Санкт-Петербургский Государственный Политехнический Университет

Институт информационных технологий и управления

Кафедра Компьютерных Систем и Программных Технологий

Отчет по лабораторной работе № 3

По дисциплине «Проектирование ОС и компонентов»

По теме «Обфускация кода»

**Работу выполнила студентка группы № 13541/3**

Шаляпин Н.С. \_\_\_\_\_\_\_\_\_\_\_

**Работу принял преподаватель:**

Душутина Е.В. \_\_\_\_\_\_\_\_\_\_\_

.

Санкт-Петербург

2017

## Программа работы

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

Используются в качестве приложений: утилита, выводящая информацию о файле для для Windows и для Linux. Утилиты были разработаны в прошлом семестре.

Лабораторная работа выполнялась в ОС: Windows 7 и Ubuntu 14.04.

Windows

|  |
| --- |
| Имя узла: ADMIN-ПК  Название ОС: Microsoft Windows 7 Максимальная  Версия ОС: 6.1.7601 Service Pack 1 сборка 7601  Изготовитель ОС: Microsoft Corporation  Параметры ОС: Изолированная рабочая станция  Сборка ОС: Multiprocessor Free  Зарегистрированный владелец: Admin  Зарегистрированная организация:  Код продукта: 00426-OEM-8992662-00173  Дата установки: 02.04.2017, 14:53:54  Время загрузки системы: 27.05.2017, 21:26:27  Изготовитель системы: ASUSTeK Computer Inc.  Модель системы: K53SM  Тип системы: x64-based PC  Процессор(ы): Число процессоров - 1.  [01]: Intel64 Family 6 Model 42 Stepping 7 GenuineIntel ~2501 МГц  Версия BIOS: American Megatrends Inc. K53SM.214, 15.08.2012 |

Linux

|  |
| --- |
| nikita@nikita-K53SM:~$ uname -a  Linux nikita-K53SM 3.19.0-80-generic #88~14.04.1-Ubuntu SMP Fri Jan 13 14:54:07 UTC 2017 x86\_64 x86\_64 x86\_64 GNU/Linux |

**Теоретический обзор**

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.

Цели обфускации:

* Затруднение декомпиляции/отладки и изучения программ с целью обнаружения функциональности.
* Затруднение декомпиляции проприетарных программ с целью предотвращения обратной разработки или обхода DRM и систем проверки лицензий.
* Оптимизация программы с целью уменьшения размера работающего кода и (если используется некомпилируемый язык) ускорения работы.
* Демонстрация неочевидных возможностей языка и квалификации программиста (если производится вручную, а не инструментальными средствами).

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.

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 инструкции. 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.

* 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 её графа потока управления.

**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. Данное преобразование является обратным 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 его выполнение 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 две или более функций объединяются в одну функцию. Списки параметров исходных функций объединяются, и к ним добавляется ещё один 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.

**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.

1. **Mangle-It C++**

Обфускатор Mangle-It C++ – обфускатор программ на языке С++ для Windows. Данное ПО представляет пользователю удобный графический интерфейс, является коммерческим продуктом, но есть пробная версия. Создадим новый проект «FullInfoFile» (Projects->Create New Project Wizard), укажем в нем путь к директории, содержащей исходный коды, и путь к директории для измененных файлов. Были изменены настройки обфускации - добавлен выбор изменения строк и символов (рис. 1).

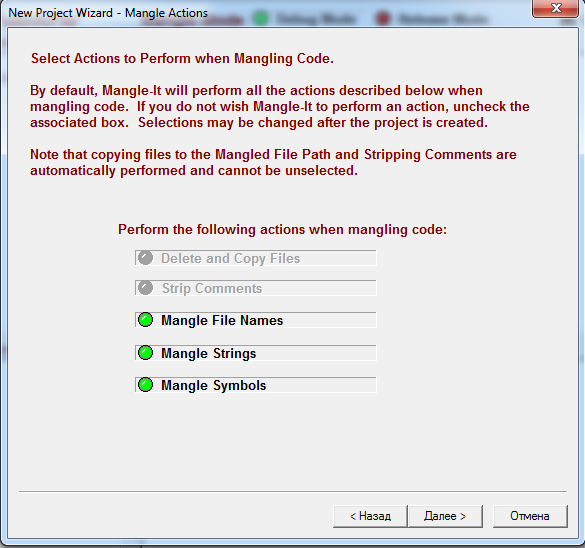


Рис. 1. Настройка обфускации

Создаем файл для шифрования:

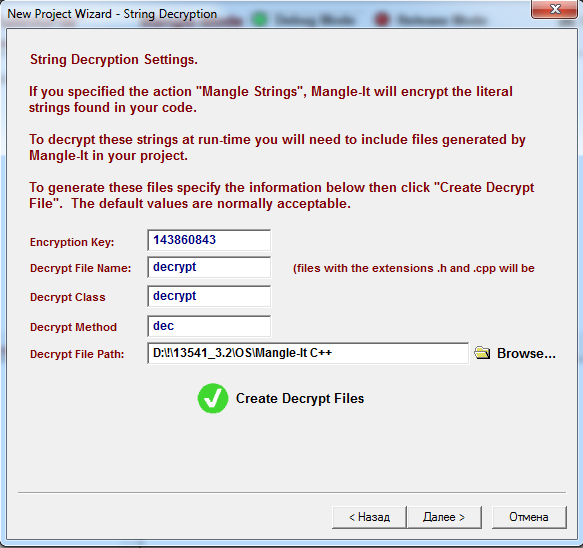


Рис. 2. Создание файла для шифрования строк

После завершения настройка нажимаем кнопку «Готово»:

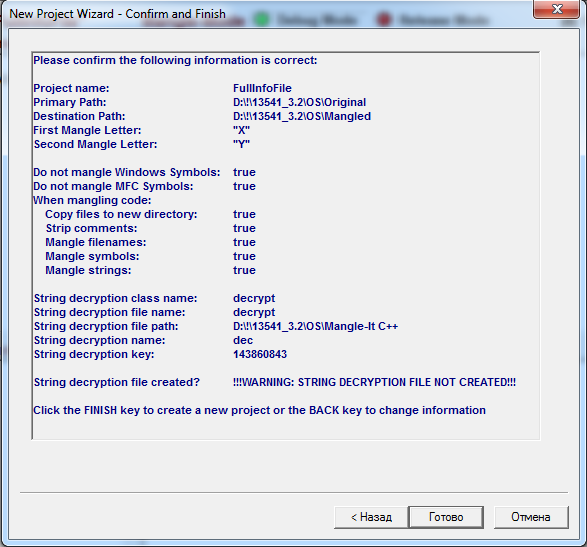


Рис. 3. Настройка завершена

После успешной обфускации (Actions -> Mangle project «FullInfoFile» files), просмотрим коды полученных файлов и сравним их с исходным кодом (рис. 4).

