Санкт-Петербургский Политехнический университет

Институт Компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Лабораторная работа №3

Дисциплина: Проектирование ОС и компонентов

Тема: Обфускация, обработка кода заданными методами и применение, и сравнение заданных и выбранных обфускаторов

Выполнил студент гр. 13541/4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Шайтан

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. В. Душутина

(подпись)

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Санкт-Петербург

2017

**Выполнение работы:**

Цель работы:

В данной работе необходимо выполнить:

• Обфускацию утилиты Windows

• Обфускацию утилиты Linux

Все эти пункты необходимо выполнить по отношению к утилите, которая была написана ранее. Суть этой утилиты заключается в блокировке файлов с заданным расширением. Эти пункты выполняются в ОС двух семейств.

Обфускация, как один из методов защиты программ

В большинстве случаев для обхода защиты, взломщику требуется изучить принцип работы ее кода, и то, как она взаимодействует с самой защищаемой программой, этот процесс изучения называется процессом реверсивной (обратной) инженерии. Этот процесс часто зависит от свойств человеческой психики, поэтому использование этих свойств позволяет снизить эффективность самого процесса реверсивной инженерии.

Обфускация ("obfuscation" - запутывание), это один из методов защиты программного кода, который позволяет усложнить процесс реверсивной инженерии кода защищаемого программного продукта.

Обфускация может применяться не только для защиты ПП, она имеет более широкое применение, например она, может быть использована создателями вирусов, для защиты их творений и т.д.

Суть процесса обфускации заключается в том, чтобы запутать программный код и устранить большинство логических связей в нем, то есть трансформировать его так, чтобы он был очень труден для изучения и модификации посторонними лицами (будь то взломщики, или программисты которые собираются узнать уникальный алгоритм работы защищаемой программы).

Из этого следует, что обфускация одна не предназначена для обеспечения наиболее полной и эффективной защиты программных продуктов, так как она не предоставляет возможности предотвращения нелегального использования программного продукта. Поэтому обфускацию обычно используют вместе с одним из существующих методов защиты (шифрование программного кода и т.д.), это позволяет значительно повысить уровень защиты ПП в целом (рисунок 0011, "Т1,Т2,...,Tn" - процессы обфускации).

Процесс обфускации как метод зашиты, можно считать сравнительно новым (первые статьи, посвященные обфускации, как методу защиты кода программных продуктов, появились примерно три-четыре года назад), и перспективным.

Обфускация соответствует принципу экономической целесообразности, так как ее использование не сильно, увеличивает стоимость программного продукта, и позволяет при этом снизить потери от пиратства, и уменьшить возможность плагиата в результате кражи уникального алгоритма работы защищаемого программного продукта.

2. Процесс обфускации

Существуют различные определения процесса обфускации. Рассматривая данный процесс с точки зрения защиты ПП, и трансформации кода программы без возможности в последствии вернуться к его первоначальному виду (трансформация "в одну сторону"), можно дать такое определение: Определение. Пусть "TR" будет трансформирующим процессом, тогда при "PR1 =TR=> PR2" программа "PR2" будет представлять собой трансформированный код программы "PR1". Процесс трансформации "TR" будет считаться процессом обфускации если, будут удовлетворены такие требования:

код программы "PR2" в результате трансформации будет существенно отличаться от кода программы "PR1", но при этом он будет выполнять те же функции что и код программы "PR1", а также будет работоспособным.

изучение принципа работы, то есть процесс реверсивной инженерии, программы "PR2" будет более сложным, трудоемким, и будет занимать больше времени, чем программы "PR1".

при каждом процессе трансформации одного и того же кода программы "PR1", код программ "PR2" будут различны.

создание программы детрансформирующей программу "PR2" в ее наиболее похожий первоначальный вид, будет неэффективно.

Так как код, получаемый после осуществления обфускации, над одной и той же программой, разный то процесс обфускации можно использовать для быстрой локализации нарушителей авторских прав (то есть тех покупателей, которые будут заниматься нелегальным распространением купленных копий программ). Для этого определяют контрольную сумму каждой копии программы прошедшей обфускацию, и записывают ее вместе с информацией о покупателе, в соответствующую базу данных. После этого для определения нарушителя, достаточно будет, определив контрольную сумму нелегальной копии программы, сопоставить ее с информацией хранящейся в базе данных.

