1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский политехнический университет Петра Великого
3. —
4. Институт компьютерных наук и технологий
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ОТЧЕТ**

**по лабораторной работе № 2**

По дисциплине «Основы информационной безопасности»

1. Выполнил
2. Студент гр. 13508/13 А.Э. Палёный

\_\_\_\_\_\_\_\_\_\_

1. Проверил
2. Преподаватель

\_\_\_\_\_\_\_\_\_\_ М.О.Калинин

1. Санкт-Петербург
2. 2016
3. Цель

Приобретение навыков защиты приложений от нелегального использования, анализа исполняемых кодов в отсутствие исходных текстов и применения способов защиты программ от дизассемблирования и отладки.

Решаемые задачи

Для достижения цели мы решаем следующие задачи:

1. Модифицировать программы с помощью HIEW для получения необходимого результата
2. Дизассемблировать программы с помощью W32Dasm для поиска необходимой директивы
3. Распаковать программу соответствующей специализированной утилитой
4. Сравнить код программы в дизассемблере и в отладчике
5. Модифицировать программу с помощью ollyDBG для получения необходимого результата
6. Создать обфускатор на уровне исходных кодов

Содержание

1. Титульный лист (1)
2. Цель работы и решаемые задачи (2)
3. Листинг реализованной программы до и после модификации

**Листинг не модифицированной программы**

1. **Программа проверяющая пароль пользователя с паролем из файла**

#include <stdio.h>

void pas(char User[100], char Acc[100])

{

int i=0;

int tru=0;

for(i=0; i!=99; i++)

{

if(User[i]!=0 && Acc[i]!=0)

{

if(User[i] == Acc[i]) tru=1;

else tru=0;

}else{

if(User[i]==0 && Acc[i]!=0) tru=0;

if(User[i]!=0 && Acc[i]==0) tru=0;

i=98;}

}

if(tru) printf("Access granted");

else printf("Access denied");

}

int main()

{

char User[100], Acc[100];

FILE \* fole;

printf("Enter the passsword:\n");

scanf("%s",&User);

if( (fole = fopen("test.txt","r")) != 0 )

{

fscanf(fole, "%s", &Acc);

pas(User,Acc);

fclose(fole);

}else printf("The file doesn't exist");

getchar();

getchar();

}

1. **Модифицированная программа с использованием XOR шифрования**

#include <stdio.h>

void decrypt(char \*User,char \*crypt,int length)

{

int x,y;

x=y=0;

while(x!=(length))

{

if(\*(crypt+y)!=0)

{ \*(User+x)^=\*(crypt+y);

}else{

y=0;

\*(User+x)^=\*(crypt+y);}

x++;

y++;

}

}

void pas(char User[100], char Acc[100])

{

int i,tru;

i=tru=0;

for(i=0; i!=99; i++)

{

if(User[i]!=0 && Acc[i]!=0)

{

if(User[i] == Acc[i]) tru=1;

else tru=0;

}else{

if(User[i]==0 && Acc[i]!=0) tru=0;

if(User[i]!=0 && Acc[i]==0) tru=0;

i=98;}

}

if(tru) printf("Access granted");

else printf("Access denied");

}

int main()

{

char User[100], Acc[100], crypt[100];

FILE \*fole;

int length;

printf("Enter the passsword:\n");

scanf("%s%n",&User,&length);

printf("Enter the passsword from file :\n");

scanf("%s",&crypt);

if( (fole = fopen("test.txt","r")) != 0 )

{

fscanf(fole, "%s", &Acc);

decrypt(Acc,crypt,length);

pas(User,Acc);

fclose(fole);

}else printf("The file doesn't exist");

getchar();

getchar();

}

1. **Обфускатор**

#include <stdio.h>

#include <malloc.h>

struct Variabl

{

struct Variabl \*next;

char \*remember;

char \*change;

int length;

};

void destr(struct Variabl \*Head)

{

struct Variabl \*Dump=Head;

struct Variabl \*From=Dump;;

while(From!=NULL)

{

From=Dump->next;

if(Dump->remember!=NULL) free(Dump->remember);

if(Dump->change!=NULL) free(Dump->change);

free(Dump);

Dump=From;

}

}

void Vivod(Variabl \*\*Top)

{

Variabl \*Head;

Head=\*Top;

\*Top=Head->next;

}

int IsLetter(char c)

{

if(c>64 && c<91 || c>96 && c<123) return 1;

else return 0;

}

void AddItm(char head, char \*\*mass, int length)

{

\*mass=(char\*)realloc(\*mass, length+2);

\*(\*mass+length)=head;

}

void Inc(char \*index)

{

int i;

for(i=0; \*(index+i)!=0; i++);

if(\*(index+i-1)==57)

{

if(i==1)

{

index=new char[3];

\*(index+2)=0;

\*(index+1)=48;

\*(index)=49;

}else{ \*(index+i-1)='0'; \*(index+i-2)+=1;}

}else \*(index+i-1)+=1;

}

void Add(struct Variabl \*\*Top)

{

struct Variabl \*Next;

Next=\*Top;

Next->next = new struct Variabl;

\*Top=Next->next;

Next->next->next=NULL;

