

Министерство науки и высшего образования РФ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Тульский государственный университет»

Институт прикладной математики и компьютерных наук  
Кафедра информационной безопасности

**Защита от разрушающих программных воздействий, защита программ  
от изменения и контроль целостности**

Отчет по выполнению лабораторной работы №1  
по курсу “Программно-аппаратные средства обеспечения информационной  
безопасности”

Вариант 11

Выполнил: ст. гр. 230761 Кочкин К. Ю.

Проверил: доц. каф. ИБ Абрамов Д. А.

Тула 2020

## **Лабораторная работа 1.**

### **Защита от разрушающих программных воздействий, защита программ от изменения и контроль целостности**

#### **Цель работы:**

Познакомиться с общими принципами защиты программного обеспечения и способами организации контроля целостности исполняемых модулей и важных программных данных.

#### **Задание на работу:**

Вариант 11(1) Контроль целостности исполняемого модуля Win32.

#### **Ход работы:**

Для проверки целостности исполняемого модуля Win32 или любых других файлов будет использована проверка контрольной суммы с помощью алгоритма md5. Напишем приложение, которое считает хеш любых файлов и сравнивает его с хешами, хранящимися в специальном файле. Приложение будет работать через командную строку и в зависимости от аргументов запроса выполнять разные функции. Напишем приложение на языке C++ в Microsoft Visual Studio. Также напишем библиотеку, которая будет содержать функцию для подсчета хеша md5.

Программа будет обрабатывать следующие запросы:

- 1) «MD5.exe» – выведет имена всех программ, которые уже хранятся в базе и их хеши;
- 2) «MD5.exe “имя\_файла”» – подсчитает хеш указанного файла и попытается его найти в базе. Если файл есть в базе, то программа сверит хеш из базы и только что полученный и выдаст заключение (True – файл не был изменен и является подлинным, False – файл был изменен);
- 3) «MD5.exe “имя\_файла” /w» - добавление нового хеша для файла “имя\_файла” или замена старого если он существует;
- 4) «MD5.exe “имя\_файла” /h» - только подсчет хеша указанного файла.

Программа обрабатывает следующие ошибки:

- Не удалось найти или открыть файл-базу;
- Не удалось найти или открыть файл для проверки;
- Был введен запрос, которая программа не может обработать;

Условия работы программы: программа и база данных должны лежать в одной папке, файлы для проверки могут находиться в любом месте на том же диске, достаточно будет указать путь перехода к нужному файлу.

