Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчет по дисциплине
«Проектирование ОС и их компонентов»

Обзор средств обфускации кода (C/C++) под Windows/Linux

Работу выполнил студент группы №: 13541/3 Чеботарёв М. М.
Работу принял преподаватель: _____ Душутина Е. В.

Санкт-Петербург
2017 г.

Используемая система и версия ядра

а) Windows

Процессор:	Intel(R) Core(TM) i5-2450 CPU @2.50GHz 2.50GHz
ОЗУ:	8,00 Гб
Тип системы:	Windows 7 Ultimate Compact (2009) Service Pack 1. x64.

б) Linux

```
michael@michael-LIFEBOOK-AH531:~$ lsb_release  -a
No LSB modules are available.
Distributor ID:         Ubuntu
Description:            Ubuntu 16.04.1 LTS
Release:                16.04
Codename:               xenial


michael@michael-LIFEBOOK-AH531:~$ cat /proc/version
Linux version 4.4.0-38-generic (buildd@lgw01-58) (gcc version 5.4.0 20160609
(Ubuntu 5.4.0-6ubuntu1~16.04.2) ) #57-Ubuntu SMP Tue Sep 6 15:42:33 UTC 2016
```

1. ОБФУСКАЦИЯ КОДА

Обфускация - это процесс, в результате которого код программы приобретает вид, трудный для анализа. Обфускация осуществляется с целью защиты программного кода и алгоритмов, которые он реализует.

Существует несколько разных видов обфускации в зависимости от того, какой именно код запутывается при ней. Для тех языков программирования, для которых программа представляется прямо в виде **исходных текстов**, используются специальные методы, делающие код нечитабельным для человека, но приемлемым для интерпретатора:

- запись программы в одну строку;
- замена имён переменных;
- замена имён функций;
- вставка ничего не значащих комментариев.

Для обфускации **двоичного кода** (как для исполняемых файлов, так и двоичного кода виртуальных машин .NET и Java) используются уже более серьёзные методы, в том числе и шифрование.

Специальные программы, производящие обфускацию, называемые обфускаторами, которые решают поставленную задачу по-разному. Например, одни изменяют исходные тексты (удаляют комментарии, дают переменным бессмысленные имена, шифруют строковые константы и т.д.), другие изменяют байт-код виртуальных машин Java и .NET, что технически сделать намного труднее. Более развитые обфускаторы изменяют машинный код, вставляя бессмысленные инструкциями. Кроме того они могут делать структурные или математические преобразования, изменяющие программу до неузнаваемости.

Запутывающие преобразования можно разделить на несколько групп в зависимости от того, на трансформацию какой из компонент программы они нацелены. На основе сказанного можно выделить основные методы обфускации:

- преобразования на уровне **исходного текста**;
- преобразования форматирования, которые изменяют только **внешний вид программы.**
- преобразования структур данных, изменяющие **структуры данных**, с которыми работает программа;
- преобразования потока управления программы, которые изменяют структуру графа **потока управления**.

1.1. Преобразования потока управления.

Преобразования потока управления изменяют граф потока управления одной функции. Они могут приводить к созданию в программе новых функций. Краткая характеристика методов приведена ниже.

- Открытая вставка функций заключается в том, что тело **функции подставляется в точку вызова функции**;
- Вынос группы операторов**.** Это **обратное преобразование к предыдущему** и хорошо дополняет его. Некоторая группа операторов исходной программы выделяется в отдельную функцию. При необходимости создаются формальные параметры;
- Внесение **недостижимого** кода;
- Внесение **мёртвого** кода. В отличие от недостижимого кода, мёртвый код в программе **выполняется, но никак не влияет** на результат работы программы;
- Внесение **избыточного** кода. Избыточный код, в отличие от мёртвого кода выполняется, и результат его выполнения используется в дальнейшем в программе, но такой код можно упростить или совсем удалить, так как вычисляется либо константное значение, либо значение, уже вычисленное ранее;
- Устранение библиотечных вызовов;
- Переплетение функции. Идея этого запутывающего преобразования в том, что **несколько функций объединяются в одну** функцию. Списки параметров исходных функций объединяются, и **добавляется один управляющий параметр**, который позволяет определить, какая функция в действительности выполняется;
- Клонирование функций. При обратной инженерии функций в первую очередь изучается сигнатура функции, а также то, как эта функция используется, в каких местах программы, с какими параметрами и в каком окружении вызывается. Анализ контекста использования функции можно затруднить, если **каждый вызов некоторой функции будет выглядеть как вызов какой-то другой, каждый раз новой функции;**
- Развёртка циклов. Развёртка циклов заключается в том, что тело цикла размножается два или более раз.