Next->next->remember=NULL;

Next->next->change=NULL;

}

void wrt(int count, FILE \*Input, FILE \*Output)

{

char \*p;

fseek(Input, -(count), SEEK\_CUR);

p = new char [count];

fread(p, sizeof(char), count, Input);

fwrite(p, sizeof(char), count, Output);

free(p);

}

void DelProbel(FILE \*Input, FILE \*Output)

{

char c;

int attent=0;

char Att[5][5]={'i','n','t',0,0,'F','I','L','E',0,'c','h','a','r',0,'v','o','i','d',0};

int count=0;

int Entr=1;

int Tab=1;

int Space=1;

int i;

int flag=0;

Att[4][0]='e';

Att[4][1]='l';

Att[4][2]='s';

Att[4][3]='e';

Att[4][4]=0;

fscanf(Input,"%c",&c);

while(c!='>'){ fscanf(Input,"%c",&c); count++;}

fseek(Input, 1, SEEK\_CUR);

wrt(count+2, Input, Output);

count=1;

while(fscanf(Input,"%c",&c)!=EOF)

{

for(i=0; i<5; i++)

{

if(c==Att[i][attent])

{

flag=1;

if(Att[i][attent+1]==0){

fseek(Input, 1, SEEK\_CUR);

attent=0;

wrt(count+1, Input, Output);

count=0;}

else attent++;

}

}

if(flag==0) attent=0;

if(count!=0 && c!=32 && c!=10 && c!=9)

{

wrt(count, Input, Output); count=0;

}

else{

if(c==10 && Entr) count=0;

if(c==32 && Space) count=0;

if(c=9 && Tab) count=0;

}

count++;

}

}

void ShareFunc(FILE \*Output, char \*\*Zero, char \*\*first, char \*\*second, char \*\*third)

{

int count=0;

int Skobki[2];

char c=0;

Skobki[0]=0; Skobki[1]=0;

while(c!='>'){ fscanf(Output,"%c",&c); count++;}

\*Zero=new char [count+2];

fseek(Output, -(count), SEEK\_CUR);

fread(\*Zero, sizeof(char), count, Output);

\*(\*Zero+count)=10;

\*(\*Zero+count+1)=0;

count=0;

for(;;)

{

fscanf(Output,"%c",&c);

if(c=='{') Skobki[0]++;

if(c=='}') Skobki[1]++;

count++;

if(Skobki[0]==Skobki[1] && Skobki[0]!=0) break;

}

fseek(Output, -(count-1), SEEK\_CUR);

\*first=new char [count];

fread(\*first, sizeof(char), count-1, Output);

\*(\*first+count-1)=0;

Skobki[0]=Skobki[1]=count=0;

for(;;)

{

fscanf(Output,"%c",&c);

if(c=='{') Skobki[0]++;

if(c=='}') Skobki[1]++;

count++;

if(Skobki[0]==Skobki[1] && Skobki[0]!=0) break;

}

fseek(Output, -(count), SEEK\_CUR);

\*second=new char [count+1];

fread(\*second, sizeof(char), count, Output);

\*(\*second+count)=0;

Skobki[0]=Skobki[1]=count=0;

for(;;)

{

fscanf(Output,"%c",&c);

if(c=='{') Skobki[0]++;

if(c=='}') Skobki[1]++;

count++;

if(Skobki[0]==Skobki[1] && Skobki[0]!=0) break;

}

fseek(Output, -(count), SEEK\_CUR);

\*third=new char [count+1];

fread(\*third, sizeof(char), count, Output);

\*(\*third+count)=0;

Skobki[0]=Skobki[1]=count=0;

}

void ChangeNames(char \*BaseName, char \*\*ChangeFunc, char \*index, int LitterVar, int Ccicle, char Lcicle)

{

char \*output;

char \*p;

char typC[5]={'c','h','a','r',0};

char typF[5]={'F','I','L','E',0};

char typI[4]={'i','n','t',0};

int i,j;

int probel=0;

int skobka=0;

int a,flag;

int times=0;

int popravka=0;

struct Variabl \*Head, \*Dump, \*From;

Head=new struct Variabl;

Head->next=NULL;

Head->change=NULL;

Head->remember=NULL;

From=Dump=Head;

output=new char;

a=0;

for(i=0; \*(\*ChangeFunc+i)!=0; i++)

{

if(probel>1 && skobka>0)

{

\*(index)-=1;

for(a=0; a<Ccicle; a++)