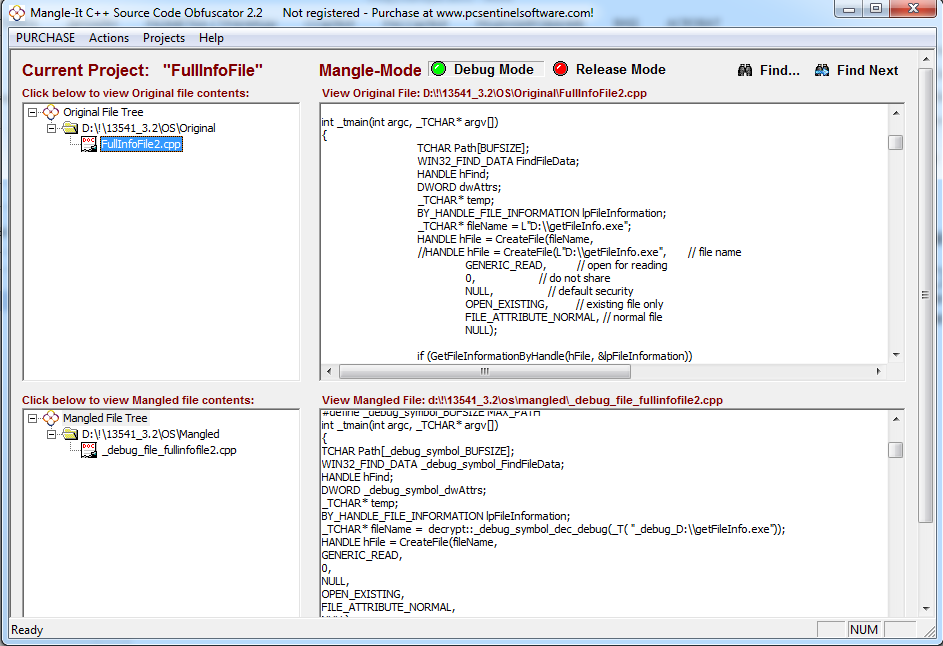


Рис. 4. Сравнение файлов до обфускации и после

При обфускации были удалены комментарии. В начало программы добавлен новый комментарий:

|  |
| --- |
| /\*Mangle-It C++ Source Code Obfuscator\*/ |

Также все переменные, типы, вызовы функций были дополнены вставкой в начало имени строки \_debug\_symbol\_. Данные преобразования делают код программы менее читаемым, но при желании программу легко привести к исходному виду даже вручную.

1. **cxx-obfus**

Обфускатор Stunnix С/С++ obfuscator является кросс-платформенным, поддерживаемые ОС: Windows, Linux, Mac. Данная программа является платной, но у нее есть бесплатная пробная версия. Пробная версия программы значительно ограничена в функциональности. Приведем примеры, созданные разработчиками.

Оригинальный код:

static  int  error\_level  =  4;  /\*  error  level  to  show  alerts  for,  initialized  by  some  accessor\*/  
void  log\_error(const  char\*  whereStr,const  char\*  msg)  
{  
        log\_common(3,whereStr,msg);  //call  common  function  
}  
void  log\_common(int  lvl,const  char\*  whereStr,const  char\*  msg)  
{  
        char  buf[2000];  
        time\_t  currtime  =  time(NULL);  
        snprintf(buf,sizeof(buf),  "%s:  level  %d  at"    
                "  %s:  %s",ctime(&currtime),lvl,whereStr,msg);  
        if  (lvl  >=  error\_level)  
                tracelog.write(buf);  
        fulllog.write(buf);  
}

Пример 1. Пробелы не удаляются, нумерация строк сохранена, комментарии удалены. К каждой из функций и переменных добавлен префикс **ReplacementFor\_**.

static  int  ReplacementFor\_error\_level  =  4;      
void  ReplacementFor\_log\_error(const  char\*  ReplacementFor\_whereStr,const  char\*  ReplacementFor\_msg)  
{  
        ReplacementFor\_log\_common(3,ReplacementFor\_whereStr,ReplacementFor\_msg);    
}  
void  ReplacementFor\_log\_common(int  ReplacementFor\_lvl,const  char\*  ReplacementFor\_whereStr,const  char\*  ReplacementFor\_msg)  
{  
        char  ReplacementFor\_buf[2000];  
        ReplacementFor\_time\_t  ReplacementFor\_currtime  =  ReplacementFor\_time(ReplacementFor\_NULL);  
        ReplacementFor\_snprintf(ReplacementFor\_buf,sizeof(ReplacementFor\_buf),  "%s:  level  %d  at"    
                "  %s:  %s",ReplacementFor\_ctime(&ReplacementFor\_currtime),ReplacementFor\_lvl,ReplacementFor\_whereStr,ReplacementFor\_msg);  
        if  (ReplacementFor\_lvl  >=  ReplacementFor\_error\_level)  
                ReplacementFor\_tracelog.ReplacementFor\_write(ReplacementFor\_buf);  
        ReplacementFor\_fulllog.ReplacementFor\_write(ReplacementFor\_buf);  
}

Пример 2. Отличие от первого примера – отсутствие форматирования.

static  int  ReplacementFor\_error\_level=4;void  ReplacementFor\_log\_error(const  char  
\*ReplacementFor\_whereStr,const  char\*ReplacementFor\_msg){  
ReplacementFor\_log\_common(3,ReplacementFor\_whereStr,ReplacementFor\_msg);}void    
ReplacementFor\_log\_common(int  ReplacementFor\_lvl,const  char\*  
ReplacementFor\_whereStr,const  char\*ReplacementFor\_msg){char  ReplacementFor\_buf[  
2000];ReplacementFor\_time\_t  ReplacementFor\_currtime=ReplacementFor\_time(  
ReplacementFor\_NULL);ReplacementFor\_snprintf(ReplacementFor\_buf,sizeof(  
ReplacementFor\_buf),"%s:  level  %d  at""  %s:  %s",ReplacementFor\_ctime(&  
ReplacementFor\_currtime),ReplacementFor\_lvl,ReplacementFor\_whereStr,  
ReplacementFor\_msg);if(ReplacementFor\_lvl>=ReplacementFor\_error\_level)  
ReplacementFor\_tracelog.ReplacementFor\_write(ReplacementFor\_buf);  
ReplacementFor\_fulllog.ReplacementFor\_write(ReplacementFor\_buf);}

Пример 3. Имена переменных не изменены, комментарии удалены, форматирование отсутствует, символьные и числовые константы обфусцированы.