Программный код может быть представлен в двоичном виде (последовательность байтов представляющих собой так называемый машинный код, который получается после компиляции исходного кода программы) или исходном виде (текст содержащий последовательность инструкций какого-то языка программирования, который понятен человеку, этот текст в последствии будет подвержен компиляции или интерпретации на компьютере пользователя).

Процесс обфускации может быть осуществлен над любым из выше перечисленных видов представления программного кода, поэтому принято выделять следующие уровни процесса обфускации:

низший уровень, когда процесс обфускации осуществляется над ассемблерным кодом программы, или даже непосредственно над двоичным файлом программы хранящим машинный код.

высший уровень, когда процесс обфускации осуществляется над исходным кодом программы написанном на языке высокого уровня.

Осуществление обфускации на низшем уровне считается менее комплексным процессом, но при этом более трудно реализуемым по ряду причин. Одна из этих причин заключается в том, что должны быть учтены особенности работы большинства процессоров, так как способ обфускации, приемлемый на одной архитектуре, может оказаться неприемлемым на другой.

Также на сегодняшний день процесс низкоуровневой обфускации исследован мало (наверно потому что не успел получить широкой популярности).

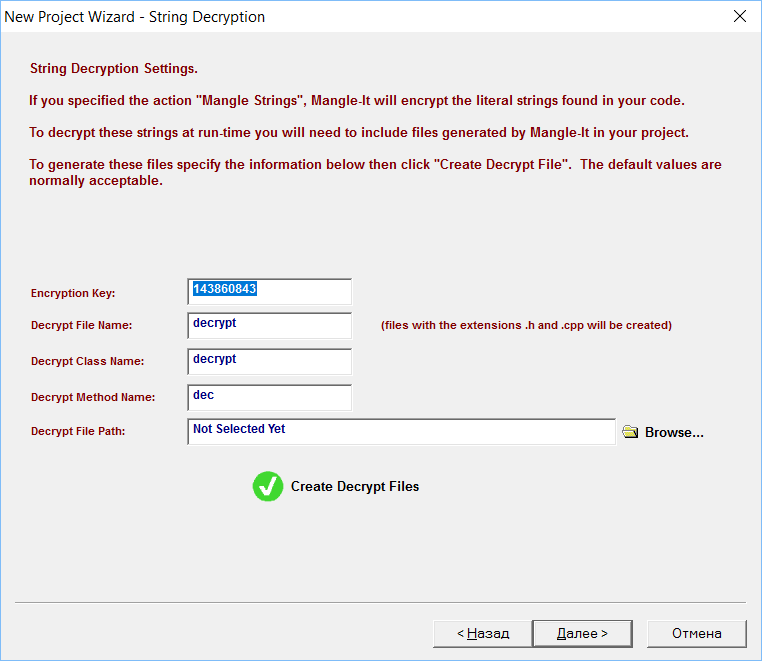
Большинство существующих алгоритмов и методов обфускации (включая те которые будут рассмотрены ниже) могут быть применены для осуществления процесса обфускации как на низшем, так и на высшем уровне.

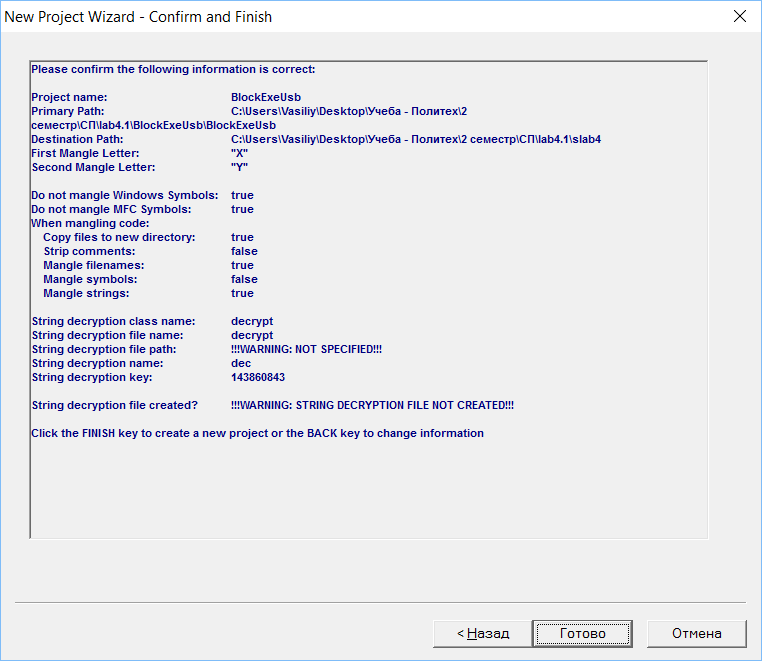
Также иногда может быть неэффективно, подвергать обфускации весь код программы (например, из-за того, что в результате может значительно снизится время выполнения программы), в таких случаях целесообразно осуществлять обфускацию только наиболее важных участков кода.