1.2. Преобразования форматирования

К преобразованиям форматирования относятся удаление комментариев, переформатирование программы, удаление отладочной информации, изменение имён идентификаторов.

- **Удаление комментариев и переформатирование** программы применимы, когда запутывание выполняется на уровне исходного кода программы. Эти преобразования не требуют только лексического анализа программы. При переформатировании программы исходное форматирование теряется безвозвратно.

- **Удаление отладочной информации** приводит к тому, что имена локальных переменных становятся невосстановимы.

- **Изменение имён** локальных **переменных** требует семантического анализа в пределах одной функции. Изменение имён всех переменных и **функций** программы помимо полной привязки имён в каждой единице компиляции требует анализа межмодульных связей. Имена, определённые в программе и не используемые во внешних библиотеках, могут быть изменены произвольным, но согласованным во всех единицах компиляции образом, в то время как имена библиотечных переменных и функций меняться не могут.

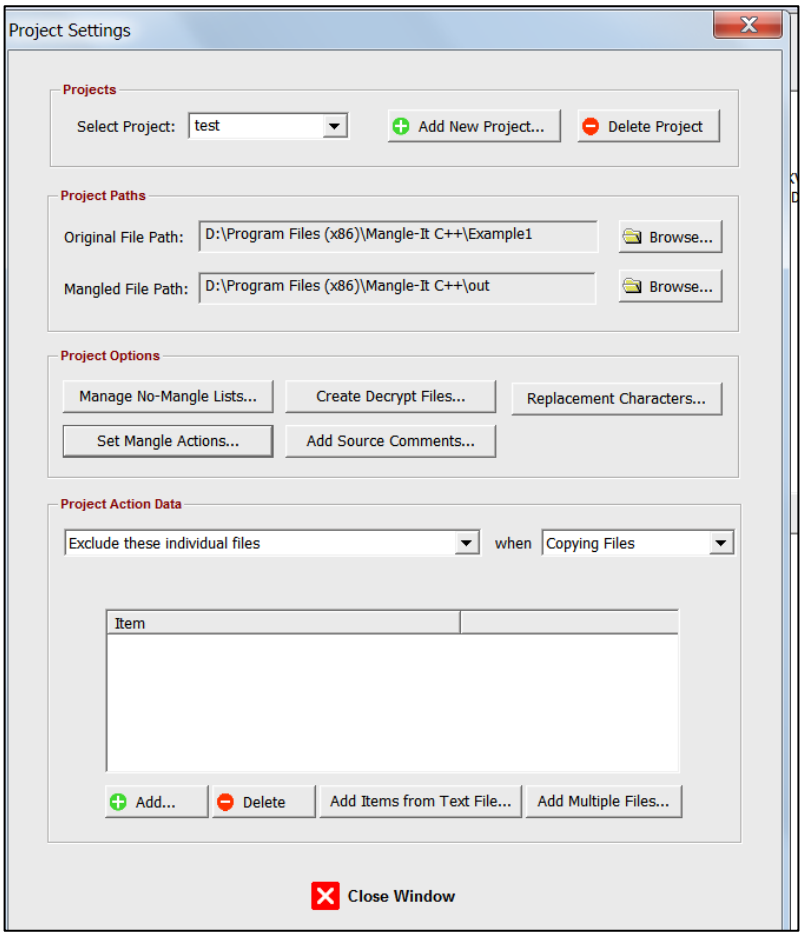
2. ИСПОЛЬЗОВАНИЕ ОБФУСКАТОРОВ

В настоящей работе было рассмотрено много различных обфускаторов. Однако возникла проблема в том, что для языка C/C++ в большинстве случаях обфускаторы является коммерческими продуктами. Свободно распространяемыми являются в основном ограниченные версии обфускаторов, которые не защищают должным образом исходный код, а являются лишь примером для ознакомления с GUI или консолью данного ПО.

