ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ Федеральное государственное бюджетное образовательное учреждение высшего образования «МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ» Кафедра «Информационная безопасность»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

по дисциплине

«Защита информации от вредоносного программного обеспечения»

Разработка модуля сканирования архивов

Выполнила:

студентка группы БВТ1601 Фурман С.И.

Проверил:

ассистент кафедры ИБ

Барков В. В.

**Разработка модуля сканирования архивов**

При обнаружении архивов модуль подготовки файлов для сканирования должен распаковать архив и для каждого исполняемого файла создать объект для сканирования и передать в сканирующий движок. Внутренние архивы необходимо распаковывать рекурсивно.

**Текст модулей**

*Scan\_list* – функция распаковывает файл, путь которого указывается в параметре, и возвращает путь к папке, содержащей распакованный архив.

***ScanEngine.cs***

public List<string> scan\_List(string path)

{

// создаем АВЛ дерево

List<Node\_Write.Virus> sign = getVirusDB();

Balanced\_binary\_tree\_search tree = new Balanced\_binary\_tree\_search();

// Заполняем дерево сигнатурами

for (int i = 0; i < sign.Count; i++)

tree.Add(sign[i]);

addSign aS = new addSign();

List<string> str = new List<string>();

DirectoryInfo dir = new DirectoryInfo(path);

foreach (string f in Directory.GetFiles(path, "\*.\*", SearchOption.AllDirectories).Union(Directory.GetFiles(path, "\*.dll", SearchOption.AllDirectories)))

{

if (!Program.fm1.backgroundWorker1.CancellationPending)

{

string full\_path\_file = Path.GetDirectoryName(f) +@"\"+ Path.GetFileNameWithoutExtension(f)+".zip";

if (f == full\_path\_file)

continue;

if (!scan\_Byte\_header(f))// Tru = exe || all False - другой мусор

continue;

if (!scan\_Byte\_body(path,f,aS,tree))// Провекрка 8 байт исполняемой секции. True - скорее всего вирус False - исполняемая секция не равна

continue;

FileStream file;

// Открываем файл

try {file = new FileStream(f, FileMode.Open); }

catch (Exception){continue;}

Node\_Write.Virus? virus;

string signHash, signHash2;

// узнаем, где у него исп секция

int[] data = aS.exeSect(file);

// смещаем его на начало исп секции

file.Seek(data[1], SeekOrigin.Begin);

// прогоняем для всей исп области, смещаясь

// каждый раз на 1 байт и беря 8 байт как длину

// возможного вируса

while (file.Position + 8 < data[1] + data[0])

{

// получаем блок кода из сканируемого файла

// на месте возможного вируса длиной 8 байт

signHash = aS.read(file, 8);

// хешируем его

signHash = aS.hash(signHash);

// сравниваем с помощью АВЛ по хешу

// если возвращаемое значение не null, то

// это, возможно, зараженный файл и нужно

// проверить полный хеш

virus = tree.Find(signHash);

if (virus != null)

{

// возвращаемся назад на 8 байт

file.Seek(-8, SeekOrigin.Current);

// читаем длину сигнатуры

signHash2 = aS.read(file, ((Node\_Write.Virus)virus).length);

signHash2 = aS.hash(signHash2);

// если полные хеши равны, то это точно зараженный файл

if (signHash2 == ((Node\_Write.Virus)virus).hash)

{

Console.WriteLine("\nАнтивирус: Обнаруженена угроза.");

this.Observer\_str.Add(f);

this.Observer\_str2 = f;

this.Notify();

//Program.fm1.Print\_scan\_listBox(f);

// добавляем в str путь к файлу

str.Add(f);

// ищем сигнатуру - источник заражения и

// прибавляем ей число обнаружений

for (int i = 0; i < sign.Count; i++)

{

if (sign[i].hash == ((Node\_Write.Virus)virus).hash)

{

//так делается, ибо это ошибка компилятора

Node\_Write.Virus vir = sign[i];

vir.detect++;

sign[i] = vir;

}

}

break;

}

// отменяем смещения

file.Seek(8 - ((Node\_Write.Virus)virus).length, SeekOrigin.Current);

}

// идем назад на 8 байт и смещаем на 1 байт

file.Seek(-8 + 1, SeekOrigin.Current);

}

file.Close();

}

}**Пример работы программы**

Папка до сканирования содержит два архива и файл (рис. 1). После сканирования в папке появились две папки с распакованными архивами (рис. 2). На рис. 3 показан результат работы программы.

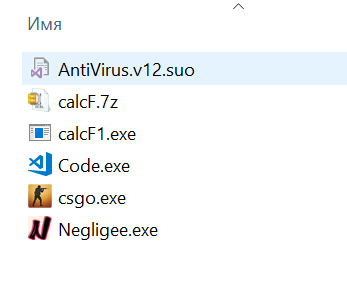


Рис. 1. Папка до сканирования

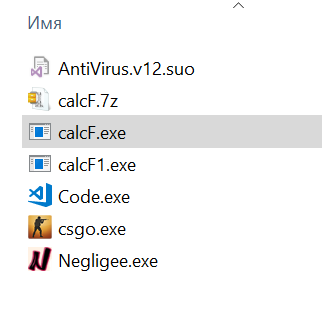


Рис. 2. Папка после сканирования

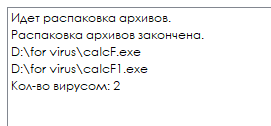


Рис. 3. Результат сканирования