static  int  error\_level=(0x2235+176-0x22e1);void  log\_error(const  char\*whereStr,  
const  char\*msg){log\_common((0xb68+1365-0x10ba),whereStr,msg);}void  log\_common(  
int  lvl,const  char\*whereStr,const  char\*msg){char  buf[(0x2339+2481-0x251a)];  
time\_t  currtime=time(NULL);snprintf(buf,sizeof(buf),  
"\x25\x73\x3a\x20\x6c\x65\x76\x65\x6c\x20\x25\x64\x20\x61\x74"  
"\x20\x25\x73\x3a\x20\x25\x73",ctime(&currtime),lvl,whereStr,msg);if(lvl>=  
error\_level)tracelog.write(buf);fulllog.write(buf);}

Пример 4. Удалены комментарии.

static  int  error\_level  =  4;    
void  log\_error(const  char\*  whereStr,const  char\*  msg)  
{  
        log\_common(3,whereStr,msg);  }  
void  log\_common(int  lvl,const  char\*  whereStr,const  char\*  msg)  
{  
        char  buf[2000];  
        time\_t  currtime  =  time(NULL);  
        snprintf(buf,sizeof(buf),  "%s:  level  %d  at"    
                "  %s:  %s",ctime(&currtime),lvl,whereStr,msg);  
        if  (lvl  >=  error\_level)  
                tracelog.write(buf);  
        fulllog.write(buf);  
}

Пример 5. Все символы и переменные заменены их md5-хэшами. Комментарии, пробелы удаляются, строки и целые числа искажаются тоже.

static  int  za6a4d01e87=(0x1506+871-0x1869);void  z9fe5637c80(const  char\*  
zea4f2a49a5,const  char\*z4796679a82){z2ddfb2b116((0x570+7693-0x237a),zea4f2a49a5,  
z4796679a82);}void  z2ddfb2b116(int  z072f095beb,const  char\*zea4f2a49a5,const  char  
\*z4796679a82){char  z0aee77a0de[(0x108a+3588-0x16be)];zb125c3488c  zf020607702=  
z8e4d239361(z3f6bbee370);zffa19ce93a(z0aee77a0de,sizeof(z0aee77a0de),  
"\x25\x73\x3a\x20\x6c\x65\x76\x65\x6c\x20\x25\x64\x20\x61\x74"  
"\x20\x25\x73\x3a\x20\x25\x73",z716ec554aa(&zf020607702),z072f095beb,zea4f2a49a5  
,z4796679a82);if(z072f095beb>=za6a4d01e87)zb5837460d1.z8ff0c54161(z0aee77a0de);  
zcfb1131784.z8ff0c54161(z0aee77a0de);}

Пример 6. Все лексемы заменяются на IIlIllI-подобные. Комментарии, пробелы удаляются, строки и целые числа искажаются тоже.

static  int  IllII=(0x571+109-0x5da);void  llIII(const  char\*IlIlI,const  char\*lIIlI)  
{lIlII((0xbe4+4407-0x1d18),IlIlI,lIIlI);}void  lIlII(int  IIIlI,const  char\*IlIlI,  
const  char\*lIIlI){char  llIlI[(0xc49+4026-0x1433)];IIlll  IIlII=Illll(lIlIl);lIlll  
(llIlI,sizeof(llIlI),  
"\x25\x73\x3a\x20\x6c\x65\x76\x65\x6c\x20\x25\x64\x20\x61\x74"  
"\x20\x25\x73\x3a\x20\x25\x73",IlIII(&IIlII),IIIlI,IlIlI,lIIlI);if(IIIlI>=IllII)  
lllll.lllII(llIlI);llIll.lllII(llIlI);}

Пример 7. Обфускация проводится с минимизацией длины используемых лексем. Комментарии, пробелы удаляются, строки и целые числа остаются в прежнем виде.

static  int  a=4;void  e(const  char\*U,const  char\*M){b(3,U,M);}void  b(int  N,const    
char\*U,const  char\*M){char  D[2000];E  d=h(o);n(D,sizeof(D),"%s:  level  %d  at"  
"  %s:  %s",k(&d),N,U,M);if(N>=a)g.V(D);w.V(D);}

Пример 8. Обфускация с использованием рандомизирования лексем. Комментарии, пробелы удаляются, строки и целые числа искажаются тоже.

static  int  \_Dnm7=(0xb07+6813-0x25a0);void  vOLTY(const  char\*FQFFs,const  char\*  
cUb\_e){ey7uq((0x1a62+2267-0x233a),FQFFs,cUb\_e);}void  ey7uq(int  OaHpI,const  char\*  
FQFFs,const  char\*cUb\_e){char  ZMusA[(0x9a0+4497-0x1361)];s1PpD  uxqVt=GQYDB(EiAvD)  
;tABx3(ZMusA,sizeof(ZMusA),  
"\x25\x73\x3a\x20\x6c\x65\x76\x65\x6c\x20\x25\x64\x20\x61\x74"  
"\x20\x25\x73\x3a\x20\x25\x73",YRxMw(&uxqVt),OaHpI,FQFFs,cUb\_e);if(OaHpI>=\_Dnm7)  
HK9Hr.K1RLX(ZMusA);wmf6D.K1RLX(ZMusA);}

* 1. **Тестирование на ОС Windows.**

Создадим новый проект «FullInfoFile», укажем путь к входной и выходной директории (рис. 6).

