**Тема**: Обфускация кода

**Цель работы**

Изучить методы и средства обфускации кода. Попробовать одно из средств обфускации на практике.

**Выполнение работы**

Обфускация или запутывание кода — приведение исходного текста или исполняемого кода программы к виду, сохраняющему её функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции. 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).

Обфускация кода может проводиться на нескольких уровнях:

* На уровне исходного кода
* На уровне машинного кода
* На уровне промежуточного (байт) кода

**Обфускация на уровне исходного кода**

Для изучения обфускации был выбран обфускатор исходных кодов Stunnix С/С++ obfuscator. Данная утилита является платной, но у нее есть бесплатная триальная версия. Для проверки работоспособности обфускатора была написана небольшая программа на С, которая считывает строку из входного потока и печатает ее в файл:

#include <string.h>

#include <unistd.h>

#include <stdio.h>

#define DEFAULT\_FILE "./myout"

int main(int argc, char\*\* argv) {

char \*portname = DEFAULT\_FILE;

if (argc > 1)

portname = argv[1];

FILE\* fd = fopen (portname, "w");

if (fd < 0) {

//error\_message ("error %d opening %s: %s", errno, portname, strerror (errno));

printf("error opening the device\n");

return 1;

}

char buf [100];

int n = scanf ("%s", buf); // read up to 100 characters if ready to read

printf("Input: %s\n", buf);

fprintf(fd, "%s\n", buf); // send 8 character greeting

return 0;

}

Пример работы:

kivi@kivi-VirtualBox:~/workspace/obfus$ gcc -o test dev\_test.c

kivi@kivi-VirtualBox:~/workspace/obfus$ ./test

myinput

Input: myinput

kivi@kivi-VirtualBox:~/workspace/obfus$ cat myout

myinput

Запустим обфускатор :

kivi@kivi-VirtualBox:~/workspace/obfus$ ~/Downloads/bin/cxx-obfus -x xpg4 dev\_test.cpp -o obfuscated.cpp

switching filename/dirname hasher to 'prefix' due to trial limitations. Get non-trial edition to fix this.

kivi@kivi-VirtualBox:~/workspace/obfus$

Результат обфускации этого кода:

#include <string.h>

#include <unistd.h>

#include <stdio.h>

#define ReplacementFor\_DEFAULT\_FILE "\x2e\x2f\x6d\x79\x6f\x75\x74"

int main(int ReplacementFor\_argc,char\*\*ReplacementFor\_argv){char\*

ReplacementFor\_portname=ReplacementFor\_DEFAULT\_FILE;if(ReplacementFor\_argc>

(0x13e0+3725-0x226c))ReplacementFor\_portname=ReplacementFor\_argv[

(0x1d0+2656-0xc2f)];FILE\*fd=fopen(ReplacementFor\_portname,"\x77");if(fd<

(0x864+5839-0x1f33)){printf(

"\x65\x72\x72\x6f\x72\x20\x6f\x70\x65\x6e\x69\x6e\x67\x20\x74\x68\x65\x20\x64\x65\x76\x69\x63\x65" "\n"

);return(0xea7+3377-0x1bd7);}char buf[(0xb77+3445-0x1888)];int ReplacementFor\_n=

scanf("\x25\x73",buf);printf("\x49\x6e\x70\x75\x74\x3a\x20\x25\x73" "\n",buf);

fprintf(fd,"\x25\x73" "\n",buf);return(0x1949+1543-0x1f50);}

Как мы видим, обфускатор произвел несколько действий:

* Удалил все комментарии
* Изменил форматирование (убрал переносы строк и пробелы)
* Заменил все строковые константы на строки в шестнадцатеричном формате
* Заменил все целочисленные константы
* Заменил все имена переменных

Проверим, что обфусцированный код работает так же:

kivi@kivi-VirtualBox:~/workspace/obfus$ gсс -o test obfuscated.cpp

obfuscated.cpp: In function ‘int main(int, char\*\*)’:

obfuscated.cpp:5:37: warning: deprecated conversion from string constant to ‘char\*’ [-Wwrite-strings]

#define ReplacementFor\_DEFAULT\_FILE "\x2e\x2f\x6d\x79\x6f\x75\x74"

^

obfuscated.cpp:7:25: note: in expansion of macro ‘ReplacementFor\_DEFAULT\_FILE’

ReplacementFor\_portname=ReplacementFor\_DEFAULT\_FILE;if(ReplacementFor\_argc>

^

kivi@kivi-VirtualBox:~/workspace/obfus$ ./test

input

Input: input

kivi@kivi-VirtualBox:~/workspace/obfus$ cat myout

input

kivi@kivi-VirtualBox:~/workspace/obfus$

**Собственный обфускатор**

Напишем собственный обфускатор, который будет:

* Убирать все форматирование (переносы строк, пробелы и табуляции)
* Удалять все комментарии
* Заменять строковые константы на константы в шестнадцатеричном формате

Обфускатор был реализован на языке С++. Исходный код:

#include <iostream>

#include <fstream>

#include <algorithm>

#include <sstream>

void deleteComments(std::string& str) {

size\_t commentStart = 0;

size\_t commentEnd = 0;

while (true) {

commentStart = str.find("/\*", commentStart);

if (commentStart == std::string::npos) break;

commentEnd = str.find("\*/", commentStart + 1);

if (commentEnd == std::string::npos) {

std::cout << "No end of comment, exiting" << std::endl;

exit(-1);

}

str.erase(str.begin() + commentStart, str.begin() + commentEnd + 2);

}

commentStart = 0;

while (true) {

commentStart = str.find("//", commentStart);

if (commentStart == std::string::npos) break;

commentEnd = str.find("\n", commentStart + 1);

if (commentEnd == std::string::npos) {

std::cout << "No end of comment, exiting" << std::endl;

exit(-1);

}

str.erase(str.begin() + commentStart, str.begin() + commentEnd);

}

}

void replaceStrings(std::string& str) {

size\_t start = 0;

size\_t end\_ = 0;

while (true) {

start = str.find("\"", start);

if (start == std::string::npos) break;

end\_ = str.find("\"", start + 1);

if (end\_ == std::string::npos) {

std::cout << "No end of string, exiting" << std::endl;

exit(-1);

}

std::ostringstream ss;

for (auto i = start + 1; i < end\_; ++i) {

if (str[i] == '\\' && str[i + 1] == 'n') {

ss << "\\x" << std::hex << (int)'\n';

++i;

} else

ss << "\\x" << std::hex << (int)str[i];

}

str.erase(str.begin() + start + 1, str.begin() + end\_);

str.insert(start + 1, ss.str());

start = start + 2 + ss.str().size();

}

}

void removeFreeLinesAndSpaces(std::string& str) {

size\_t start = 0;

while (true) {

start = str.find("\n\n", start);

if (start == std::string::npos) break;

str.erase(str.begin() + start, str.begin() + start + 1);

}

start = 0;

while (true) {

start = str.find(" ", start);

if (start == std::string::npos) break;

str.erase(str.begin() + start, str.begin() + start + 1);

}

}

int main(int argc, char\*\* argv) {

if (argc < 2) {

std::cout << "No input file" << std::endl;

return -1;

}

std::ifstream in(argv[1]);

std::string str((std::istreambuf\_iterator<char>(in)),

std::istreambuf\_iterator<char>());

std::cout << str << std::endl << std::endl;

std::string result;

std::string work;

std::vector<size\_t> indexes;

indexes.push\_back(str.rfind("#include"));

indexes.push\_back(str.rfind("#define"));

indexes.push\_back(str.rfind("#pragma"));

indexes.push\_back(str.rfind("#using"));

indexes.erase(std::remove(indexes.begin(), indexes.end(), std::string::npos), indexes.end());

auto max = std::max\_element(indexes.begin(), indexes.end());

if (max == indexes.end()) {

work = str;

} else {

auto endOfProproc = str.find('\n', \*max);

result.append(str.substr(0, endOfProproc + 1));

work = str.substr(endOfProproc + 1, str.size());

}

deleteComments(work);

work.erase(std::remove(work.begin(), work.end(), '\n'), work.end());

work.erase(std::remove(work.begin(), work.end(), '\t'), work.end());

result.append(work);

removeFreeLinesAndSpaces(result);

replaceStrings(result);

std::cout << result << std::endl;

return 0;

}

Результат работы обфускатора:

#include <string.h>

#include <unistd.h>

#include <stdio.h>

#define DEFAULT\_FILE "\x2e\x2f\x6d\x79\x6f\x75\x74"

int main(int argc, char\*\* argv) {char \*portname = DEFAULT\_FILE; if (argc > 1) portname = argv[1];FILE\* fd = fopen (portname, "\x77");if (fd < 0){ printf("\x65\x72\x72\x6f\x72\x20\x6f\x70\x65\x6e\x69\x6e\x67\x20\x74\x68\x65\x20\x64\x65\x76\x69\x63\x65\xa"); return 1;} char buf [100]; int n = scanf ("\x25\x73", buf); printf("\x49\x6e\x70\x75\x74\x3a\x20\x25\x73\xa", buf);fprintf(fd, "\x25\x73\xa", buf); return 0;}

По результату видно, что обфускатор выполняет свои функции. Проверим, что результат выполнения обфусцированного кода такой же:

kivi@kivi-VirtualBox:~/workspace/obfus$ gcc -o test obfuscated\_my.cpp

obfuscated\_my.cpp: In function ‘int main(int, char\*\*)’:

obfuscated\_my.cpp:4:22: warning: deprecated conversion from string constant to ‘char\*’ [-Wwrite-strings]

#define DEFAULT\_FILE "\x2e\x2f\x6d\x79\x6f\x75\x74"

