Outils d'expérimentation

(Sécurité : C++ vs Java)

Document présenté à :
M. Dominique Goutier

Dans le cadre du cours :
Nouvelles technologies (420-620-SF)

Étudiants :
David Dastous St-Hilaire
Simon Ducas-Desrosiers

Cégep de Sainte-Foy, département d'informatique
Mercredi le 7 février 2007

Table des matières

1. La documentation technique des outils d'expérimentation

1.1 Devis du protocole d'expérimentation mis à jour

(Voir document en annexe)

1.2 Le guide d'installation des outils d'expérimentation

Programme expérimental :

C++ : Il s'agit seulement d'un exécutable, on double clique dessus et il fonctionne.

Java : Il s'agit du code source et d'un fichier class, on l'importe avec éclipse dans son workspace, ensuite on clique sur le fichier .java et on clique sur « run as an application»

Décompilation :

odbg110 : Il s'agit d'un fichier zip qu'on décompresse avec winzip et double clique sur l'exécutable.

PVDasm.v1.6d : Il s'agit d'un fichier zip qu'on décompresse avec winzip et double clique sur l'exécutable.

Rec : Il s'agit d'un fichier zip qu'on décompresse avec winzip et double clique sur l'exécutable.

DjJava : Il s'agit d'un fichier qui installe tout ce dont on a besoin pour s'exécuter. Il suffit de remplir les champs convenablement et d'appuyer sur suivant.

Obfuscation :

Stunnix C,C++ Obfuscator : Il s'agit d'un fichier qui installe tout ce dont on a besoin pour s'exécuter. Il suffit de remplir les champs convenablement et d'appuyer sur suivant.

ProGuard 3.7 : Il s'agit d'un fichier zip qu'on décompresse avec winzip et double clique sur l'exécutable.

1.3 Le guide d'utilisation des outils d'expérimentation

Programme maison :

C++ : Appuyer sur les touches au clavier quand le programme le demande.

Java : Appuyer sur les touches au clavier quand le programme le demande.

Décompilation :

odbg110 : Le manuel d'utilisation est fourni avec le programme et accessible à l'intérieur de celui-ci. Par ailleurs, on peut le trouver dans le répertoire où il a été décompressé.

PVDasm.v1.6d : Le manuel d'utilisation est fourni avec le programme et accessible à l'intérieur de celui-ci. Par ailleurs, on peut le trouver dans le répertoire où il a été décompressé.

Rec : Le manuel d'utilisation est fourni avec le programme et accessible à l'intérieur de celui-ci. Par ailleurs, on peut le trouver dans le répertoire où il a été décompressé.

djJava : Le manuel d'utilisation est fourni avec le programme et accessible à l'intérieur de celui-ci. Par ailleurs, on peut le trouver dans le répertoire où il a été installé.

Obfuscation :

Stunnix C,C++ Obfuscator : Le manuel d'utilisation est fourni avec le programme. Il est inclus aussi au site web local, dans lequel on créer nos projets à obfusquer.

ProGuard 3.7 : Le manuel d'utilisation est fourni avec le programme. Par ailleurs, on peut le trouver dans le répertoire où il a été décompressé.

2. Les programmes

2.1 Tous les programmes source documentés

C++ :

Main.cpp :

```
#include "main.h"

//Entry point
int main(int argc, char *argv[])
{
	int chs =0;
	cout << "Menu For option, enter your choice" << endl;
	cout << "1. Number Module" << endl;
	cout << "2. String Module" << endl;
	cout << "Q pour quitter" << endl;
	cin >> chs;

	switch (chs)
	{
	case 1:
		Number();
		break;
	case 2:
		Alphabet();
		break;
	}
	system("cls");
	cout << "Press any key to exit";
	cin.clear();
	cin.get();
	cin.get();
}

//GEt a sentence from user
void Alphabet()
{
	string strUser;
	cout << "Enter you sentence here : " << endl;
	cin >> strUser;
	cout << strUser;
	reverseString(strUser);
	cout << strUser;
}
//reverse a string
void reverseString(string& _str)
{
    int i=0;
	int size=_str.size();
	while( i < (size / 2))
	{
		swap(_str[i], _str[size - i]);
		i++;
	}
}
//Get a table of number
void Number()
{
	int size=0;
	cout << "Enter a number to choose how many, you will enter : ";
	cin >> size;
```

```
        cout << endl;
        int* tabInt = new int[size];
        getNumbers(tabInt, size);
        ShowTab(tabInt, size);
        SortTab(tabInt, size);
        ShowTab(tabInt, size);
}

//get numbers
void getNumbers(int* tabInt, const int& p_size)
{
        int i = 0;
        for(i=0;i<p_size;i++)
        {
                cout << i << "  nombre : ";
                cin >> tabInt[i];
                cout << endl;
        }
}

//Sorting of number
void SortTab(int* tabInt, const int& p_size)
{
        int tempSize = p_size;
        tempSize--;
  for (int i = 0; i < tempSize; i++)
    for (int j = tempSize; j > i; j--)
      if ( tabInt[j] < tabInt[j-1] )
                      swap( tabInt[j], tabInt[j-1] );
}

//Display table
void ShowTab(int* tabInt, const int& p_size)
{
        int i = 0;
        for(i=0;i<p_size;i++)
        {
                cout << "|" << tabInt[i] << "|";
        }
        cout << "\n" << endl;
}
```