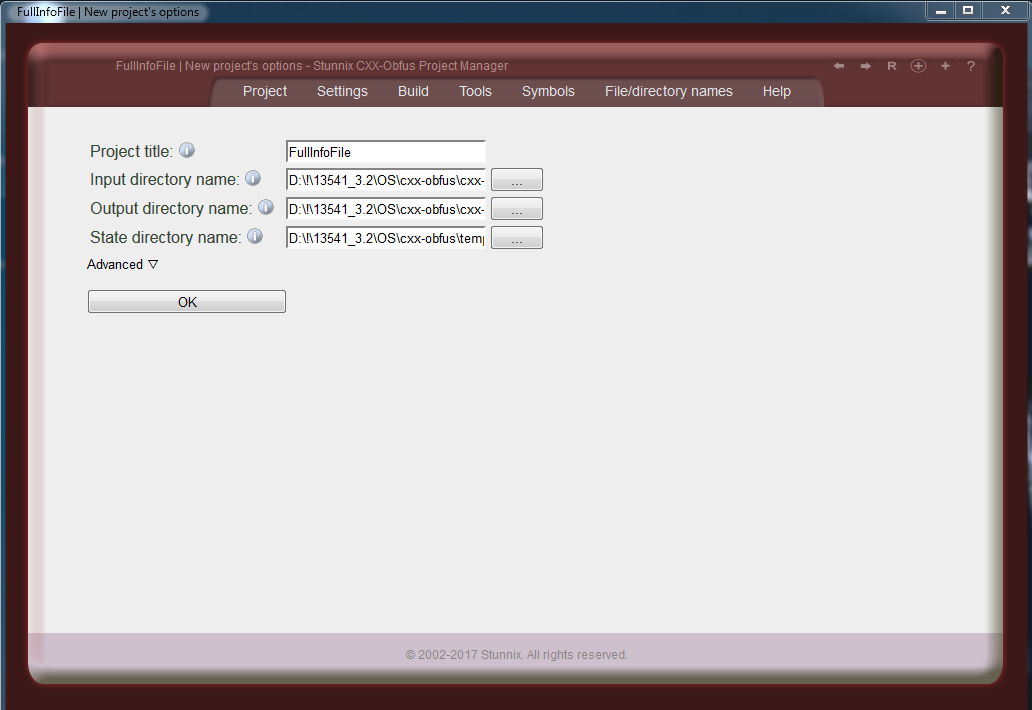


Рис. 6. Создание проекта

Можно поменять режим работы (Settings->Edit modes).

Открываем проект (Project->Open), после этого запускаем обфускацию через меню программы (Build->Rebuild All). Просмотрим обфусцированный код программы:

|  |
| --- |
| …  ReplacementFor\_DWORD ReplacementFor\_dwRet;ReplacementFor\_dwRet=  ReplacementFor\_GetFinalPathNameByHandle(ReplacementFor\_hFile,ReplacementFor\_Path  ,ReplacementFor\_BUFSIZE,ReplacementFor\_VOLUME\_NAME\_DOS);if(ReplacementFor\_dwRet<  ReplacementFor\_BUFSIZE){ReplacementFor\_\_tprintf(ReplacementFor\_TEXT(  "\nThe final path is: %s\n"),ReplacementFor\_Path);}else ReplacementFor\_printf(  "\nThe required buffer size is %d.\n",ReplacementFor\_dwRet);ReplacementFor\_DWORD  ReplacementFor\_retval=0;ReplacementFor\_TCHAR ReplacementFor\_buffer[  ReplacementFor\_BUFSIZE]=ReplacementFor\_TEXT("");ReplacementFor\_TCHAR\*\*  ReplacementFor\_lppPart={ReplacementFor\_NULL};ReplacementFor\_retval=  ReplacementFor\_GetFullPathName(L"D:\\getFileInfo.exe",ReplacementFor\_BUFSIZE,  ReplacementFor\_buffer,ReplacementFor\_lppPart);if(ReplacementFor\_retval==0){  ReplacementFor\_printf("GetFullPathName failed (%d)\n",  ReplacementFor\_GetLastError());return 1;}else{ReplacementFor\_\_tprintf(  ReplacementFor\_TEXT("The full path name is: %s\n"),ReplacementFor\_buffer);if(  ReplacementFor\_lppPart!=ReplacementFor\_NULL&&\*ReplacementFor\_lppPart!=0){  ReplacementFor\_\_tprintf(ReplacementFor\_TEXT(  "The final component in the path name is: %s\n"),\*ReplacementFor\_lppPart);}}  ReplacementFor\_CloseHandle(ReplacementFor\_hFile);ReplacementFor\_hFind=  ReplacementFor\_FindFirstFile(ReplacementFor\_TEXT("D:\\getFileInfo.exe") ,&  ReplacementFor\_FindFileData);if(ReplacementFor\_hFind==  ReplacementFor\_INVALID\_HANDLE\_VALUE){ReplacementFor\_printf(  "FindFirstFile failed (%d)\n",ReplacementFor\_GetLastError());return 1;}else{  ReplacementFor\_dwAttrs=ReplacementFor\_GetFileAttributes(ReplacementFor\_TEXT(  "d:\\getFileInfo.exe"));if(ReplacementFor\_dwAttrs==  ReplacementFor\_INVALID\_FILE\_ATTRIBUTES){ReplacementFor\_printf(  "INVALID\_FILE\_ATTRIBUTES\n");}else{ReplacementFor\_printf("\nFile attributes:");  if(ReplacementFor\_dwAttrs&ReplacementFor\_FILE\_ATTRIBUTE\_ARCHIVE)  ReplacementFor\_printf("\nThe attribute FILE\_ATTRIBUTE\_ARCHIVE is established");  if(ReplacementFor\_dwAttrs&ReplacementFor\_FILE\_ATTRIBUTE\_COMPRESSED)  ReplacementFor\_printf("\nThe attribute FILE\_ATTRIBUTE\_COMPRESSED is established"  );if(ReplacementFor\_dwAttrs&ReplacementFor\_FILE\_ATTRIBUTE\_DEVICE)  ReplacementFor\_printf("\nThe attribute FILE\_ATTRIBUTE\_DEVICE is established");if  (ReplacementFor\_dwAttrs&ReplacementFor\_FILE\_ATTRIBUTE\_DIRECTORY)  ReplacementFor\_printf("\nThe attribute FILE\_ATTRIBUTE\_DIRECTORY is established")  ;if(ReplacementFor\_dwAttrs&ReplacementFor\_FILE\_ATTRIBUTE\_ENCRYPTED)  ReplacementFor\_printf("\nThe attribute FILE\_ATTRIBUTE\_ENCRYPTED is established")  … |

Обфускатор произвел несколько действий:

* Удалил все комментарии
* Изменил форматирование (убрал переносы строк и пробелы, максимальная длина строки была установлена в 300 символов)
* Заменил все строковые константы на строки в шестнадцатеричном формате
* Заменил все целочисленные константы
* Имена переменных и функций были изменены, путем добавления в начало имени строки ReplacementFor\_.