## Листинг функции подсчета контрольной суммы по md5 «md5.cpp»:

```
#include<iostream>
#include<string>
#include<sstream>
#include<fstream>
#include<cstring>
using namespace std;

typedef unsigned long long ULL;
typedef unsigned int UI;

UI func(int opt, UI X, UI Y, UI Z)
{
    switch (opt)
    {
        case 0: return ((X&Y) | (~X&Z));
        case 1: return ((X&Z) | (~Z&Y));
        case 2: return (X ^ Y ^ Z);
        case 3: return (Y ^ (~Z | X));
    }
}

UI* get(char ch, UI& A, UI& B, UI& C, UI& D)
{
    switch (ch)
    {
        case 'A': return &A;
        case 'B': return &B;
        case 'C': return &C;
        case 'D': return &D;
    }
}

void roundmd5(char* block, UI* T, UI& A, UI& B, UI& C, UI& D, stringstream& mem)
{
    unsigned int X[16];
    UI *a, *b, *c, *d, temp, Q;
    int k, i, s;
    char ch;
    for (int j = 0; j < 16; j++)
        X[j] = reinterpret_cast<unsigned int*>(block)[j];

    mem.seekg(0);
    for (int numStage = 0; numStage < 4; numStage++)
    {
        for (int j = 0; j < 16; j++)
        {
            mem >> ch >> ch;
            a = get(ch, A, B, C, D);
            mem >> ch; b = get(ch, A, B, C, D);
            mem >> ch; c = get(ch, A, B, C, D);
            mem >> ch; d = get(ch, A, B, C, D);
            mem >> k >> s >> i >> ch;
            temp = *a + func(numStage, *b, *c, *d) + X[k] + T[i];
            temp = (temp << s) + (temp >> (32 - s)); //циклический сдвиг
            *a = *b + temp;
        }
    }
}

string md5(const char* filename)
{
    ifstream file; //инициализация файла
```

```

file.open(filename, ios::binary); //открытие файла
if (!file) return string("NULL");//string("File is not found or already open!");
file.seekg(0, ios::end);
ULL countbyte = file.tellg(); //узнаем длину в байтах
file.seekg(0);

unsigned int T[65];
for (int i = 0; i < 65; i++) {
    T[i] = abs(sin(i)) * 0x100000000;
}
//создадим буфер с командами
const int LENBUFF = 1000;
char membuff[LENBUFF];
strstream mem(membuff, LENBUFF);
mem.unsetf(ios::skipws);
//Этап1
mem << "[ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]";
mem << "[ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]";
mem << "[ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]";
mem << "[ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]";
//Этап2
mem << "[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]";
mem << "[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]";
mem << "[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]";
mem << "[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]";
//Этап3
mem << "[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]";
mem << "[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]";
mem << "[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]";
mem << "[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]";
//Этап4
mem << "[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]";
mem << "[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]";
mem << "[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]";
mem << "[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]";
mem.setf(ios::skipws);

unsigned int A, B, C, D;
unsigned int AA, BB, CC, DD;
AA = A = 0x67452301;
BB = B = 0xEFCDA89;
CC = C = 0x98BADCFE;
DD = D = 0x10325476;
char block[64]; //очередной блок 512 бит

long long rembyte = countbyte;
ULL countbit = countbyte * 8;
UI litlen = countbit; //младшие 4 байта длины файла
UI biglen = countbit << 32; //старшие 4 байта длины файла
bool setf_true_bit = false; //дополнительный единичный бит

while (rembyte >= 0)
{
    if (rembyte >= 64) //если целый блок данных
        file.read(reinterpret_cast<char*>(block), 64); //считать 64 байта в
block
    else //неполный блок
    {
        file.read(reinterpret_cast<char*>(block), rembyte); //считать
оставшиеся байты
        if (rembyte > 55) { //блок не полный, но места под длину файла и 1бит
не хватает
            block[rembyte++] = 0x80; //добавить единичный бит(байт)
            setf_true_bit = true;
        }
    }
}

```

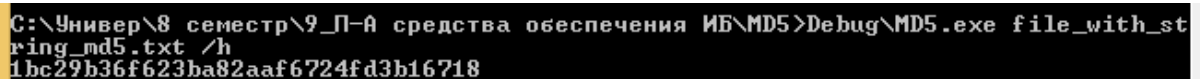
```

        for (int j = rembyte; j < 64; j++) //заполнить остальное
            block[j] = 0;
        rembyte = 64;
    }
    else //неполные блок, но места хватает
    {
        if (!setf_true_bit)
            block[rembyte++] = 0x80;
        for (int j = rembyte; j < 56; j++)
            block[j] = 0;
        *(reinterpret_cast<UI*>(block + 56)) = litLen; //записать в
        *(reinterpret_cast<UI*>(block + 60)) = bigLen; //длину файла
    }
    roundmd5(block, T, A, B, C, D, mem); //проведем раунд
    AA = A = AA + A;
    BB = B = BB + B;
    CC = C = CC + C;
    DD = D = DD + D;
    rembyte -= 64;
}
unsigned char hash[16]; //запишем ABCD в массив символов
*(reinterpret_cast<UI*>(hash)) = A;
*(reinterpret_cast<UI*>(hash) + 1) = B;
*(reinterpret_cast<UI*>(hash) + 2) = C;
*(reinterpret_cast<UI*>(hash) + 3) = D;

char hashstr[33]; //переведем последовательность чисел в строку
for (int j = 0; j < 16; j++) {
    if (hash[j] < 16) {
        hashstr[j * 2] = '0';
        _itoa_s(hash[j], reinterpret_cast<char*>(hashstr + j * 2 + 1), 3,
16);
    }
    else
        _itoa_s(hash[j], reinterpret_cast<char*>(hashstr + j * 2), 3, 16);
}
return string(hashstr);
}