Main.h :

```
#ifndef MAIN_H
#define MAIN_H

#include <iostream>
#include <string>
#include <limits>
using namespace std;

void Alphabet();
void reverseString(string&);
void Number();
void getNumbers(int*, const int&);
void ShowTab(int*, const int& );
void SortTab(int*, const int&);

#endif
```

Java :

```
package bin.core;
import java.io.*;
/**
 * @author David Dastous ST-Hilaire, Simon Ducas Desrosier.
 */
public class entry {

	// Entry point
	public static void main(String[] args) {
		System.out.println("Menu For option, enter your choice");
		System.out.println("1. Number Module");
		System.out.println("2. String Module");
		try
		{
			//Look for input of the user
			DataInputStream in = new DataInputStream(System.in);
			int i = in.readUnsignedByte();
			switch (i)
			{
			case 49:
				getNumber();

			case 50:
				Alphabet();
			}
		}
		catch(Exception ex)
		{
			System.out.println("Erreur" + ex.getMessage());
		}
	}

	//Get a string and reverse it
	private static void Alphabet()
	{
		String strUser ="";
		System.out.println("Enter you sentence here : ");
		try{
			int i = 0;
			byte name[] = new byte[100];
			i = System.in.read(name);
			strUser = name.toString();
		}catch(Exception ex) {}
		System.out.println(strUser);
		strUser = reverseString(strUser);
		System.out.println(strUser);

	}

	//Reverse a string
	private static String reverseString(String _str)
	{
	    int i=0;
	    char[] str = new char[_str.length()];
		while( i < _str.length())
		{
			str[i] = _str.charAt(_str.length() - i);
			i++;
		}
		return str.toString();

	}

	//Get a given number of numbers and sort them
	private static void getNumber()
	{
```

```
		int size=0;
		System.out.println("Enter a number to choose how many you will enter : ");
		try{
		size = System.in.read();
		}catch(Exception ex) {}
		int[] tabInt = new int[size];
		getNumbers(tabInt, size);
		ShowTab(tabInt, size);
		SortTab(tabInt, size);
		ShowTab(tabInt, size);
	}

	//Get nubmers form the user
	private static void getNumbers(int[] tabInt, int p_size)
	{
		int i = 0;
		for(i=0;i<p_size;i++)
		{
			System.out.println(i + "  nombre : ");
			try{
			tabInt[i] = System.in.read();
			}catch(Exception ex) {}
		}
	}

	//Sorting numbers
	private static void SortTab(int[] tabInt, int p_size)
	{
		int tempSize = p_size;
		tempSize--;
	  for (int i = 0; i < tempSize; i++)
	    for (int j = tempSize; j > i; j--)
	      if ( tabInt[j] < tabInt[j-1] )
	      {
	       int t=tabInt[j];
	       tabInt[j]=tabInt[j-1];
	       tabInt[j-1]=t;
	      }
	}

	//Display the table of number
	private static void ShowTab(int[] tabInt, int p_size)
	{
		int i = 0;
		for(i=0;i<p_size;i++)
		{
			System.out.println("|" + tabInt[i] + "|");
		}
	}
}
```