После обфускации код стал не рабочим, пришлось вручную удалять лишние ReplacementFor\_:

|  |
| --- |
| ReplacementFor\_\_\_leave  ReplacementFor\_\_\_finally  ReplacementFor\_strcpy\_s |

Результат работы обфусцированного кода:

|  |
| --- |
| File Index : 295211048576  Number Of Links : 1  File Type: FILE\_TYPE\_DISK  The final path is: \\?\D:\getFileInfo.exe  The full path name is: D:\getFileInfo.exe  File attributes:  The attribute FILE\_ATTRIBUTE\_ARCHIVE is established  File size= 77404656 bytes  File creation time: 02/06/2017 18:23  File last access time: 02/06/2017 18:23  File last write time: 09/14/2016 18:04  FILE\_ENCRYPTABLE  Section: .text  =======================  Virtual size: 0xf244  Raw size: 0xf400  Virtual address: 0x1000  Raw address: 0x400  Characteristics: R X  Section: .itext  =======================  Virtual size: 0xf64  Raw size: 0x1000  Virtual address: 0x11000  Raw address: 0xf800  Characteristics: R X  Section: .data  =======================  Virtual size: 0xc88  Raw size: 0xe00  Virtual address: 0x12000  Raw address: 0x10800  Characteristics: R W  Section: .bss  =======================  Virtual size: 0x56bc  Raw size: 0  Virtual address: 0x13000  Raw address: 0x11600  Characteristics: R W  Section: .idata  =======================  Virtual size: 0xe04  Raw size: 0x1000  Virtual address: 0x19000  Raw address: 0x11600  Characteristics: R W  Section: .tls  =======================  Virtual size: 0x8  Raw size: 0  Virtual address: 0x1a000  Raw address: 0x12600  Characteristics: R W  Section: .rdata  =======================  Virtual size: 0x18  Raw size: 0x200  Virtual address: 0x1b000  Raw address: 0x12600  Characteristics: R  Section: .rsrc  =======================  Virtual size: 0xb200  Raw size: 0xb200  Virtual address: 0x1c000  Raw address: 0x12800  Characteristics: R  SCS\_32BIT\_BINARY  Для продолжения нажмите любую клавишу . . . |

* 1. **Тестирование в ОС Linux**

Обфусцированный код утилиты для Linux, выводящей информацию о файле:

|  |
| --- |
| #include <unistd.h>  #include <stdio.h>  #include <sys/stat.h>  #include <sys/types.h>  #include <time.h>  int main(int ReplacementFor\_argc,char\*\*ReplacementFor\_argv){struct stat fStat;if  (ReplacementFor\_argc!=(0x3+3116-0xc2d)||stat(ReplacementFor\_argv[  (0x6f9+2084-0xf1c)],&fStat)<(0x4fa+1751-0xbd1))return(0x183b+2771-0x230d);  ReplacementFor\_printInfo(fStat);return(0x1b1c+394-0x1ca6);}static char  ReplacementFor\_ftypelet(mode\_t ReplacementFor\_bits){if(S\_ISREG(  ReplacementFor\_bits))return((char)(0xaf8+3281-0x179c));if(S\_ISDIR(  ReplacementFor\_bits))return((char)(0x86+8984-0x233a));if(S\_ISBLK(  ReplacementFor\_bits))return((char)(0x203a+1353-0x2521));if(S\_ISCHR(  ReplacementFor\_bits))return((char)(0x4af+4193-0x14ad));if(S\_ISLNK  (ReplacementFor\_bits))return((char)(0x14f+2449-0xa74));if(S\_ISFIFO(  ReplacementFor\_bits))return((char)(0x39d+7386-0x2007));if(  S\_ISSOCK(ReplacementFor\_bits))return((char)(0x90c+167-0x940));  return((char)(0x1953+867-0x1c77));}void ReplacementFor\_strmode(mode\_t mode,char\*  ReplacementFor\_str){ReplacementFor\_str[(0x692+8060-0x260e)]=  ReplacementFor\_ftypelet(mode);ReplacementFor\_str[(0xbdf+229-0xcc3)]=mode&S\_IRUSR  ?((char)(0x177+2687-0xb84)):((char)(0x727+8068-0x267e));ReplacementFor\_str[  (0x159a+3287-0x226f)]=mode&S\_IWUSR?((char)(0xc2+3070-0xc49)):  ((char)(0x1ff6+1532-0x25c5));ReplacementFor\_str[(0xe65+3348-0x1b76)]=(mode&  S\_ISUID?(mode&S\_IXUSR?((char)(0xc3c+5965-0x2316)):((char)(0x29d+1693-0x8e7))):(  mode&S\_IXUSR?((char)(0xe69+1713-0x14a2)):((char)(0x8a5+3210-0x1502))));  ReplacementFor\_str[(0x317+1198-0x7c1)]=mode&S\_IRGRP?((char)(0x1a28+1130-0x1e20))  :((char)(0x1c44+1478-0x21dd));ReplacementFor\_str[(0x16f6+1739-0x1dbc)]=mode&  S\_IWGRP?((char)(0x3b1+93-0x397)):((char)(0x188+1152-0x5db));ReplacementFor\_str[  (0x3c5+6802-0x1e51)]=(mode&S\_ISGID?(mode&S\_IXGRP?((char)(0xf93+1171-0x13b3)):  ((char)(0xcdd+4877-0x1f97))):(mode&S\_IXGRP?((char)(0x1293+2559-0x1c1a)):  ((char)(0x30b+8230-0x2304))));ReplacementFor\_str[(0x1dbc+567-0x1fec)]=mode&  S\_IROTH?((char)(0x2ac+8847-0x24c9)):((char)(0xd26+3532-0x1ac5));  ReplacementFor\_str[(0x2044+826-0x2376)]=mode&S\_IWOTH?  ((char)(0x10d5+3771-0x1f19)):((char)(0x4f8+1135-0x93a));ReplacementFor\_str[  (0x242+680-0x4e1)]=(mode&S\_ISVTX?(mode&S\_IXOTH?((char)(0x2d8+2886-0xdaa)):  ((char)(0xa3b+5788-0x2083))):(mode&S\_IXOTH?((char)(0x303+67-0x2ce)):  ((char)(0x54b+4144-0x154e))));ReplacementFor\_str[(0xf0c+4991-0x2281)]=  ((char)(0x1133+449-0x12d4));ReplacementFor\_str[(0xab7+6372-0x2390)]='\0';}void  ReplacementFor\_printFileType(struct stat fStat){switch(fStat.st\_mode&S\_IFMT){  case S\_IFBLK:printf("блочное устройство\n");break;case S\_IFCHR:  printf("символьное устройство\n");break;case S\_IFDIR:printf(  "директория\n");break;case S\_IFIFO:printf(  "\x46\x49\x46\x4f\x2f\x70\x69\x70\x65" "\n");break;case S\_IFLNK:printf(  "символьная ссылка\n");break;case S\_IFREG:printf(  "обычный файл\n");break;case S\_IFSOCK:printf(  "\x73\x6f\x63\x6b\x65\x74" "\n");break;default:printf("\x3f" "\n");break;}}void  ReplacementFor\_printInfo(struct stat fStat){printf(  "Размер файла: \n\t%d байт\n",fStat.st\_size);printf(  "Число жёстких ссылок: \n\t%d\n",fStat.st\_nlink);printf(  "inode файла: \n\t%d\n",fStat.st\_ino);printf(  "ID устройства, содержащего файл: \n\t%d\n",fStat.  st\_dev);printf("ID пользователя владельца: \n\t%d\n",fStat.  st\_uid);printf("ID группы владельца: \n\t%d\n",fStat.st\_gid);  printf(  "ID устройства (если специальный файл): \t%d\n",  fStat.st\_rdev);printf("Размер блока для I/O: \n\t%d\n",fStat.  st\_blksize);printf(  "Число выделенных блоков размером 512 байт:\n\t%d\n"  ,fStat.st\_blocks);char ReplacementFor\_perms[(0x22b1+1005-0x2692)];  ReplacementFor\_strmode(fStat.st\_mode,ReplacementFor\_perms);printf(  "Права доступа к файлу:\n\t%s\n",ReplacementFor\_perms);printf(  "Тип файла:\n\t");ReplacementFor\_printFileType(fStat);struct tm\*  ReplacementFor\_timeinfo;printf(  "Последнее изменение статуса:\n\t%s",ctime(&fStat.  st\_ctime));printf("Последний доступ к файлу:\n\t%s",ctime(&  fStat.st\_atime));printf(  "Последния модификация файла:\n\t%s",ctime(&fStat.  st\_mtime));} |