Далее в тексте употребляются такие понятия: Исходная программа - защищаемая программа, подвергающаяся процессу обфускации. Злоумышленник - личность, занимающаяся изучением алгоритма работы исходной программы (реверсивной инженерией), для каких-либо своих корыстных целей. Объект - имя какого-то хранилища данных, ssнапример, переменной, массива и т.д. Деобфускация - процесс, который позволяет обойти обфускацию, используется злоумышленниками (деобфускация описана в отдельном разделе данного материала).

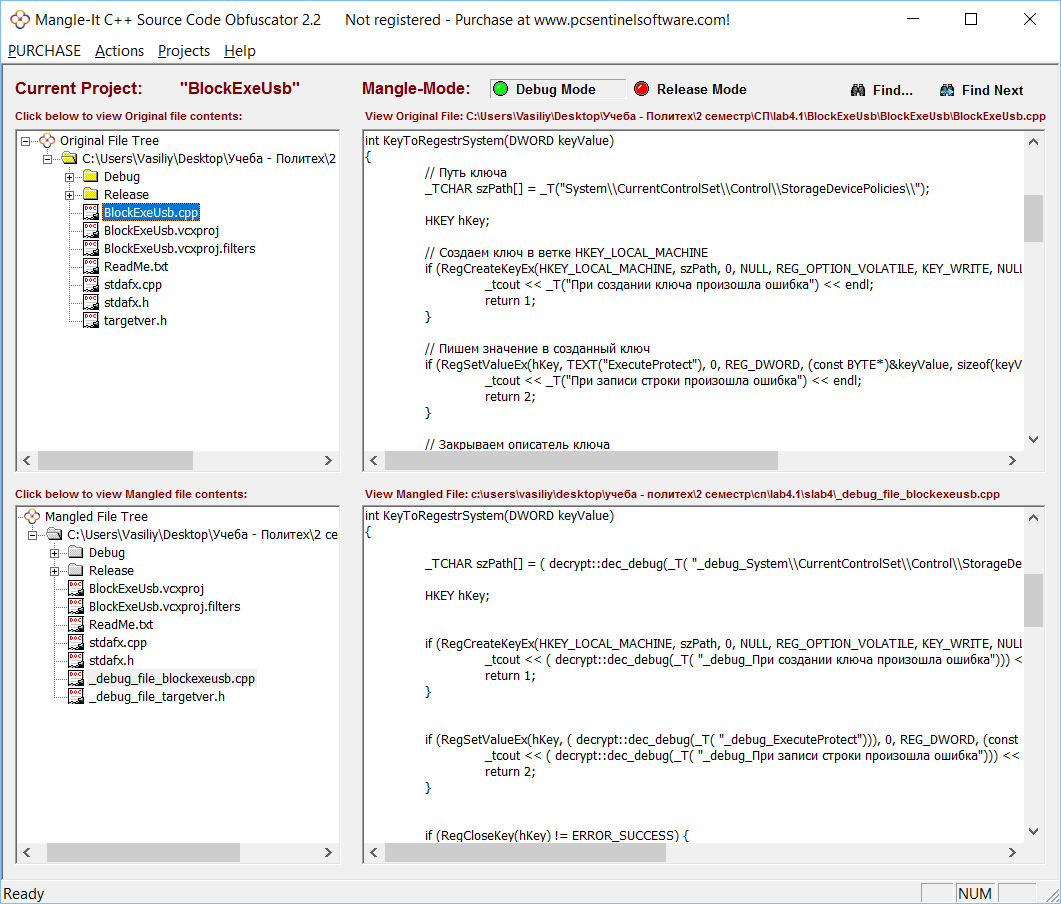