2.4 Les logiciels nécessaires à la réalisation de l'expérimentation

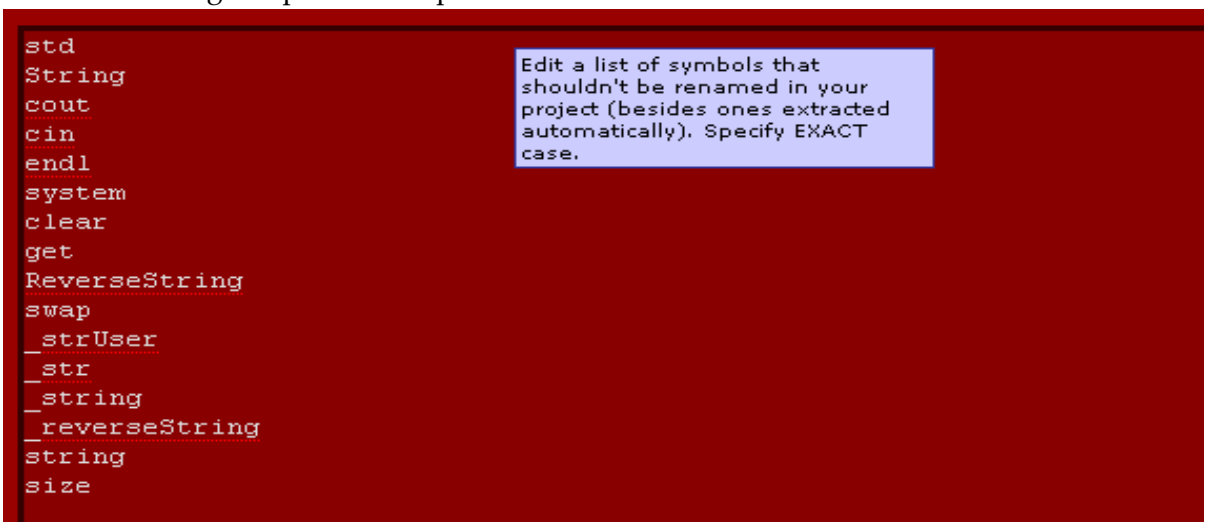
- Eclipse
- Java virtual machine
- Rec
- OllyDbg (v1.10)
- PVDasm.v1.6d
- djJava
- Stunnix C,C++ Obfuscator
- ProGuard 3.7

3. Les résultats de votre expérimentation

3.1 Les captures ou de rapports produits par l'expérimentation.

Obfuscation C++ :

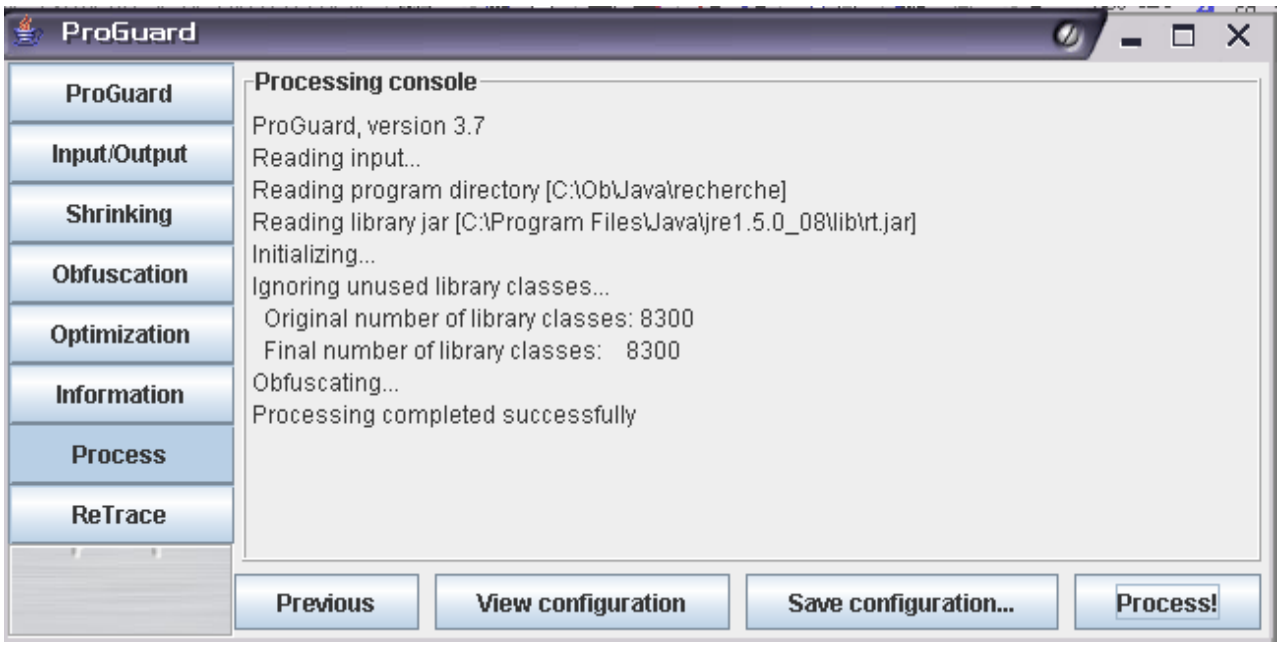
Il fut long et laborieux avec le programme que nous avions d'obfuscer notre programme c++. D'autres sont payants, mais croûtent très j'imagine qu'ils sont plus faciles à utiliser. Voici la liste des mots que nous avons dû exclure de l'obfuscation pour un code d'environ 100 lignes qui n'utilise pas les classes.