^

obfuscated\_my.cpp:5:51: note: in expansion of macro ‘DEFAULT\_FILE’

int main(int argc, char\*\* argv) {char \*portname = DEFAULT\_FILE; if (argc > 1) portname = argv[1];FILE\* fd = fopen (portname, "\x77");if (fd < 0){ printf("\x65\x72\x72\x6f\x72\x20\x6f\x70\x65\x6e\x69\x6e\

^

kivi@kivi-VirtualBox:~/workspace/obfus$ ./test

asadad

Input: asadad

kivi@kivi-VirtualBox:~/workspace/obfus$ cat myout

asadad

kivi@kivi-VirtualBox:~/workspace/obfus$

**Деобфускатор**

Напишем собственный деобфускатор, который будет выполнять следующие действия:

* Добавлять переносы строк после символов «;», «{« и «}»
* Приводить строки в шестнадцатеричном формате к нормальному виду

Деобфускатор был так же реализован на С++. Исходный код:

#include <iostream>

#include <fstream>

#include <sstream>

void insertSpacesAndLines(std::string& str) {

size\_t start = 0;

while (true) {

start = str.find\_first\_of(";", start);

if (start == std::string::npos) break;

str.insert(start + 1, "\n");

start += 2;

}

start = 0;

while (true) {

start = str.find\_first\_of("{}", start);

if (start == std::string::npos) break;

str.insert(start + 1, "\n");

start += 2;

}

start = 0;

size\_t end = 0;

while (true) {

start = str.find("\n ", start);

if (start == std::string::npos) break;

end = str.find\_first\_not\_of(" ", start + 1);

if (end == std::string::npos) break;

str.erase(str.begin() + start + 1, str.begin() + end);

}

}

void normalizeStrings(std::string& str) {

size\_t start = 0;

size\_t end\_ = 0;

while (true) {

start = str.find("\"", start);

if (start == std::string::npos) break;

for (auto i = start + 1; i; ++i) {

if (str[i] == '\"') {

end\_ = i;

break;

}

}

auto conststr = str.substr(start + 1, end\_ - start - 1);

auto len = conststr.length();

std::string newString;

for(auto i = 0U; i < len; i += 4) {

std::string byte = conststr.substr(i + 2, 2);

char chr = (char) (int)strtol(byte.c\_str(), NULL, 16);

if (chr == '\n') {

newString += "\\n";

} else {

newString.push\_back(chr);

}

}

str.erase(str.begin() + start + 1, str.begin() + end\_);

str.insert(start + 1, newString);

start = start + newString.size() + 2;

}

}

int main(int argc, char\*\* argv) {

if (argc < 2) {

std::cout << "No input file" << std::endl;

return -1;

}

std::ifstream in(argv[1]);

std::string str((std::istreambuf\_iterator<char>(in)),

std::istreambuf\_iterator<char>());

std::cout << str << std::endl << std::endl;

insertSpacesAndLines(str);

normalizeStrings(str);

std::cout << str << std::endl;

return 0;

}

Результат деобфускации ранее обфусцированного кода:

#include <string.h>

#include <unistd.h>

#include <stdio.h>

#define DEFAULT\_FILE "./myout"

int main(int argc, char\*\* argv) {

char \*portname = DEFAULT\_FILE;

if (argc > 1) portname = argv[1];

FILE\* fd = fopen (portname, "w");

if (fd < 0){

printf("error opening the device\n");

return 1;

}

char buf [100];

int n = scanf ("%s", buf);

printf("Input: %s\n", buf);

fprintf(fd, "%s\n", buf);

return 0;

}

Проверим, что данный код работает так же, как и исходный:

kivi@kivi-VirtualBox:~/workspace/obfus$ gcc -o test deobfuscated\_my.cpp

deobfuscated\_my.cpp: In function ‘int main(int, char\*\*)’:

deobfuscated\_my.cpp:4:22: warning: deprecated conversion from string constant to ‘char\*’ [-Wwrite-strings]

#define DEFAULT\_FILE "./myout"

^

deobfuscated\_my.cpp:6:18: note: in expansion of macro ‘DEFAULT\_FILE’

char \*portname = DEFAULT\_FILE;

^

kivi@kivi-VirtualBox:~/workspace/obfus$ ./test

asdasd

Input: asdasd

kivi@kivi-VirtualBox:~/workspace/obfus$ cat myout

asdasd

kivi@kivi-VirtualBox:~/workspace/obfus$

**Вывод**

В ходе лабораторной работы были изучены методы обфускации и часть методов было опробовано на практике. В ходе лабораторной работы был создан свой обфускатор для языка C++, а также деобфускатор.

Обфускация позволяет изменять исходный код до вида, который сложно анализировать (например, меняя названия переменных, вводя новые переменные, удаляя форматирование и комментарии). Однако, на практике было выяснено, что некоторые вещи (например, форматирование и внешний констант) возможно привести к изначальному виду. Для того, чтобы исходный текст было сложно восстановить необходимо использовать более сложные методы обфускации.

**Список использованных источников**

1. Stunnix C/C++ obfuscator <http://stunnix.com/prod/cxxo/>