**Windows**

Obfuscator делает C/C++ код нечитаемым. Комментарии удаляются, Символ и имена файлов подогнаны, строки в зашифрованном виде. Работа в Debug и режим Release, операция командной строки позволяет включение в сценариях сборки. Обширный список более 150 000 библиотечных символов, которые исключены из коверкая экономит ваше время. Интерфейс позволяет одновременный просмотр источника и искаженной кода, ваши исходные файлы никогда не изменяются.



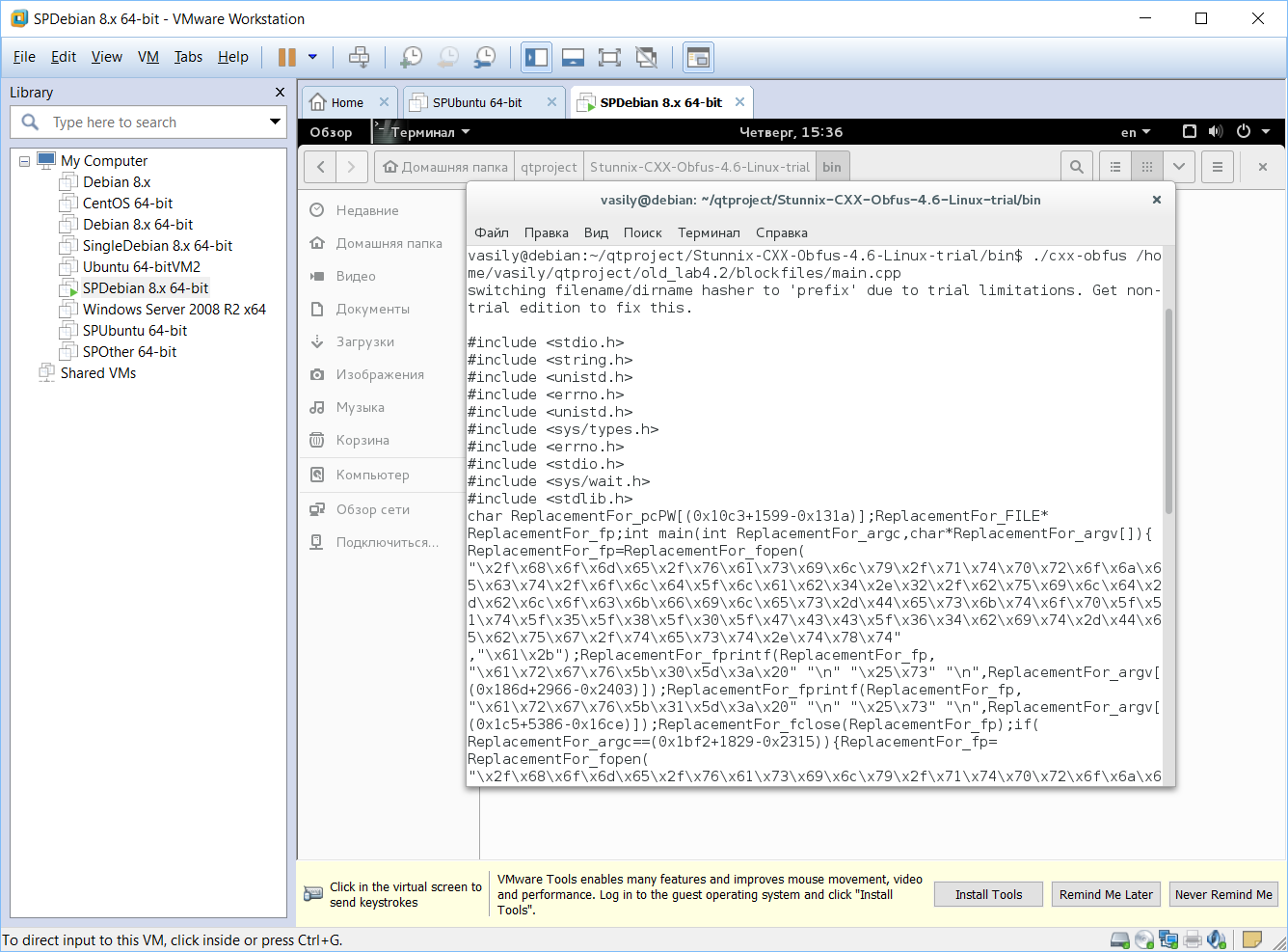


После произведения обфускации, исходный код будет выглядеть следующим образом:

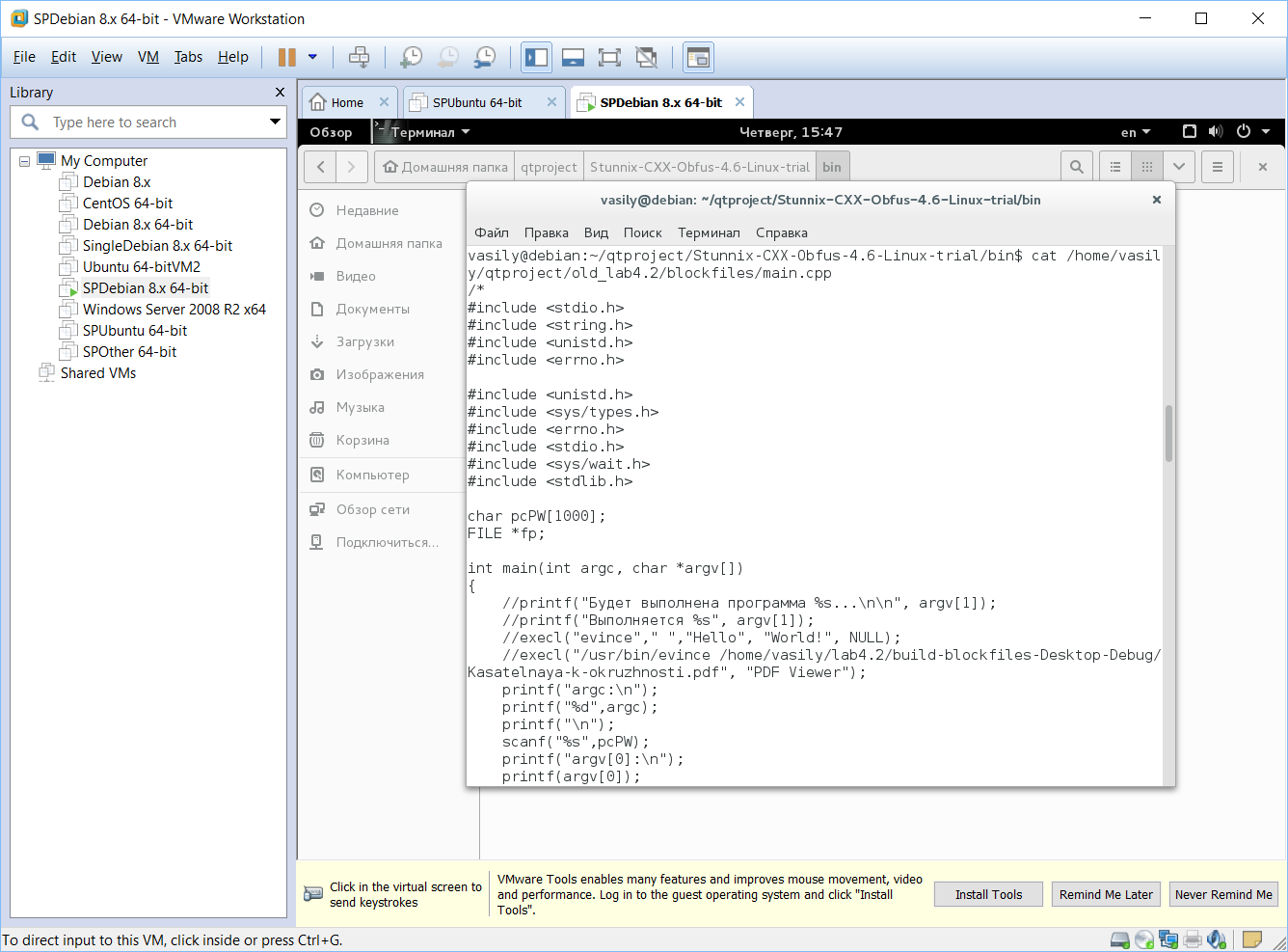


**Linux**

Для обфускации на Linux была использована утилита C/C++ Obfuscator CXX-OBFUS. Сама утилита является платной. Однако, имеется пробная версия с ограниченной функциональностью. Пробная версия может выполнять только один вид преобразования для имен. Заключается этот вид в добавлении строки ReplacementFor\_. В полной версии утилиты, эта строка заменяется, например, на a12b3c4d. Данная утилита не вносит изменений в исходный код обфусцируемой программы. Можно задать соответствующие настройки, чтобы обфусцированный код сохранился в отдельный файл. По умолчанию код выводится в консоль. Ниже показан пример и результат выполнения утилиты.



В качестве сравнения приведем исходный текст файла.



Полный код после обфускации.