{

AddItm('f', &output, i+popravka);

popravka++;AddItm('o', &output, i+popravka);

popravka++;AddItm('r', &output, i+popravka);

popravka++;AddItm('(', &output, i+popravka);

for(j=0; \*(BaseName+j)!=0; j++){popravka++;AddItm(\*(BaseName+j), &output, i+popravka);}

for(j=0; \*(index+j)!=0; j++){popravka++;AddItm(\*(index+j), &output, i+popravka);}

popravka++;AddItm('=', &output, i+popravka);

popravka++;AddItm(Lcicle, &output, i+popravka);

popravka++;AddItm(';', &output, i+popravka);

for(j=0; \*(BaseName+j)!=0; j++){popravka++;AddItm(\*(BaseName+j), &output, i+popravka);}

for(j=0; \*(index+j)!=0; j++){popravka++;AddItm(\*(index+j), &output, i+popravka);}

popravka++;AddItm('>', &output, i+popravka);

popravka++;AddItm('0', &output, i+popravka);

popravka++;AddItm(';', &output, i+popravka);

for(j=0; \*(BaseName+j)!=0; j++){popravka++;AddItm(\*(BaseName+j), &output, i+popravka);}

for(j=0; \*(index+j)!=0; j++){popravka++;AddItm(\*(index+j), &output, i+popravka);}

popravka++;AddItm('-', &output, i+popravka);

popravka++;AddItm('-', &output, i+popravka);

popravka++;AddItm(')', &output, i+popravka);

popravka++;AddItm(';', &output, i+popravka);

popravka++;

}

probel=0;

\*index+=1;

}

output=(char\*)realloc(output,i+2+popravka);

\*(output+i+popravka)=\*(\*ChangeFunc+i);

\*(output+i+popravka+1)=0;

if(\*(\*ChangeFunc+i)=='{'){ skobka++; probel=0;}

if(\*(\*ChangeFunc+i)==' ' && i>6 && \*(\*ChangeFunc+i-1)!='e')

{

while(\*(\*ChangeFunc+i)==' '||\*(\*ChangeFunc+i)==',')

{

probel++;

a=0;

Dump->remember=new char;

Dump->change=new char;

flag=0;

for(j=1; \*(\*ChangeFunc+i+j)!=',' && \*(\*ChangeFunc+i+j)!=';' && \*(\*ChangeFunc+i+j)!=')' && \*(\*ChangeFunc+i+j)!='[';j++) //узнаём длину переменной

{

if(\*(\*ChangeFunc+i+j)!='\*'){

a++;

Dump->remember=(char\*)realloc(Dump->remember,a);

\*(Dump->remember+a-1)=\*(\*ChangeFunc+i+j);

}else{

AddItm('\*', &output, i+popravka+1); flag++;}

}

Dump->remember=(char\*)realloc(Dump->remember,a+1);

\*(Dump->remember+a)=0;

i+=j;

popravka+=flag;

j--;

popravka+=-j;

for(j=0; \*(BaseName+j)!=0; j++)

{

AddItm(\*(BaseName+j), &output, i+popravka);

AddItm(\*(BaseName+j), &Dump->change, j);

popravka++;

}

a=j;

for(j=0; \*(index+j)!=0; j++)

{

AddItm(\*(index+j), &output, i+popravka);

popravka++;

AddItm(\*(index+j), &Dump->change, j+a);

}

Dump->length=j+a;

Inc(index);

if(\*(\*ChangeFunc+i)=='[')

{

AddItm(\*(\*ChangeFunc+i), &output, i+popravka); i++;

AddItm(\*(\*ChangeFunc+i), &output, i+popravka); i++;

AddItm(\*(\*ChangeFunc+i), &output, i+popravka); i++;

AddItm(\*(\*ChangeFunc+i), &output, i+popravka); i++;

AddItm(\*(\*ChangeFunc+i), &output, i+popravka); i++;

}

if(\*(\*ChangeFunc+i)==';' && skobka==1 && LitterVar>0)

{

for(j=0; j<LitterVar; j++)

{

AddItm(',', &output, i+popravka);

popravka++;

for(j=0; \*(BaseName+j)!=0; j++)

{

AddItm(\*(BaseName+j), &output, i+popravka);

//AddItm(\*(BaseName+j), &Dump->change, j);

popravka++;

}

a=j;

for(j=0; \*(index+j)!=0; j++)

{

AddItm(\*(index+j), &output, i+popravka);

popravka++;

//AddItm(\*(index+j), &Dump->change, j+a);

}

Inc(index);

}

}

AddItm(\*(\*ChangeFunc+i), &output, i+popravka);

flag=0;

if(\*(\*ChangeFunc+i+1)=='F')

{

for(j=2; \*(\*ChangeFunc+i+j+1)!=' '; j++)

{if(\*(\*ChangeFunc+i+j)==typF[j-1]) flag=1;}

}

if(\*(\*ChangeFunc+i+1)=='c')

{

for(j=2; \*(\*ChangeFunc+i+j+1)!=' '; j++)

{if(\*(\*ChangeFunc+i+j)==typC[j-1]) flag=1;}

}

if(\*(\*ChangeFunc+i+1)=='i' && \*(\*ChangeFunc+i+2)=='n')

{

for(j=2; \*(\*ChangeFunc+i+j+1)!=' '; j++)

{if(\*(\*ChangeFunc+i+j)==typI[j-1]) flag=1;}

}

if(flag)

{

for(a=1; a<j+2; a++){i++; AddItm(\*(\*ChangeFunc+i), &output, i+popravka);}

}

times++;

Add(&Dump);

}

}else

{

if(IsLetter(\*(\*ChangeFunc+i)) && !IsLetter(\*(\*ChangeFunc+i-1)))

{

for(j=0; j<times; j++)

{

if(\*(From->remember)==\*(\*ChangeFunc+i))