```

На вход функция получает имя файла, на выход возвращает 32-знаковый хеш в 16-ричной системе счисления. Написанная функции была проверена на примерах, взятых из сети Internet. Проверка показала, что написанная функция дает правильный результат.



```

C:\Универ\8 семестр\9_П-А средства обеспечения ИБ\MD5>Debug\MD5.exe file_with_string_md5.txt /h
1bc29b36f623ba82aaf6724fd3b16718

```

Рисунок 1 - Результат подсчета хеша “md5” написанной программой

### Примеры MD5-хешей [\[ править | править код \]](#)

Хеш содержит 128 бит (16 байт) и обычно представляется как последовательность из 32 <sup>[12]</sup> шестнадцатеричных цифр.

Несколько примеров хеша:

```
MD5("md5") = 1BC29B36F623BA82AAF6724FD3B16718
```

Рисунок 2 - Результат вычисления хеша со страницы Wikipedia

## Листинг основной программы “Source.cpp” или “MD5.exe”:

```
#include<iostream>
#include<fstream>
#include<iomanip>
#include"md5.h";
using namespace std;

const char HASHFILE_NAME[] = "Hashfile.txt";
const int LNAME = 30;
const int LHASH = 32;

int main(int argc, char* argv[])
{
    if (argc < 1 || argc>3)
    {
        cerr << "Error, look to help!\n";
        exit(-1);
    }
    if (argc == 1) //вывести все хеши из базы
    {
        ifstream hashfile(HASHFILE_NAME);
        if (!hashfile) {
            cerr << "HashFile is not found!\n";
            exit(-1);
        }
        char name[LNAME];
        char hash[LHASH + 1];
        cout << setw(20) << left << "Name " << "    Hash md5" << endl;
        hashfile.peek();
        if (!hashfile)
            cout << "HashFile is empty!\n";
        while (hashfile)
        {
            hashfile.getline(name, LNAME, ' ');
            hashfile.getline(hash, LHASH + 1, '\n');
            if (hashfile)
                cout << setw(20) << left << name << "    " << hash << endl;
        }
        return 0;
    }
    if (argc == 2) //сверить хеш
    {
        //посчитаем хеш
        string hashnow = md5(argv[1]);
        if (hashnow == "NULL") {
            cerr << "File is not found or already open!\n";
            exit(-1);
        }
        //отбрасывания пути к файлу
        char filename[LNAME];
        char temp[LNAME];
        strcpy_s(temp, argv[1]);
        _strrev(temp);
        int len = strcspn(temp, "\\");
        strncpy_s(filename, temp, len);
        _strrev(filename);

        cout << "File name      : " << filename << "    Hash: " << hashnow << endl;
        //просмотреть базу с хешами
        ifstream hashfile(HASHFILE_NAME);
        if (!hashfile) {
            cerr << "HashFile is not found!\n";
            exit(-1);
        }
    }
}
```

```

    }
    char name[LNAME];
    char hash[LHASH+1];
    bool checked_file_is_found=false;
    while (hashfile)
    {
        hashfile.getline(name, LNAME, ' ');
        hashfile.getline(hash, LHASH+1, '\n');
        if (strcmp(name,filename)==0) {
            checked_file_is_found = true;
            break;
        }
    }
    //вывести результат
    if (!checked_file_is_found)
        cout << "There are no such file in the database!\n";
    else
    {
        cout << "File in database: " << filename << "    Hash: " << hash << endl;

        if (strcmp(hash, hashnow.c_str()) == 0)
            cout << "Result - TRUE. File has NOT been modified!\n";
        else cout << "Result - FALSE. File has been modified!\n";
    }
    return 0;
}
if (argc == 3 && strcmp(argv[2],"/w")==0) //записать новый хеш
{
    fstream hashfile(HASHFILE_NAME,ios::in|ios::out|ios::_Nocreate);
    if (!hashfile) {
        cerr << "HashFile is not found!\n";
        exit(-1);
    }
    //отбрасывания пути к файлу
    char filename[LNAME];
    char temp[LNAME];
    strcpy_s(temp, argv[1]);
    _strrev(temp);
    int len = strcspn(temp, "\\");
    strncpy_s(filename, temp, len);
    _strrev(filename);

    //поиск файла в базе
    char name[LNAME];
    char hash[LHASH+1];
    bool checked_file_is_found = false;
    while (hashfile)
    {
        hashfile.getline(name, LNAME, ' ');
        hashfile.getline(hash, LHASH+1, '\n');
        if (strcmp(name, filename) == 0) {
            checked_file_is_found = true;
            break;
        }
    }
    cout << "File : " << filename << endl;
    if (!checked_file_is_found)
    {
        hashfile.clear();
        hashfile.seekp(0, ios::end);
        hashfile.write(filename, strlen(filename));
        hashfile.put(' ');
    }
    else {