Обфускатор произвел несколько действий:

* Удалил все комментарии
* Изменил форматирование
* Заменил некоторые строковые константы на строки в шестнадцатеричном формате
* Заменил все целочисленные константы
* Имена переменных и функций были изменены, путем добавления в начало имени строки ReplacementFor\_.

Русские символы остались в первоначальном виде. Лог действий:

|  |
| --- |
| **// Запускаем обфускацию**  nikita@nikita-K53SM:~/labs/2semester/lab3$ cxx-obfus/bin/cxx-obfus -x xpg4 infofile-source/main.c -o cxx-obfus-infofile/main.c  switching filename/dirname hasher to 'prefix' due to trial limitations. Get non-trial edition to fix this.  nikita@nikita-K53SM:~/labs/2semester/lab3$ cd cxx-obfus-infofile/  **// Пытаемся скомпилировать обфусцированный код, но возникает ошибка, были вставлены лишние ReplacementFor\_, их пришлось удалить вручную**  nikita@nikita-K53SM:~/labs/2semester/lab3/cxx-obfus-infofile$ gcc -o infofile  ...  /tmp/ccXxmdkn.o: In function `ReplacementFor\_ftypelet':  main.c:(.text+0x11a): undefined reference to `ReplacementFor\_S\_ISLNK'  main.c:(.text+0x14a): undefined reference to `ReplacementFor\_S\_ISSOCK'  collect2: error: ld returned 1 exit status  **// Компилируем утилиту после ручного вмешательства**  nikita@nikita-K53SM:~/labs/2semester/lab3/cxx-obfus-infofile$ gcc -o infofile main.c  …  **// Утилита работает корректно**  nikita@nikita-K53SM:~/labs/2semester/lab3/cxx-obfus-infofile$ ./infofile main.c  Размер файла:  4369 байт  Число жёстких ссылок:  1  inode файла:  808327  ID устройства, содержащего файл:  2055  ID пользователя владельца:  1000  ID группы владельца:  1000  ID устройства (если специальный файл): 0  Размер блока для I/O:  4096  Число выделенных блоков размером 512 байт:  16  Права доступа к файлу:  -rw-rw-r--  Тип файла:  обычный файл  Последнее изменение статуса:  Mon May 8 16:01:54 2017  Последний доступ к файлу:  Mon May 8 16:02:52 2017  Последния модификация файла:  Mon May 8 16:01:54 2017  nikita@nikita-K53SM:~/labs/2semester/lab3/cxx-obfus-infofile$ |

Код после обфускации оказался нерабочим, после удаления лишних ReplacementFor\_ код заработал.

|  |
| --- |
| ReplacementFor\_S\_ISLNK  ReplacementFor\_S\_ISSOCK |

1. **ASPack**

ASPack - программа для сжатия исполняемых файлов под Win32 и защиты от непрофессионального реверс-инжиниринга. ASPack уменьшает размер файлов и библиотек под Windows XP/Vista/7/8/10 и Windows Server 2003/2008/2012 до 70% (степень сжатия выше стандарта ZIP на 10-20%), а также сокращает время загрузки таких приложений в локальных сетях и Интернет. ASPack также защищает программы от непрофессионального анализа. Программы, сжатые упаковщиком ASPack, являются автономными и запускаются так же, как и до сжатия, без потери времени и ухудшения производительности.

## Возможности упаковщика ASPack

* улучшенная обработка исполняемых файлов (EXE, DLL, OCX)
* шифрование и сжатие exe файлов, программного кода, данных и ресурсов
* полностью прозрачная, автономная работа с поддержкой длинных имен файлов
* быстрая процедура распаковки обеспечивает большую производительность в сравнении с продуктами конкурентов
* непосредственная интеграция в оболочку Windows, что упрощает схему работы
* полная совместимость с Windows XP/Vista/7/8/10 и Windows Server 2003/2008/2012

## Преимущества упаковщика ASPack

* значительное сжатие размеров exe файлов, в среднем на 40-70%
* сокращение времени загрузки приложений в локальных сетях и Интернет
* встроенные Windows-приложения требуют значительно меньше места
* защита ресурсов и кода от дизассемблеров, декомпиляторов и считывания содержимого памяти
* за дистрибуцию сжатых программ роялти не взимается
* совместимость с исполняемыми файлами, сгенерированными Microsoft Visual C++, Visual Basic, Inprise (Borland) Delphi and C++ Builder и другими Win32-компиляторами

ASPack является коммерчески продуктом. Персональная лицензия для некоммерческих приложений (2 ПК) - €34. Лицензия для частного разработчика (3 ПК) - €57. Лицензия для компании (30 ПК, поддержка запуска ASPack на виртуальных машинах) - €208.

1. **ASProtect 64**

Продукт предназначен для профессиональной защиты от несанкционированного использования, промышленного и домашнего копирования, профессионального взлома и анализа программных продуктов, распространяемых через Интернет и на любых физических носителях. Отличительной особенностью нового продукта является поддержка защиты только 64-битных приложений и .NET приложений для Windows. Кроме того, продукт построен на совершенно новом ядре защиты, ранее не использовавшимся в продуктах ASPack Software.