|  |
| --- |
| vasily@debian:~/qtproject/Stunnix-CXX-Obfus-4.6-Linux-trial/bin$ ./cxx-obfus /home/vasily/qtproject/old\_lab4.2/blockfiles/main.cpp  switching filename/dirname hasher to 'prefix' due to trial limitations. Get non-trial edition to fix this.  #include <stdio.h>  #include <string.h>  #include <unistd.h>  #include <errno.h>  #include <unistd.h>  #include <sys/types.h>  #include <errno.h>  #include <stdio.h>  #include <sys/wait.h>  #include <stdlib.h>  char ReplacementFor\_pcPW[(0x10c3+1599-0x131a)];ReplacementFor\_FILE\*  ReplacementFor\_fp;int main(int ReplacementFor\_argc,char\*ReplacementFor\_argv[]){  ReplacementFor\_fp=ReplacementFor\_fopen(  "\x2f\x68\x6f\x6d\x65\x2f\x76\x61\x73\x69\x6c\x79\x2f\x71\x74\x70\x72\x6f\x6a\x65\x63\x74\x2f\x6f\x6c\x64\x5f\x6c\x61\x62\x34\x2e\x32\x2f\x62\x75\x69\x6c\x64\x2d\x62\x6c\x6f\x63\x6b\x66\x69\x6c\x65\x73\x2d\x44\x65\x73\x6b\x74\x6f\x70\x5f\x51\x74\x5f\x35\x5f\x38\x5f\x30\x5f\x47\x43\x43\x5f\x36\x34\x62\x69\x74\x2d\x44\x65\x62\x75\x67\x2f\x74\x65\x73\x74\x2e\x74\x78\x74"  ,"\x61\x2b");ReplacementFor\_fprintf(ReplacementFor\_fp,  "\x61\x72\x67\x76\x5b\x30\x5d\x3a\x20" "\n" "\x25\x73" "\n",ReplacementFor\_argv[  (0x186d+2966-0x2403)]);ReplacementFor\_fprintf(ReplacementFor\_fp,  "\x61\x72\x67\x76\x5b\x31\x5d\x3a\x20" "\n" "\x25\x73" "\n",ReplacementFor\_argv[  (0x1c5+5386-0x16ce)]);ReplacementFor\_fclose(ReplacementFor\_fp);if(  ReplacementFor\_argc==(0x1bf2+1829-0x2315)){ReplacementFor\_fp=  ReplacementFor\_fopen(  "\x2f\x68\x6f\x6d\x65\x2f\x76\x61\x73\x69\x6c\x79\x2f\x71\x74\x70\x72\x6f\x6a\x65\x63\x74\x2f\x6f\x6c\x64\x5f\x6c\x61\x62\x34\x2e\x32\x2f\x62\x75\x69\x6c\x64\x2d\x62\x6c\x6f\x63\x6b\x66\x69\x6c\x65\x73\x2d\x44\x65\x73\x6b\x74\x6f\x70\x5f\x51\x74\x5f\x35\x5f\x38\x5f\x30\x5f\x47\x43\x43\x5f\x36\x34\x62\x69\x74\x2d\x44\x65\x62\x75\x67\x2f\x74\x65\x73\x74\x2e\x74\x78\x74"  ,"\x61\x2b");ReplacementFor\_fprintf(ReplacementFor\_fp,  "\x61\x72\x67\x63\x3a\x20" "\n" "\x25\x64" "\n",ReplacementFor\_argc);  ReplacementFor\_fclose(ReplacementFor\_fp);if(ReplacementFor\_strlen(  ReplacementFor\_argv[(0xe42+5092-0x2225)])>=(0x10ca+2279-0x19aa)&&  