Le processus d'obfuscation avec ce programme est laborieux et long, mais semble efficace lorsqu'il est fait et fonctionnel.

Obfuscation Java :

Le programme est plus facile d'utilisation, mais quand il y une erreur on ne comprend pas vraiment la nature de l'erreur et comment elle se produit, une procédure d'essais et erreur est donc requise. Par contre, le programme est gratuit et intuitif. De plus, il semble offrir plus d'options que l'autre que nous avons essayé.

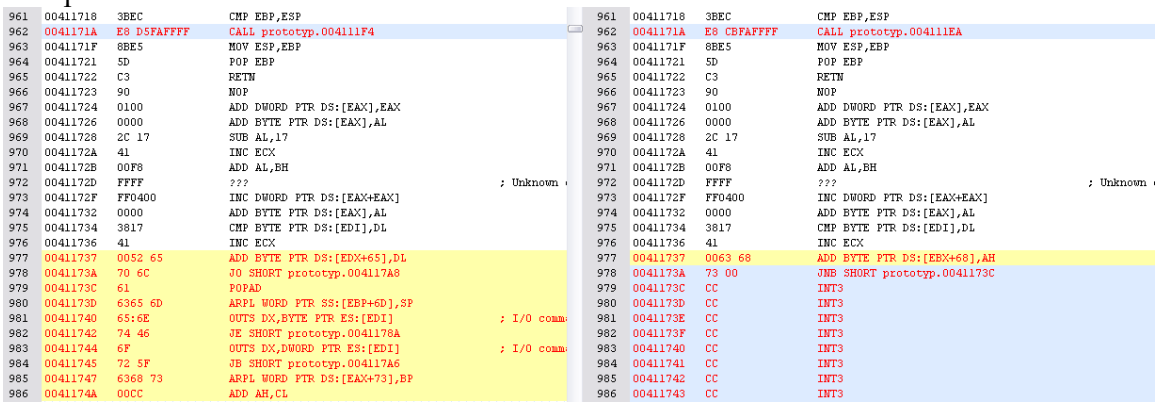


3.2 Un constat relatif à vos expérimentations.

Les résultats de l'expérimentation sont exclusivement composés des tests de décompilation des programmes prototypes.

Pour ce qui est de la version C++ du prototype, 2 logiciels de décompilation ont été utilisés. Le premier, PVDasm (version 1.6d), n'a pas été capable de décompiler la version obfusquée du prototype. Par contre, la version originale a été décompilée avec succès. Puisque le logiciel ne donne aucune information lorsqu'une erreur de décompilation survient, il a été impossible de déterminer avec précision la raison. Une bonne hypothèse serait que le point d'entrée n'est pas situé dans le code lui-même, mais plutôt vers la fin du fichier, comme si une fonction spéciale devait décompresser ou décoder le programme avant son exécution.