## Возможности продукта ASProtect 64

* Защита приложения, скомпилированного в PE-файл, содержащий исполняемый машинный код (native-код) процессора x64 (расширение таких файлов – exe, dll)
* Возможна защита приложения, скомпилированного в исполняемый .NET-файл (managed-код)
* Защита кода программы от анализа и взлома
* Шифрование исходного кода программы
* Проверка целостности приложения
* Противодействие отладчикам и дизассемблерам
* Защита от изменений данных в памяти программы
* API для взаимодействия программного обеспечения и процедур защиты
* Создание и верификация регистрационных ключей с использованием алгоритмов шифрования с открытым ключом
* Генерация регистрационных ключей, основанная на конфигурации конкретного компьютера
* Возможность создавать пробные (триальные) версии, которые ограничивают функциональность приложения в соответствии с оставшимся временем использования, количеством запусков и общим временем работы приложения
* Отображения окон-напоминаний об истечении срока действия
* Отдельно поддержана реализация демо-режима через ограничение функционала защищаемого приложения
* Наличие встроенного и модифицируемого графического интерфейса от системы защиты для обработки процесса активации защищённого приложения
* Полная совместимость с Windows XP/Vista/7/8/10 64 bit и Windows Server 2003/2008/2012 64 bit
* Защита от запуска защищенных приложений на виртуальных машинах.

Лицензия для частного разработчика (3 ПК; привязка к ПК обязательна) - €144. Лицензия для компании (30 ПК; с привязкой и без привязки к ПК) - €376. Продление лицензии для частного разработчика, подписка на поддержку и обновления на 1 год - €73. Продление лицензии для компании, подписка на поддержку и обновления на 1 год - €189

1. **Semantic Designs: C++ Source Code Obfuscator**

Табл. 1. Поддерживаемые диалекты C++

|  |  |
| --- | --- |
| Диалект | Опиание |
| ECMA372c2005 | The ECMA 372:2005 standard of C++/CLI. |
| GCC3 | The GCC 3 dialect of C++. |
| GCC4 | The GCC 4 dialect of C++. |
| ISO14882c1998 | The ISO/IEC 14882:1998 standard of C++. |
| ISO14882c2011 | The ISO/IEC 14882:2011 standard of C++. |
| ISO14882c2014 | The ISO/IEC 14882:2014 standard of C++. |
| Union | A combination of all supported dialects of C++. |
| VisualCpp6 | The Microsoft Visual C++ 6 dialect of C++. |
| VisualStudio2005 | The Microsoft Visual Studio 2005 dialect of C++. |
| VisualStudio2010 | The Microsoft Visual Studio 2010 dialect of C++. |
| VisualStudio2013 | The Microsoft Visual Studio 2013 dialect of C++. |

Нет поддержки диалекта Microsoft Visual Studio 2015. Пример представленный на сайте:

|  |
| --- |
| #include <string.h>  #include <stdio.h>  #include "avl\_tree.h"  // An "environment" of variables and (string) values.  class env  {  private:  struct node  {  // Child pointers.  node \*gt, \*lt;  // First character of variable name string is actually balance factor.  // Remaining characters are name as nul-terminated string.  char \*name;  // Value of variable, nul-terminated string.  char \*value;  };  …  e.set("the", "the value");  e.set("lazy", "lazy value");  e.set("dog", "dog value");  e.set("DOG", "DOG value");  e.set("DOG", 0);  printf("The value of \"dog\" is \"%s\"\n\n", e.get("dog"));  printf("DUMP\n");  e.dump();  e.clear();  e.dump();  return(0);  } |

Результат обфускации:

|  |
| --- |
| #include <string.h>  #include <stdio.h>  #include "avl\_tree.h"  class  O110001110110  {  private  :  struct  O110001110111  {  O110001110111  \*  l110001111000  ,  \*  lt  ;  char  \*  name  ;  char  \*  value  ;  }  ;  struct  l110001111001  {  typedef  O110001110111  \*  l110001111010  ;  typedef  const  char  \*  O110001111011  ;  …  O110010010110  .  set  (  "the"  ,  "the value"  )  ;  O110010010110  .  set  (  "lazy"  ,  "lazy value"  )  ;  O110010010110  .  set  (  "dog"  ,  "dog value"  )  ;  O110010010110  .  set  (  "DOG"  ,  "DOG value"  )  ;  O110010010110  .  set  (  "DOG"  ,  0  )  ;  printf  (  "The value of \"dog\" is \"%s\"\n\n"  ,  O110010010110  .  get  (  "dog"  )  )  ;  printf  (  "DUMP\n"  )  ;  O110010010110  .  l110010010001  (  )  ;  O110010010110  .  clear  (  )  ;  O110010010110  .  l110010010001  (  )  ;  return  (  0  )  ;  } |

Обфускатор удаляет комментарии, изменяет форматирование, изменяет название структур, методов. Не изменяет строковые и целочисленные константы.

* 1. **Тестирование в ОС Linux**

Ограничения пробной версии обфускатора:

* Количество строк в файле: 500
* Количество файлов: 10
* Максимальное количество строк: 3000

Для установки был скачен .exe файл, установлен Wine. Wine — свободное программное обеспечение, позволяющее пользователям UNIX-подобных систем архитектуры x86 исполнять 16-, 32- и 64- битные приложения Microsoft Windows. С помощью Wine был запущен установщик обфускатора.

Запускаем обфускатор:

|  |
| --- |
| nikita@nikita-K53SM:~/.wine/drive\_c/Program Files (x86)/SemanticDesigns/DMS/Executables$ **./DMSProjectSpecifier C~GCC4 PrettyPrinter**  DMSProjectSpecifier Launcher for Linux (v. 1.0.15062506)  DMSProjectSpecifier (Lily-Pad) 1.0.14082704  Starting Semantic Designs PrettyPrinter by  /bin/bash -c "java -cp \"/home/nikita/.wine/drive\_c/Program Files (x86)/SemanticDesigns/DMS/Tools/ProjectSpecifier/ProjectSpecifier.jar:/home/nikita/.wine/drive\_c/Program Files (x86)/SemanticDesigns/DMS/..\" DMS.Domains.C.GCC4.Tools.PrettyPrinter.ProjectSpecifier" |

Обфускатор имеет удобный графический интерфейс (рис. 7), позволяющий выбрать файлы для обфускации, директорию для сохранения выходных файлов, файлы, содержащие зарезервированные имена (если не указать их, то обфускатор изменит названия и библиотечных вызовов).

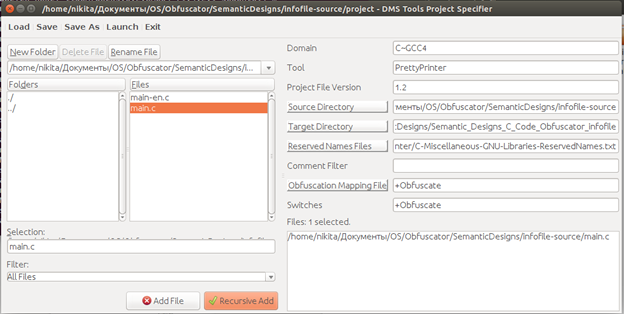


Рис. 7. Создание проекта

Проект был сохранен, после этого была запущена обфускация (Launch, рис. 8). В директории с проектом сохранился лог файл.

