compile une requète SQL qui...
$\Rightarrow$ dans ce cas, les données sont également converties à chaque étape

Obfuscation et licence

a.k.a comment montrer que deux code sont similaires

Similitude

- CFG
- CG
- Chaînes / données utilisées
- Densité de probabilité
- (d'instruction)
- Trace d'exécution
- Watermarking
- ...

Cas pratique : Calcul de *birthmark*

Une finition, des finitions

Une *birthmark* est une trace de l'exécution d'un programme qui se veut caractéristique de ce dernier. Elle doit résister au mieux à l'obfuscation.

Exo (squelette)

Examinez l'aide de la commande `strace` et proposez une méthode pour calculer une *birthmark* d'un programme. Vérifiez votre hypothèse en calculant la *birthmark* de l'interpréteur `lua` en comparant la version système avec une version compilée par vos soins.

Obfuscation toujours possible ?

Que fait ce code ?

```
main() {
char code[] =
"main()␣{␣char␣code[]␣=␣%c%s%c;␣printf(code,␣34,␣code,␣34);␣}";
printf(code, 34, code, 34);
}
```

For the lulz (0) — OCCC

```
#include                                  <math.h>
#include                                <sys/time.h>
#include                                <X11/Xlib.h>
#include                               <X11/keysym.h>
                                       double L ,o ,P
                                      ,_=dt,T,Z,D=1,d,
                                      s[999],E,h= 8,I,
                                      J,K,w[999],M,m,O
                                     ,n[999],j=33e-3,i=
                                     1E3,r,t, u,v ,W,S=
                                     74.5,l=221,X=7.26,
                                     a,B,A=32.2,c, F,H;
                                     int N,q, C, y,p,U;
                                    Window z; char f[52]
                                  ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                  XEvent z; XNextEvent(e ,&z);
                                      ++*((N=XLookupKeysym
                                        (&z.xkey,0))-IT?
                                        N-LT? UP-N?& E:&
                                        J:& u: &h); --*(
                                        DN -N? N-DT ?N==
                                        RT?&u: & W:&h:&J
                                         ); } m=15*F/l;
                                         c+=(I=M/ l,l*H
                                         +I*M+a*X)*_; H
                                         =A*r+v*X-F*l+(
                                         E=.1+X*4.9/l,t
                                         =T*m/32-I*T/24
                                          )/S; K=F*M+(
                                          h* 1e4/l-(T+
                                          E*5*T*E)/3e2
                                          )/S-X*d-B*A;
                                          a=2.63 /l*d;
                                          X+=( d*l-T/S
                                           *(.19*E +a
                                           *.64+J/1e3
                                           )-M* v +A*
                                           Z)*_; l +=
                                           K *_; W=d;
                                           sprintf(f,
                                           "%5d␣␣%3d"
                                           "%7d",p =l
                                           /1.7,(C=9E3+
                                 O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                                X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                                *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                                179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                 W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                  )/107e2)*_; D=cos(o); E=sin(o); } }
```

For the lulz (1) — http://www.sourcecodepoetry.com/

```
long long time, ago,
i, can, still, remember, how;
typedef struct s{} was,
all, we, had;

s o,  I, knew, If=I; had my, chance =
I, could, code, a, perfect, prance;

s ee; was ruling;
we were, happy
,good ,ole, times;

all that, changed, when; class es{} came;
we got, spoiled, And, thats = a, shame;

all is, broken, nothing;s same
,c_plus_plus, has, killed, the, flame;

had We, believed, in, rocknroll=
could,ve, coding, cured, our, mortal, souls;

we met=a, girl, who, sang= the, blues;
we asked, her, For, some= happy, news;

s he, said, to, me, with, pretty, smile
=If, you, are; main(){
    return
        the,
        time;
}
```

Conclusion

« Plus claire la lumière, plus sombre l'obscurité... Il est impossible d'apprécier correctement la lumière sans connaître les ténèbres. »

Jean-Paul Sartre

Conclusion

Obfusquer, c'est :

- **M**uter
- **D**iluer
- **R**ecomposer

En conservant le sens d'origine...

Conclusion

Obfusquer, c'est :

- Muter
- Diluer
- Recomposer

En conservant le sens d'origine...

Ce n'est pas une protection absolue

Bonus

http://serge-sans-paille.github.io/talks/hack.lu-2014-10-21.html