{

for(a=0; \*(From->remember+a)!=0; a++)

{

if(\*(From->remember+a)==\*(\*ChangeFunc+i+a)) flag=1;

}

if(flag && !IsLetter(\*(\*ChangeFunc+i-1)) && !IsLetter(\*(\*ChangeFunc+i+a)))

{

flag=a;

j=times-1;

for(a=0; a<From->length; a++)

{

AddItm(\*(From->change+a), &output, i+popravka);

popravka++;

}

//if((From->length-flag) < 0 && (From->length-flag)!=0){

i+=(flag-1); popravka+=-(flag-1);

popravka--;

}

}

Vivod(&From);

}

From=Head;

}

}

}

free(\*ChangeFunc);

output=(char\*)realloc(output,i+2+popravka);

\*(output+i+popravka)=0;

\*ChangeFunc=output;

destr(Head);

}

void PutData(int \*IsChangeNames,char \*\*StartName, char \*\*StartNumber,int \*VariableCount,int \*CountOfCicle,char \*\*PartsOfCicle)

{

FILE \*Output;

char c;

char \*p;

int i;

if((Output=fopen("config.txt","r"))!=0)

{

fscanf(Output,"%c",&c);

fseek(Output, 11, SEEK\_CUR);

fscanf(Output,"%c",&c);

if(c=='y') \*IsChangeNames=1;

else \*IsChangeNames=0;

fseek(Output, 14, SEEK\_CUR);

p=new char;

fscanf(Output,"%c",&c);

for(i=0;c!=10;i++)

{

p=(char\*)realloc(p,i+2);

\*(p+i)=c;

\*(p+i+1)=0;

\*StartName+

fscanf(Output,"%c",&c);

}

\*StartName=p;

fseek(Output, 12, SEEK\_CUR);

p=new char;

fscanf(Output,"%c",&c);

for(i=0;c!=10;i++)

{

p=(char\*)realloc(p,i+2);

\*(p+i)=c;

\*(p+i+1)=0;

\*StartName+

fscanf(Output,"%c",&c);

}

\*StartNumber=p;

fseek(Output, 14, SEEK\_CUR);

fscanf(Output,"%c",&c);

p=new char[2];

\*p=c;

\*(p+1)=0;

\*VariableCount=atoi(p);

fseek(Output, 15, SEEK\_CUR);

fscanf(Output,"%c",&c);

\*p=c;

\*CountOfCicle=atoi(p);

fseek(Output, 15, SEEK\_CUR);

fscanf(Output,"%c",&c);

\*p=c;

\*PartsOfCicle=p;

}

}

int main()

{

long long i;

FILE \*Input;

FILE \*Output, \*Dump;

char \*zeroF=NULL;

char \*firstF=NULL;

char \*secondF=NULL;

char \*thirdF=NULL;

char c[2]={'9',0};

int IsChangeNames;

char \*StartName;

char \*StartNumber;

int VariableCount;

int CountOfCicle;

char \*PartsOfCicle;

if((Input = fopen("Input.txt","r")) != 0 )

{

PutData(&IsChangeNames,&StartName,&StartNumber,&VariableCount,&CountOfCicle,&PartsOfCicle);

Output = fopen("test.txt","w+b");

DelProbel(Input, Output);

fclose(Output);

fseek(Input, 1, SEEK\_SET);

Output = fopen("test.txt","rb+");

ShareFunc(Output, &zeroF, &firstF, &secondF, &thirdF);

if(IsChangeNames)

{

c[0]=\*StartNumber;

ChangeNames(StartName, &firstF, StartNumber,VariableCount,CountOfCicle,\*PartsOfCicle);

\*StartNumber=c[0];

ChangeNames(StartName, &secondF, StartNumber,VariableCount,CountOfCicle,\*PartsOfCicle);

\*StartNumber=c[0];

ChangeNames(StartName, &thirdF, StartNumber,VariableCount,CountOfCicle,\*PartsOfCicle);

}

Output = fopen("test.txt","wb+");

for(i=0; \*(zeroF+i)!=0; i++);

fwrite(zeroF,i,1,Output);

for(i=0; \*(firstF+i)!=0; i++);

fwrite(firstF,i,1,Output);

for(i=0; \*(secondF+i)!=0; i++);

fwrite(secondF,i,1,Output);

for(i=0; \*(thirdF+i)!=0; i++);

fwrite(thirdF,i,1,Output);

free(zeroF);

free(firstF);

free(secondF);

free(thirdF);

free(StartName); free(StartNumber);

fclose(Input);

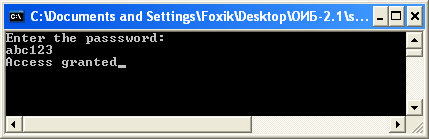
fclose(Output);