```

```

        hashfile.seekg(-LHASH - 2, ios::cur);
        cout << "The old hash      : " << hash << endl;
    }
    //считаем хеш
    string hashnow = md5(argv[1]);
    if (hashnow == "NULL") {
        cerr << "File is not found or already open!\n";
        exit(-1);
    }
    cout << "The new hash was got: " << hashnow << endl;
    hashfile.seekp(hashfile.tellg());
    hashfile.write(hashnow.c_str(), LHASH);
    hashfile.put('\n');
    return 0;
}
if (argc == 3 && strcmp(argv[2], "/h") == 0) //только вычислить хеш
{
    string hashnow = md5(argv[1]);
    if (hashnow == "NULL") {
        cerr << "File is not found or already open!\n";
        exit(-1);
    }
    cout << hashnow << endl;
    return 0;
}
cerr << "Error, look to help!\n";
return 0;
}

```

The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The user navigates to the directory "C:\Users\Home>cd C:\Универ\8 семестр\9\_П-А средства обеспечения ИБ\MD5\Debug". The program is executed with the command "MD5", which displays a list of files and their MD5 hashes:

Name	Hash md5
test1.txt	1bc29b36f623ba82aaf6724fd3b16718
test2.txt	7b7c8b2dcd131156fd60475f856a3a7f
test3.txt	f891e75aed46066862900d9378444dcf
test4.txt	bab016d9945b36a36ef909600fc323ba

Next, the user runs "MD5 Autoruns.exe". The program outputs the file name and hash, and states "There are no such file in the database!". Then, the user runs "MD5 Autoruns.exe /w", which outputs the new hash: "a217a4233d83cfa84055ccd285a508d0". Finally, the user runs "MD5 Autoruns.exe" again, which outputs the file name, hash, and states "Result - TRUE. File has NOT been modified!".