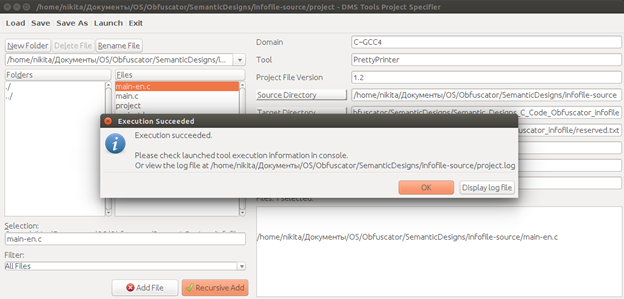


Рис. 8. Сообщение об успешном выполнении

Код после обфускации:

|  |
| --- |
| #include \  <unistd.h>  #include \  <stdio.h>  #include \  <sys/stat.h>  #include \  <sys/types.h>  #include \  <time.h>  int  main  (  int  O  ,  char  \*  \*  o  )  {  struct  stat  i  ;  if  (  O  !=  02  ||  stat  (  o  [  01  ]  ,  &  i  )  <  0  )  return  01  ;  I  (  i  )  ;  return  0  ;  }  static  char  l  (  mode\_t  Oo  )  {  if  (  S\_ISREG  (  Oo  )  )  return  '\055'  ;  …  case  S\_IFDIR  :  printf  (  "\x0434\x0438\x0440\x0435\x043a\x0442\x043e\x0440\x0438\x044f\012"  )  ;  break  ;  case  S\_IFIFO  :  printf  (  "\106\111F\117\057\160ipe\012"  )  ;  break  ;  case  S\_IFLNK  :  printf  (  "\x0441\x0438\x043c\x0432\x043e\x043b\x044c\x043d\x0430\x044f\040\x0441\x0441\x044b\x043b\x043a\x0430\012"  )  …  printf  (  "\111D\040\x0433\x0440\x0443\x043f\x043f\x044b \x0432\x043b\x0430\x0434\x0435\x043b\x044c\x0446\x0430: \012\t\045d\n"  ,  i  .  st\_gid  )  ;  … |

Обфускатор изменил форматирование, название функций и переменных, удалил комментарии, заменил строковые константы. Целочисленные константы остались без изменения.

Проверим работоспособность:

|  |
| --- |
| nikita@nikita-K53SM:~/labs/2semester/lab3/Semantic\_Designs\_C\_Source\_Code\_Obfuscator\_infofile$ gcc -o infofile main.c  nikita@nikita-K53SM:~/labs/2semester/lab3/Semantic\_Designs\_C\_Source\_Code\_Obfuscator\_infofile$ ./infofile main.c  07<5@ D09;0:  4681 109B  '8A;> 6QAB:8E AAK;>::  1  inode D09;0:  791502  ID CAB@>9AB20, A>45@60I53> D09;:  2055  ID ?>;L7>20B5;O 2;045;LF0:  1000  ID 3@C??K 2;045;LF0:  1000  ID CAB@>9AB20 (5A;8 A?5F80;L=K9 D09;): 0  07<5@ 1;>:0 4;O I/O:  4096  '8A;> 2K45;5==KE 1;>:>2 @07<5@>< 512 109B:  16  ­@020 4>ABC?0 : D09;C:  -rw-rw-r--  "8? D09;0:  >1KG=K9 D09;  ­>A;54=55 87<5=5=85 AB0BCA0:  Wed May 10 17:07:09 2017  ­>A;54=89 4>ABC? : D09;C:  Wed May 10 17:07:24 2017  ­>A;54=8O <>48D8:0F8O D09;0:  Wed May 10 17:07:09 2017 |

Программа работает, но русские символы испорчены, нельзя было так заменять строковые константы.

После обфускации файла, в котором нет русских символов в строковых константах, проблем не обнаружено:

|  |
| --- |
| nikita@nikita-K53SM:~/labs/2semester/lab3/Semantic\_Designs\_C\_Source\_Code\_Obfuscator\_infofile$ gcc -o infofile-en main-en.c  nikita@nikita-K53SM:~/labs/2semester/lab3/Semantic\_Designs\_C\_Source\_Code\_Obfuscator\_infofile$ ./infofile-en main.c  File size:  4681 byte  Number of hard links:  1  inode of file:  791502  ID device Aontaining the file:  2055  ID owner user:  1000  ID owner group:  1000  ID of device (if special file): 0  Size of block for I/O:  4096  Count of blocks (size 512 byte):  16  Access to file:  -rw-rw-r--  Type of file:  regular file  Last status change:  Wed May 10 17:20:08 2017  Last access to file:  Wed May 10 17:21:02 2017  Last modification to file:  Wed May 10 17:07:09 2017 |

Данный обфускатор делает код нечитаемым, но тот факт, что нужно указывать зарезервированные имена, огорчает. В директории установки был обнаружен файл с зарезервированными именами, но он не очень полный. Пришлось создать также свой файл и записать туда зарезервированные имена библиотечных вызовов, структур и констант.

**Вывод**

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

Пробные версии Mangle-It C++ и Semantic Designs: C++ Source Code Obfuscator не позволяют настроить обфускацию.

Программа Stunnix C and C++ Obfuscator позволяет настроить режим обфускации. В Stunnix C and C++ Obfuscator есть опции --excludeidentsfile filename (-x filename), которая позволяет указать файлы, в которых находятся зарезервированные имена, каждое на отдельной строке.

Программы ASPack и ASProtect 64 гарантируют защиту ресурсов и кода от дизассемблеров, декомпиляторов и считывания содержимого памяти.

**Список использованных источников**

1. <http://stunnix.com/prod/cxxo/>
2. <http://citforum.ru/security/articles/analysis/>
3. [https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%84%D1%83%D1%81%D0%BA%D0%B0%D1%86%D0%B8%D1%8F\_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5\_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)](https://ru.wikipedia.org/wiki/Обфускация_(программное_обеспечение))
4. <http://stunnix.com/prod/cxxo/newsample.shtml>
5. <http://www.asprotect.ru/aspack.html>
6. <http://www.asprotect.ru/asprotect64.html>
7. <http://www.semdesigns.com/Products/Obfuscators/CppObfuscationExample.html>
8. <https://www.linux.com/learn/how-install-and-use-wine-run-windows-applications-linux>
9. <https://wiki.winehq.org/Ubuntu>