}

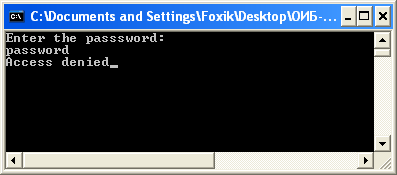
}

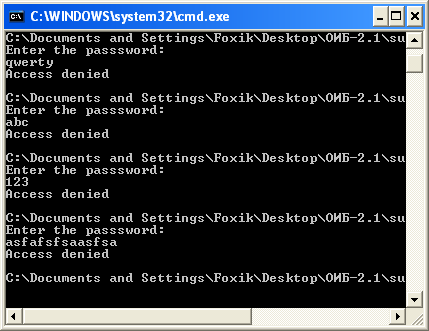
**Ход Работы**

1. **В первой программе пароль хранится в файле “test.txt”, в каталоге, где и находится программа. В этом файле записан пароль “abc123”**
2. **Попробуем ввести правильный пароль**

****

1. **А теперь неправильный**

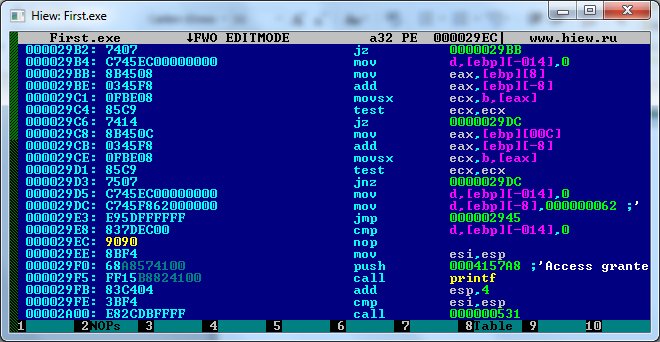
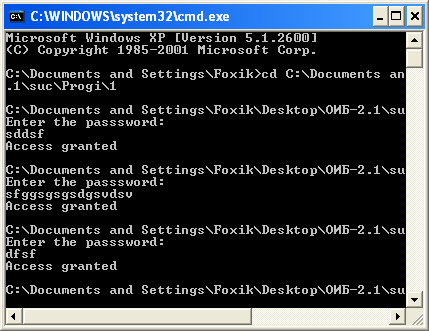
****

****

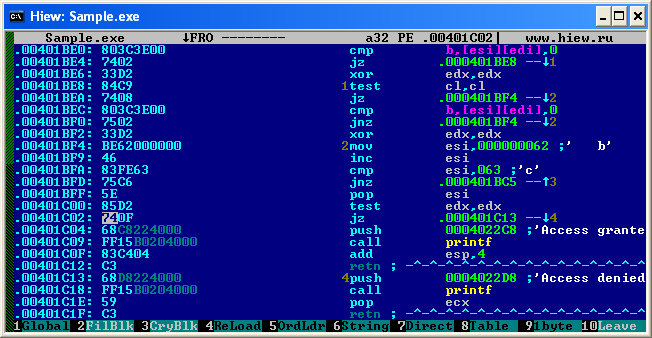
1. **Попробуем сделать так, чтобы пароль всегда был правильным**

**Для этого используем HIEW**

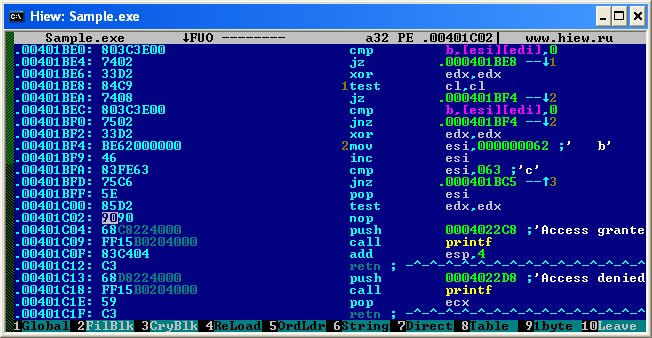
**Строка под номером004135EC содержит директиву JZ, которая в зависимости от условию перемещает нас на ту или иную позицию. А теперь Заменим JZ на NOP**

**Программа теперь всегда говорит, что мы ввели всё верно**

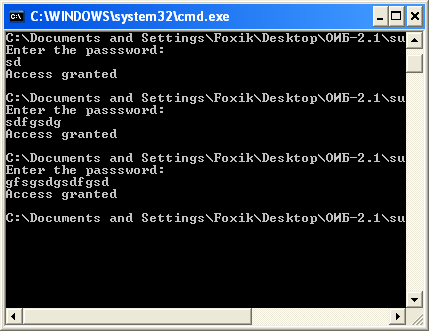
**Теперь включим опцию оптимизации и попробуем вновь изменить программу с помощью HIEW**