ReplacementFor\_strncmp(ReplacementFor\_argv[(0xd61+4653-0x1f8d)],  "\x2f\x6d\x65\x64\x69\x61\x2f",(0x29+1117-0x47f))){ReplacementFor\_fp=  ReplacementFor\_fopen(  "\x2f\x68\x6f\x6d\x65\x2f\x76\x61\x73\x69\x6c\x79\x2f\x71\x74\x70\x72\x6f\x6a\x65\x63\x74\x2f\x6f\x6c\x64\x5f\x6c\x61\x62\x34\x2e\x32\x2f\x62\x75\x69\x6c\x64\x2d\x62\x6c\x6f\x63\x6b\x66\x69\x6c\x65\x73\x2d\x44\x65\x73\x6b\x74\x6f\x70\x5f\x51\x74\x5f\x35\x5f\x38\x5f\x30\x5f\x47\x43\x43\x5f\x36\x34\x62\x69\x74\x2d\x44\x65\x62\x75\x67\x2f\x74\x65\x73\x74\x2e\x74\x78\x74"  ,"\x61\x2b");ReplacementFor\_fprintf(ReplacementFor\_fp,  "\x69\x6e\x3a\x20" "\n" "\x25\x64" "\n",(0x821+3679-0x167d));  ReplacementFor\_fclose(ReplacementFor\_fp);ReplacementFor\_system(  "\x78\x74\x65\x72\x6d");ReplacementFor\_printf(  "\x45\x6e\x74\x65\x72\x20\x70\x61\x73\x73\x77\x6f\x72\x64\x3a" "\n");  ReplacementFor\_fgets(ReplacementFor\_pcPW,(0xb08+130-0x7a2),ReplacementFor\_stdin)  ;ReplacementFor\_printf(  "\x73\x74\x72\x63\x6d\x70\x28\x70\x63\x50\x57\x2c" "\"" "\x32\x32\x32" "\\" "\x6e" "\"" "\x29\x3a" "\n"  );ReplacementFor\_printf("\x25\x64",ReplacementFor\_strcmp(ReplacementFor\_pcPW,  "\x32\x32\x32" "\n"));ReplacementFor\_printf("\n");if(ReplacementFor\_strcmp(  ReplacementFor\_pcPW,"\x32\x32\x32" "\n")==(0x1c1+8727-0x23d8)){  ReplacementFor\_printf("\x47\x6f\x6f\x64\x21" "\n");ReplacementFor\_printf(  "\x25\x64",ReplacementFor\_execl(  "\x2f\x75\x73\x72\x2f\x62\x69\x6e\x2f\x65\x76\x69\x6e\x63\x65",  "\x50\x44\x46\x20\x56\x69\x65\x77\x65\x72",ReplacementFor\_argv[  (0x682+3355-0x139c)],(char\*)ReplacementFor\_NULL));};}else ReplacementFor\_printf(  "\x25\x64",ReplacementFor\_execl(  "\x2f\x75\x73\x72\x2f\x62\x69\x6e\x2f\x65\x76\x69\x6e\x63\x65",  "\x50\x44\x46\x20\x56\x69\x65\x77\x65\x72",ReplacementFor\_argv[  (0x1251+4455-0x23b7)],(char\*)ReplacementFor\_NULL));}else ReplacementFor\_printf(  "\x45\x72\x72\x6f\x72\x21" "\n");return(0x140f+1719-0x1ac6);} |

**Вывод**

В ходе данной лабораторной работы были изучены методы обфускации программ на UNIX подобных системах и системах Windows.

**Список используемых источников:**

1. CXX-OBFUS <http://stunnix.com/prod/cxxo/>