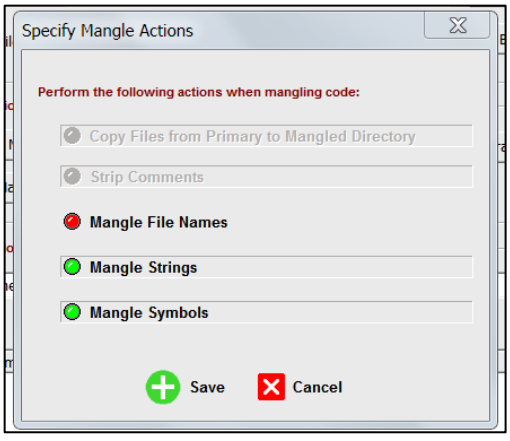
2.1. Mangle-it c++ (2010)

Обфускатор Mangle-it c++ – обфускатор программ на языке C++. Данное ПО представляет пользователю удобный графический интерфейс.
Создадим новый проект, укажем в нем путь к директории, содержащей исходный коды и путь к директории для обфусцированных файлов.

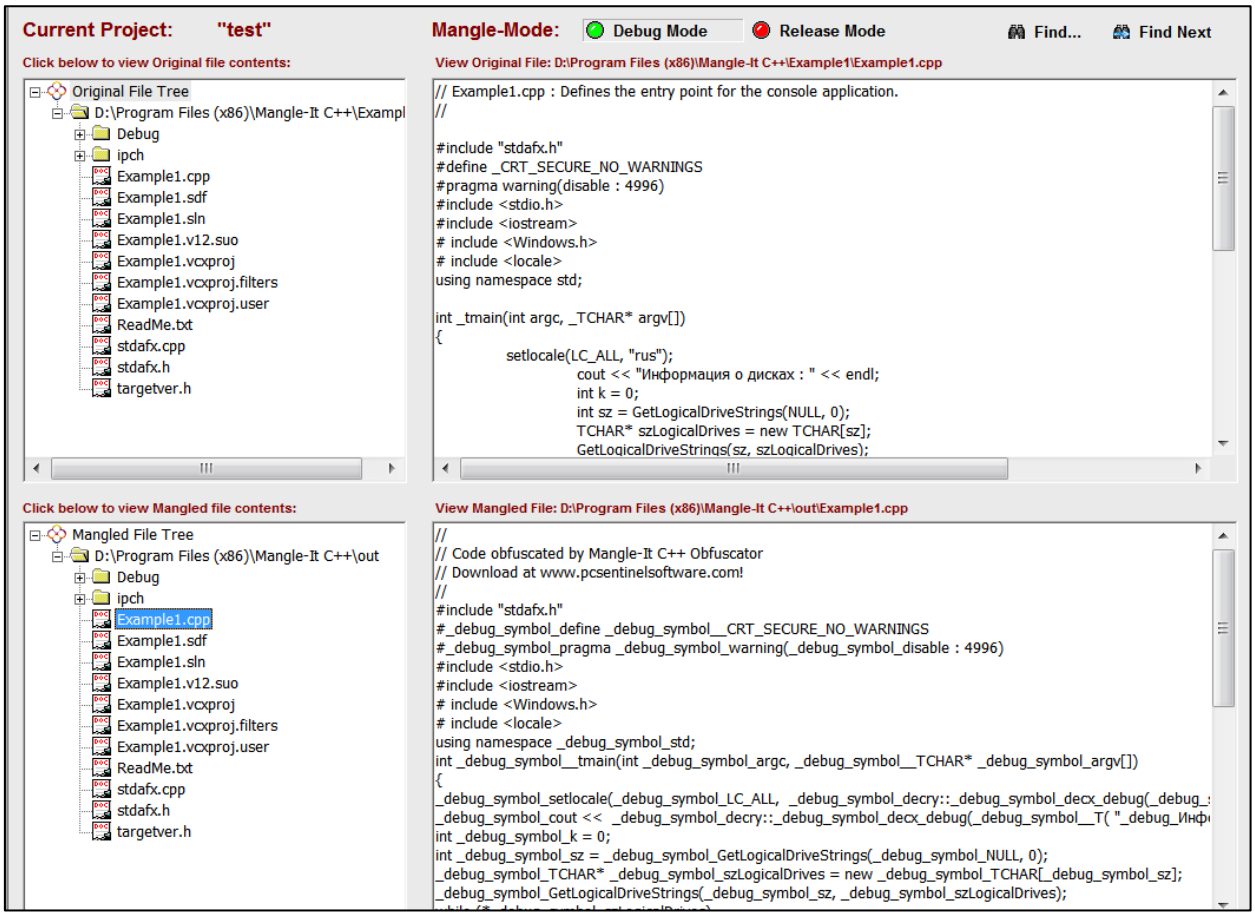
Рисунок 1. Mangle



Изменим настройки обфускации, выбрав изменение строк и символов.



После успешной обфускации, просмотрим коды полученных файлов и сравним их с исходными.



ДО	ПОСЛЕ

#include <iostream>
#include <string>
 	
using namespace std;
 	
bool password_is_valid (string password)
{
string valid_pass = "qwerty123";
if (valid_pass == password)
return true;
else
return false;
}
 	
void get_pass ()
{
string user_pass;
cout << "Введите пароль: ";
getline(cin, user_pass);
if (!password_is_valid(user_pass)) {
cout << "Неверный пароль!" <<
	/*Mangle-It C++ Source Code Obfuscator*/
	#include <string>
	using namespace std;
	bool _debug_symbol_password_is_valid (string password)
	{
	string _debug_symbol_valid_pass = decrypt::_debug_symbol_dec_debug(_T("_debu
	if (_debug_symbol_valid_pass == password)
	return true;
	else
	return false;
	}
	void _debug_symbol_get_pass ()
	{
	string _debug_symbol_user_pass;
	_debug_symbol_cout << decrypt::_debug_symbol_dec_debug(_T("_debug_Введите
	getline(_debug_symbol_cin, _debug_symbol_user_pass);
	if (!_debug_symbol_password_is_valid(_debug_symbol_user_pass)) {
	_debug_symbol_cout << decrypt::_debug_symbol_dec_debug(_T("_debug_Неверны
	_debug_symbol_get_pass ();
	} else {

Среди осуществленных преобразований**: удаление комментариев** и замена указанных символов и функций;

Также все переменные, типы, вызовы функций были дополнены вставкой в начало имени строки _debug_symbol_. Данные преобразования делают код программы менее читаемым, однако, стоит заметить, что при желании программу легко привести к исходному виду даже вручную.

2.2. CXX-obfus (2017)

Обфускатор является кросс-платформенным, поддерживаемые ОС: **Windows, Linux, Mac.**

2.2.1. Тестирование в Windows.

Окно создания тестового проекта для указания путей к входной, выходной и вспомогательной директориям.