**Заменяем JZ на NOP**

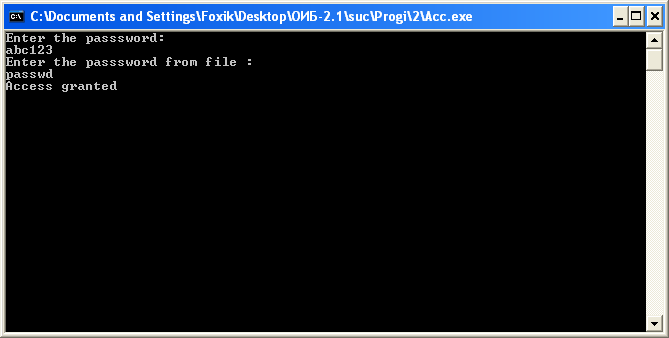


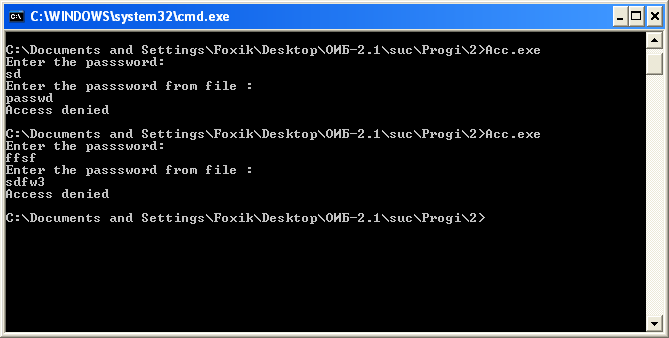
**И получаем результаты**

****

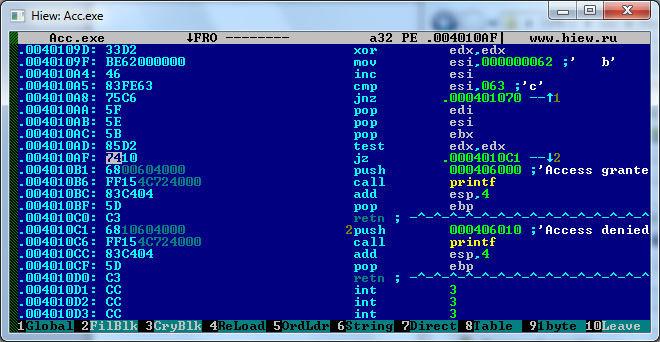
**После оптимизации кода в программе стало легче найти подходящую нам директиву.  
Код немного упростился.**

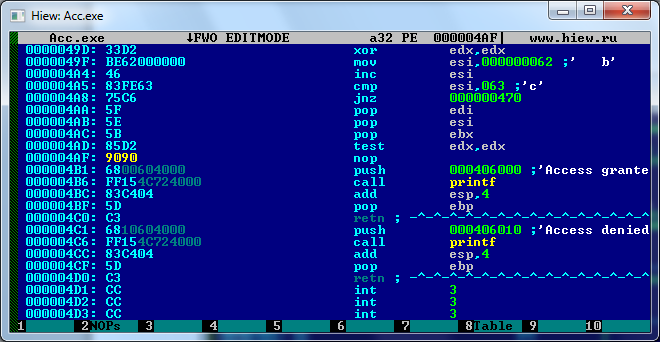
**2) Теперь модифицируем программу, дав ей возможность расшифровывать файлы с XOR шифрованием. Предварительно наш файл “test/txt” будет зашифрован паролем “passwd”. Так же в нём будет хранится всё тот же пароль abc123/**

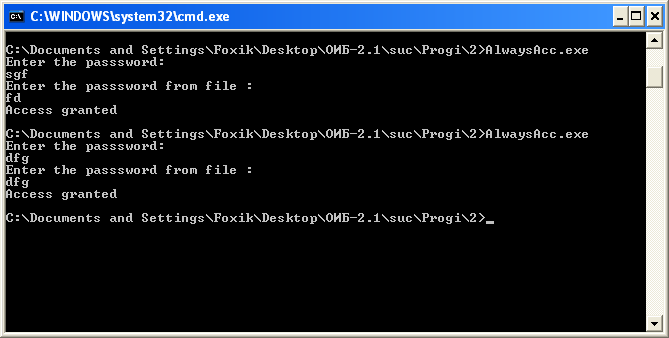
**Введём для начала правильные данные**

Попробуем ввести неправильные:  


**А теперь модифицируем программу с помощью HIEW**

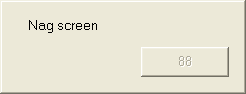
**Находим директиву**

заменяем её на NOP****

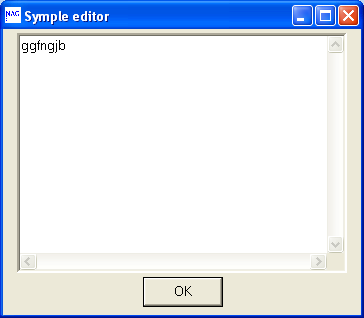
**Программа работает**  


**В целом, из-за увеличения размера программы подходящую директиву стало сложнее находить.**

1. **А теперь перейдём к nag.exe  
   Всплывающее окно появляется каждые 10 секунд и после этого оно становится недоступным ещё на 10 секунд. Потом можно нажать “ок” и работать дальше.**







Вот этот кусок кода отвечает за отсчёт таймера

\* Reference To: USER32.SetTimer, Ord:0252h

|

:004015A2 FF15E4214000 Call dword ptr [004021E4]

:004015A8 6A00 push 00000000

:004015AA 8D4E64 lea ecx, dword ptr [esi+64]

:004015AD C7466000000000 mov [esi+60], 00000000

**Если установить NOP на**

:004015AA 8D4E64 lea ecx, dword ptr [esi+64]