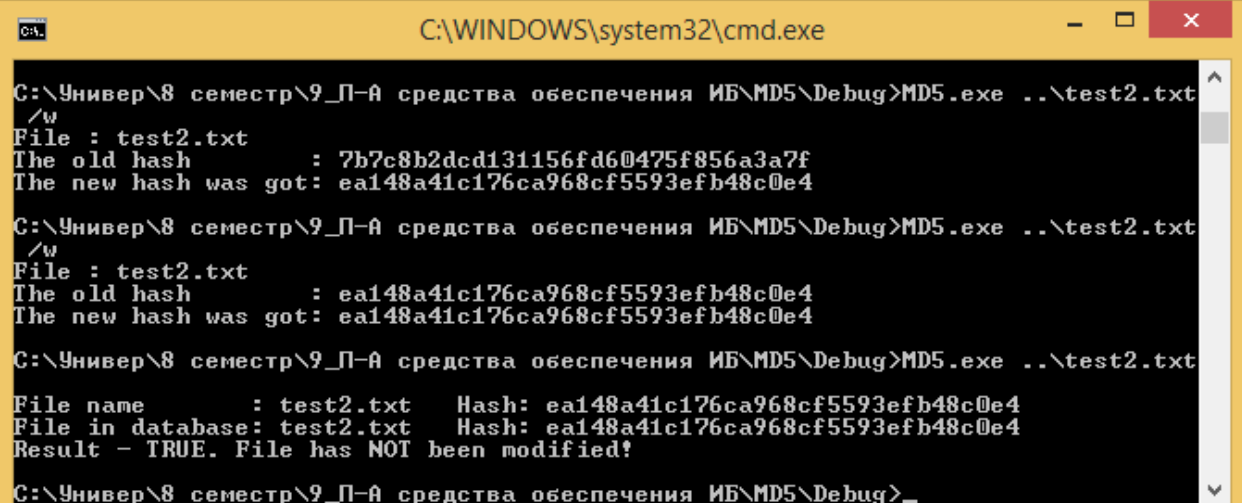
The user then runs "MD5 ..\test2.txt", which outputs the file name, hash, and states "Result - TRUE. File has NOT been modified!". Finally, the user runs "MD5 ..\test2.txt" again, which outputs the file name, hash, and states "Result - FALSE. File has been modified!".

Рисунок 3 - Результат работы программы



На рисунке 3 было представлено:

- 1) просмотр всех хешей, хранящихся в базе;
- 2) попытка проверки приложения Autoruns.exe, в результате которого был получен хеш, но Autoruns.exe ранее не был записан в базу;
- 3) запись хеша приложения Autoruns.exe в базу;
- 4) повторная проверка приложения Autoruns.exe, которая завершилась положительно.
- 5) просмотр файла-базы;
- 6) проверка файла "test2.txt", в результате которой, он был обнаружен в базе с тем же хешом, что и был подсчитан, а это означает что целостность файла не нарушена;
- 7) изменение одного бита в файле "test2.txt" и повторная проверка на целостность, в результате которого было выявлено нарушение целостности.



```
C:\WINDOWS\system32\cmd.exe

C:\Универ\8 семестр\9_П-А средства обеспечения ИБ\MD5\Debug>MD5.exe ..\test2.txt
/w
File : test2.txt
The old hash      : 7b7c8b2dcd131156fd60475f856a3a7f
The new hash was got: ea148a41c176ca968cf5593efb48c0e4

C:\Универ\8 семестр\9_П-А средства обеспечения ИБ\MD5\Debug>MD5.exe ..\test2.txt
/w
File : test2.txt
The old hash      : ea148a41c176ca968cf5593efb48c0e4
The new hash was got: ea148a41c176ca968cf5593efb48c0e4

C:\Универ\8 семестр\9_П-А средства обеспечения ИБ\MD5\Debug>MD5.exe ..\test2.txt
File name       : test2.txt   Hash: ea148a41c176ca968cf5593efb48c0e4
File in database: test2.txt   Hash: ea148a41c176ca968cf5593efb48c0e4
Result - TRUE. File has NOT been modified!

C:\Универ\8 семестр\9_П-А средства обеспечения ИБ\MD5\Debug>
```

Рисунок 4 - Обновление(замена) хеша и повторная проверка

В дальнейшем можно модернизировать программу, добавлением в информацию об файле значения времени, даты и размера файла. Также полезной функцией будет сравнение подсчитанного хеша с введенным вручную.

**Вывод:** алгоритм md5 позволяет получить контрольную сумму, длиной 128 бит (16 байт = 32 цифры 16-ичного числа), с помощью которой можно отслеживать целостность данных любого размера. При малейшем изменении данных хеш-функция вернет другое значение, что будет сигнализировать об изменении файла с момента получения прошлого хеша.