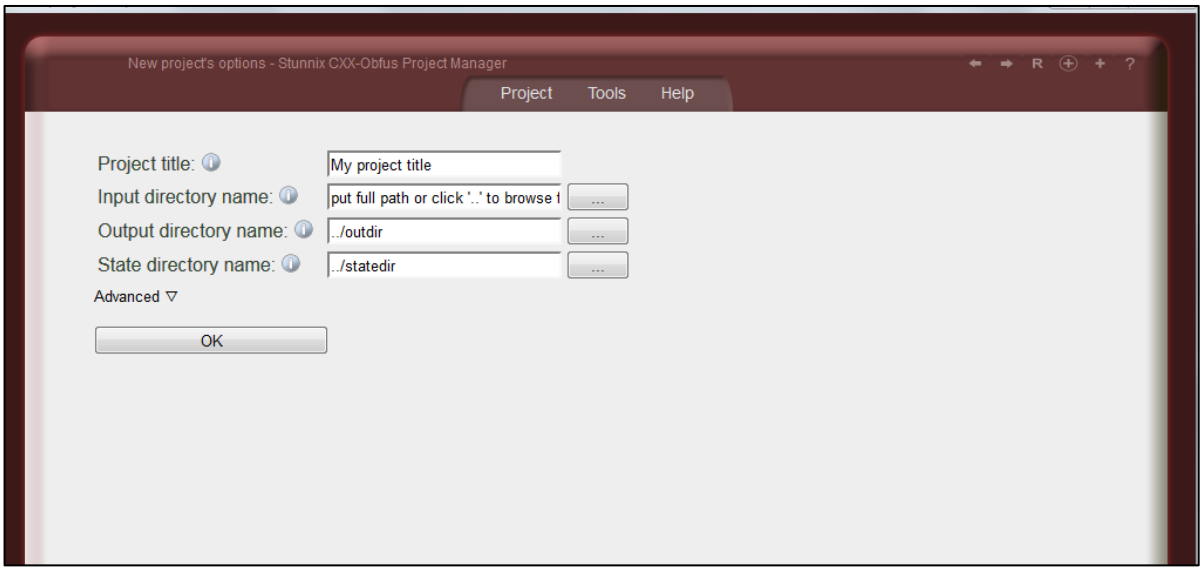


Рисунок 2. Указание параметров

После успешной обфускации просмотрим коды.

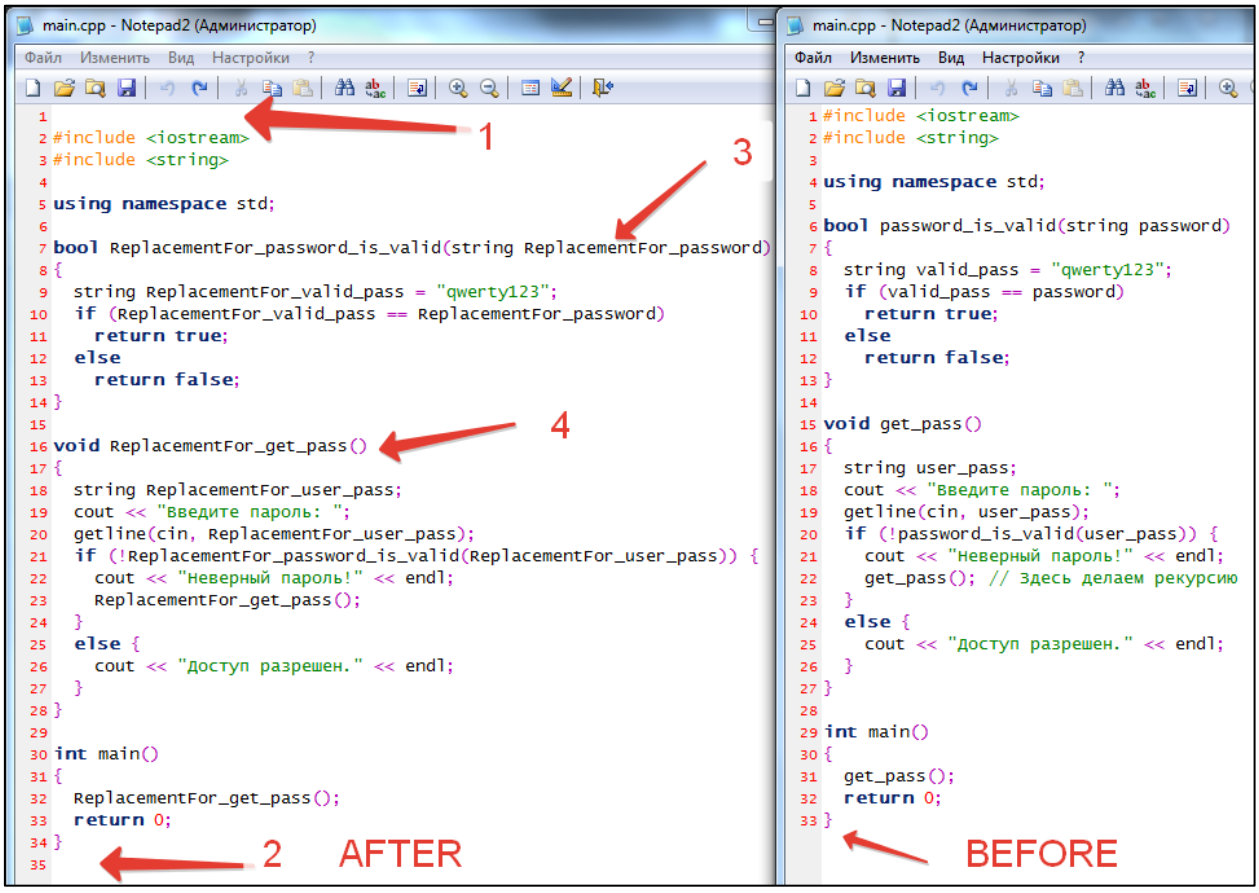


Рисунок 3. До и После обфускации

Данный обфускатор применил похожие предыдущему случаю преобразования.