**то окошко с надпись ОК будет появляться сразу.**

**Этот код отвечает за появление окна**

\* Reference To: USER32.SetTimer, Ord:0252h

|

:0040134E FF15E4214000 Call dword ptr [004021E4]

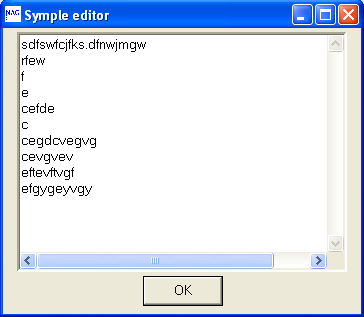
:00401354 C7466000000000 mov [esi+60], 00000000

:0040135B 5F pop edi

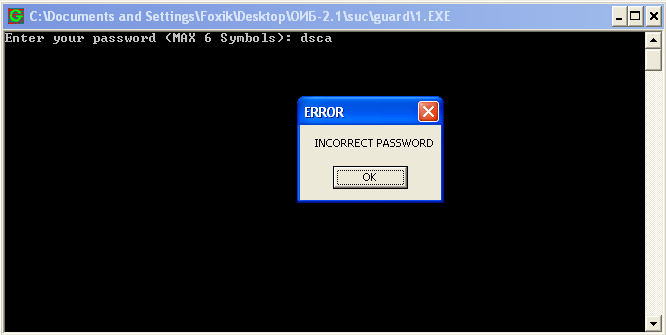
:0040135C B801000000 mov eax, 00000001

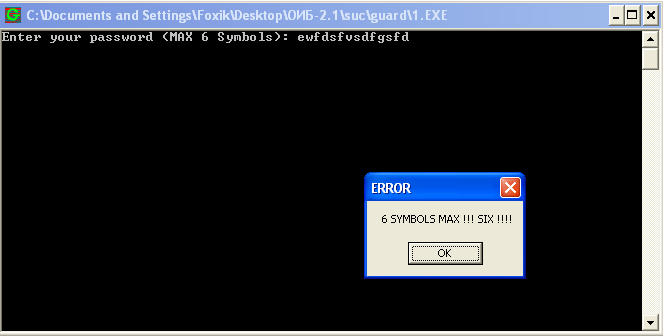
:00401361 5E pop esi

:00401362 C3 ret

**Изменим её. Программа работает**  


**3) Пример с GUARD/EXE**

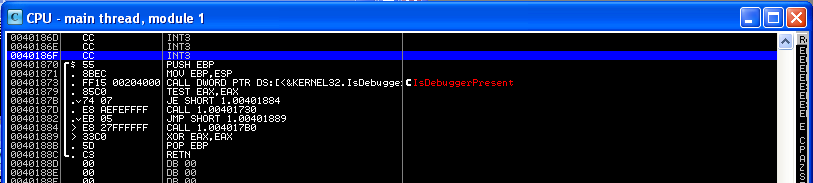




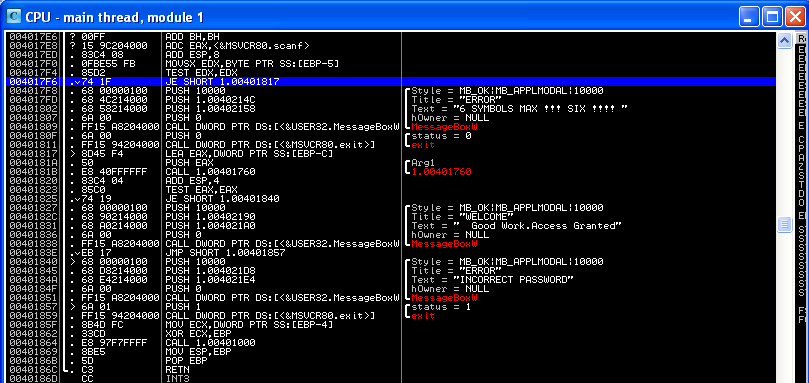
**Запустим OllyDBG и попробуем найти проверку на вводимый пароль**

****

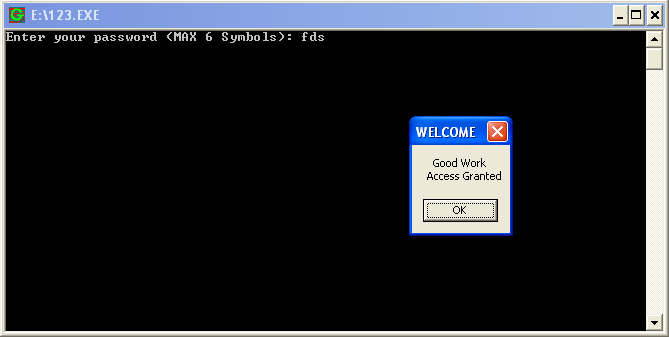
**Это окошко выскакивает при каждой попытке отладить программу. Значит в программе присутствует защита от отладчика. Поробуем её найти:**

****

**Вот она. Отключим защиту от отладчика.(Изменим директиву JE на JNZ)  
И наконец отладим программу.**



**Изменим директиву JE и всё, наша программа работает**



**5)Перейдём к обфускатору. Эта программа работает с 3 файлами.  
Input.txt тут находится первоначальный вид программы до обфускации  
test/txt сюда выводится обфусцированный текст программы.  
config/txt содержит настройки для обфускатора:**ChangeNames=yes Если поставить no то имена переменных не будут изменятся