Le deuxième logiciel de décompilation, OllyDbg (version 1.10), a pu décompiler les deux programmes avec succès. Les différences résident dans l'adresse des points d'entrées (tous deux à l'extérieur de la zone de code) ainsi que dans le déroulement général du programme. Le code assembleur généré est quelque peu différent, comme en témoigne la copie d'écran ci-dessous :



```
961 00411718   3BEC          CMP EBP,ESP
962 0041171A   E8 D5FAFFFF   CALL prototyp.004111F4
963 0041171F   8BE5          MOV ESP,EBP
964 00411721   5D            POP EBP
965 00411722   C3            RETN
966 00411723   90            NOP
967 00411724   0100          ADD DWORD PTR DS:[EAX],EAX
968 00411726   0000          ADD BYTE PTR DS:[EAX],AL
969 00411728   2C 17         SUB AL,17
970 0041172A   41            INC ECX
971 0041172B   00F8          ADD AL,BH
972 0041172D   FFFF          ???                          ; Unknown
973 0041172F   FF0400        INC DWORD PTR DS:[EAX+EAX]
974 00411732   0000          ADD BYTE PTR DS:[EAX],AL
975 00411734   3817          CMP BYTE PTR DS:[EDI],DL
976 00411736   41            INC ECX
977 00411737   0052 65       ADD BYTE PTR DS:[EDX+65],DL
978 0041173A   70 6C         JO SHORT prototyp.004117A8
979 0041173C   61            POPAD
980 0041173D   6365 6D       ARPL WORD PTR SS:[EBP+6D],SP
981 00411740   65:6E         OUTS DX,BYTE PTR ES:[EDI]    ; I/O comm
982 00411742   74 46         JE SHORT prototyp.0041178A
983 00411744   6F            OUTS DX,DWORD PTR ES:[EDI]   ; I/O comm
984 00411745   72 5F         JB SHORT prototyp.004117A6
985 00411747   6368 73       ARPL WORD PTR DS:[EAX+73],BP
986 0041174A   00CC          ADD AH,CL

961 00411718   3BEC          CMP EBP,ESP
962 0041171A   E8 CBFAFFFF   CALL prototyp.004111EA
963 0041171F   8BE5          MOV ESP,EBP
964 00411721   5D            POP EBP
965 00411722   C3            RETN
966 00411723   90            NOP
967 00411724   0100          ADD DWORD PTR DS:[EAX],EAX
968 00411726   0000          ADD BYTE PTR DS:[EAX],AL
969 00411728   2C 17         SUB AL,17
970 0041172A   41            INC ECX
971 0041172B   00F8          ADD AL,BH
972 0041172D   FFFF          ???                          ; Unknown
973 0041172F   FF0400        INC DWORD PTR DS:[EAX+EAX]
974 00411732   0000          ADD BYTE PTR DS:[EAX],AL
975 00411734   3817          CMP BYTE PTR DS:[EDI],DL
976 00411736   41            INC ECX
977 00411737   0063 68       ADD BYTE PTR DS:[EBX+68],AH
978 0041173A   73 00         JNB SHORT prototyp.0041173C
979 0041173C   CC            INT3
980 0041173D   CC            INT3
981 0041173E   CC            INT3
982 0041173F   CC            INT3
983 00411740   CC            INT3
984 00411741   CC            INT3
985 00411742   CC            INT3
986 00411743   CC            INT3
```

Les instructions suivantes sont présentes en surplus à de nombreux endroits dans le code assembleur du programme obfusqué :

```
0041173C   POPAD
0041173D   ARPL WORD PTR SS:[EBP+6D],SP
00411740   OUTS DX,BYTE PTR ES:[EDI]
00411742   JE SHORT prototyp.0041178A
00411744   OUTS DX,DWORD PTR ES:[EDI]
00411745   JB SHORT prototyp.004117A6
00411747   ARPL WORD PTR DS:[EAX+73],BP
0041174A   ADD AH,CL
```

Pour ce qui est du code Java, il en est un peu différent. Puisqu'en pratique tout fichier .class doit rendre accessible le nom des fonctions et des membres publics afin de pouvoir être interprété par la machine virtuelle Java, la décompilation dans ce cas représente un

défi moindre. Tout de même, le code restitué via DJ Java (version 3.9.9.91) est plus difficile à comprendre dans la version obfusquée. Voici un exemple du code d'une fonction simple, obtenue par décompilation :

Code résultant de la décompilation de l'application originale :

```
    private static void ShowTab(int tabInt[], int p_size)
    {
        int i = 0;
        for(i = 0; i < p_size; i++)
            System.out.println("|" + tabInt[i] + "|");

    }
```

Code résultant de la décompilation de l'application obfusquée :

```
    private static void _mthfor(int ai[], int i)
    {
        boolean flag = false;
        for(int j = 0; j < i; j++)
            System.out.println("|" + ai[j] + "|");

    }
```

À première vue, le code semble facile à comprendre. Mais pour un utilisateur non averti tentant de décompiler un gros programme, la tâche peut s'avérer très compliquée !

3.3 Votre journal d'expérimentation personnel mis à jour et détaillé.

(Voir feuilles suivantes pour feuilles de temps)