- 1,2 - добавление пустых строк;
- 3 – замена имен переменных;
- 4 – замена имен функций;

Имена переменных были изменены, путем добавления в начало имени строки ReplacementFor_ .

2.2.2. Тестирование в Linux.

Листинг 1.1. Исходных код №2

```
#include <stdio.h>
#include <sys/stat.h>
#include <sys/vfs.h>
int main ()
{
	struct statfs fs;
	unsigned long long free, size,blocks;

	statfs("/home/user/Desktop/fstat/file_name.txt", &fs);
	free = fs.f_bfree;
	size = fs.f_bsize;
	blocks=fs.f_blocks;
	double blocks_gb=blocks*size/1000000000.;
	double busy_gb=(blocks-free)*size/1000000000.;
	double free_gb=free*size/1000000000.;
	printf("Total number of bytes -> %.4f \n", blocks_gb);
	printf("Number of free bytes -> %.4f \n", free_gb);
	printf("Number of busy bytes -> %.4f \n", busy_gb);
	return 1;
}
```

Листинг 1.2. Обфусцированный код №2

```
#include <stdio.h>
#include <sys/stat.h>
#include <sys/vfs.h>
int main(){struct ReplacementFor_statfs ReplacementFor_fs;unsigned long long
ReplacementFor_free,ReplacementFor_size,ReplacementFor_blocks;
ReplacementFor_statfs(
"\x2f\x68\x6f\x6d\x65\x2f\x75\x73\x65\x72\x2f\x44\x65\x73\x6b\x74\x6f\x70\x2f\x66\x73\x74\
x61\x74\x2f\x66\x69\x6c\x65\x5f\x6e\x61\x6d\x65\x2e\x74\x78\x74"
,&ReplacementFor_fs);ReplacementFor_free=ReplacementFor_fs.
ReplacementFor_f_bfree;ReplacementFor_size=ReplacementFor_fs.
ReplacementFor_f_bsize;ReplacementFor_blocks=ReplacementFor_fs.
ReplacementFor_f_blocks;double ReplacementFor_blocks_gb=ReplacementFor_blocks*
ReplacementFor_size/1000000000.;double ReplacementFor_busy_gb=(
ReplacementFor_blocks-ReplacementFor_free)*ReplacementFor_size/1000000000.;
double ReplacementFor_free_gb=ReplacementFor_free*ReplacementFor_size/
1000000000.;ReplacementFor_printf(
"\x54\x6f\x74\x61\x6c\x20\x6e\x75\x6d\x62\x65\x72\x20\x6f\x66\x20\x62\x79\x74\x65\x73\x2
0\x2d\x3e\x20\x25\x2e\x34\x66\x20" "\n"
,ReplacementFor_blocks_gb);ReplacementFor_printf(
```

```
"\x4e\x75\x6d\x62\x65\x72\x20\x6f\x66\x20\x66\x72\x65\x65\x20\x62\x79\x74\x65\x73\x20\x2
d\x3e\x20\x25\x2e\x34\x66\x20" "\n"
,ReplacementFor_free_gb);ReplacementFor_printf(
"\x4e\x75\x6d\x62\x65\x72\x20\x6f\x66\x20\x62\x75\x73\x79\x20\x62\x79\x74\x65\x73\x20\x2
d\x3e\x20\x25\x2e\x34\x66\x20" "\n"
,ReplacementFor_busy_gb);int ReplacementFor_c=ReplacementFor_getchar();return
(0xe9+6746-0x1b43);}
```

Были проведены следующие преобразования:

- Изменены названия переменных добавлением строки ReplacementFor_;
- Изменены названия функций добавлением строки ReplacementFor_;
- Убрано форматирование;
- Функция return возвращает случайное значение;
- Функция printf выводит строки, используя 16-чные значения букв.

Выводы

В ходе лабораторной работы произошло ознакомление с процессом обфускации и основными ее методами. Были рассмотрены несколько свободно распространяемых обфускаторов для ОС Windows, Linux.