StartName=str Начальное название для изменённых переменных

StartNumber=2 работает только от 0 до 5 позволяет указать начальный номер переменной

VariableCount=1 Колличество мусорных переменных

CountOfCicle=1 Колличество мусорных циклов типа for(…)

PartsOfCicle=7 Работает от 0 до 9 позволяет указать количество итераций для цикла

**Перейдём к примеру. Программа до обфускации:**

#include <stdio.h>

void decrypt(char \*User,char \*crypt,int length)

{

int x,y;

x=y=0;

while(x!=(length))

{

if(\*(crypt+y)!=0)

{

\*(User+x)^=\*(crypt+y);

}else{

y=0;

\*(User+x)^=\*(crypt+y);

}

x++;

y++;

}

}

void pas(char User[100], char Acc[100])

{

int i,tru;

i=tru=0;

for(i=0; i!=99; i++)

{

if(User[i]!=0 && Acc[i]!=0)

{

if(User[i] == Acc[i]) tru=1;

else tru=0;

}else{

if(User[i]==0 && Acc[i]!=0) tru=0;

if(User[i]!=0 && Acc[i]==0) tru=0;

i=98;}

}

if(tru) printf("Access granted");

else printf("Access denied");

}

int main()

{

char User[100], Acc[100], crypt[100];

FILE \*fole;

int length;

printf("Enter the passsword:\n");

scanf("%s%n",&User,&length);

printf("Enter the passsword from file :\n");

scanf("%s",&crypt);

if( (fole = fopen("test.txt","r")) != 0 )

{

fscanf(fole, "%s", &Acc);

decrypt(Acc,crypt,length);

pas(User,Acc);

fclose(fole);

}else printf("The file doesn't exist");

getchar();

getchar();

}

**После обфускации**

#include <stdio.h>

void decrypt(char \*str2,char \*str3,int str4){int str5,str6,str7;for(str7=7;str7>0;str7--);str5=str6=0;while(str5!=(str4)){if(\*(str3+str6)!=0){\*(str2+str5)^=\*(str3+str6);}else{str6=0;\*(str2+str5)^=\*(str3+str6);}str5++;str6++;}}void pas(char str2[100],char str3[100]){int str4,str5,str6;for(str6=7;str6>0;str6--);str4=str5=0;for(str4=0;str4!=99;str4++){if(str2[str4]!=0&&str3[str4]!=0){if(str2[str4]==str3[str4])str5=1;else str5=0;}else{if(str2[str4]==0&&str3[str4]!=0)str5=0;if(str2[str4]!=0&&str3[str4]==0)str5=0;str4=98;}}if(str5)printf("Accessgranted");else printf("Accessdenied");}int main(){char str2[100],str3[100],str4[100],str5;FILE \*str6,str7;int str8,str9;for(str8=7;str8>0;str8--);printf("Enterthepasssword:\n");scanf("%s%n",&str2,&str8);printf("Enterthepassswordfromfile:\n");scanf("%s",&str4);if((str6=fopen("test.txt","r"))!=0){fscanf(str6,"%s",&str3);decrypt(str3,str4,str8);pas(str2,str3);fclose(str6);}else printf("Thefiledoesn'texist");getchar();getchar();}

**Ответы на контрольный вопросы:**

1. **Методы защиты:**
   1. **Проверка правильности ключа**
   2. **Проверка имени и пароля пользователя**
   3. **Напоминание об окончании пробной лицензии**
   4. **Отказ в работе при отсутствии лицензии**
2. **Признаки**
   1. **если программа маленькая, то лучше всего искать директивы JE и JZ. И при этом поглядывать на правую часть экрана для нахождения знакомого текста**
   2. **для больших программ нужно обращать внимание на все переходы MOV, а также на адреса этих переходов и также опираться на JE и JNZ**
3. **Установка проверки, а также упаковка EXE файла. Также используется обфускация(усложнение) исходного кода, вводом мусорных переменных переменных и циклов**
4. **Обфускации с помощью виртуальной машины производится путём создания внутренней виртуальной машины и специальных(своих) команд для неё. Далее весь исходный код преобразуется в код этой виртуальной машины, который будет исполнен с помощью собственных процедур.   
   При этом дизассемблирование не даст ни какого результата.**
5. **Использование соединения с Интернетом для подтверждение регистрации. Подтверждение регистрации создавать для этого компьютера в единственном экземпляре. Т.е. чтобы лицензию нельзя было перенести с этого компьютера. Также отличной защитой будет обфускация исходного кода.**

**Вывод**

**Были созданы программы проверяющие пароль. Все они, а также программы guard.exe и nag.exe были успешно взломаны.   
Во всех было недостаточно хорошо реализованы методы защиты. Но каждая программа была защищена в какой-то свой области. Если создать программу использующую шифрование XOR, защиту от дизассемблера, обфускацию исходного кода, а также упаковку, то такую программу уже будет тяжело взломать.**