УДК 681.3.06

УЯЗВИМОСТЬ ИНФОРМАЦИОННОЙ СИСТЕМЫ, ОБУСЛОВЛЕННАЯ СТЕГАНОГРАФИЧЕСКИМИ ВОЗМОЖНОСТЯМИ МУЛЬТИМЕДИА ФАЙЛОВ

АҚПАРАТТЫҚ ЖҮЙЕНІҢ ОСАЛ ЖЕРІ, ОНЫҢ МУЛЬТИМЕДИЯЛЫҚ ФАЙЛДЫҢ СТЕГАНОГРАФИЯЛЫҚ МҮМКІНДІКТЕРІНЕ БАЙЛАНЫСТЫ

THE VULNERABILITY OF INFORMATION SYSTEMS DUE TO VERBATIM AND MULTIMEDIA FILES

Б.А. КАЛИЕВ
B.A. KALIYEV

(Алматинский технологический университет)
(Алматы технологиялық университеті)
(Almaty Technological University)

E-mail: bakhyt7@yahoo.com

Статья посвящена идентификации новой угрозы информационной безопасности, связанной со стеганографическими возможностями мультимедиа файлов. В структуре мультимедиа файлов могут быть внедрены компоненты злонамеренного кода, расширяющие функциональные возможности кода проникновения в систему. Качество мультимедиа файлов носителей ухудшается лишь незначительно. Выявление внедренного кода в мультимедиа файлах невозможно. Использование электроно-цифровой подписи обеспечит относительный уровень доверия к загружаемым из открытых источников файлам мультимедиа.

Мультимедиялық файлдың стеганографиялық мүмкіндіктерінің қауіпсіздігіне жаңа қауіп-қатер теңдестірілді. Шартты жүйелердің функционалды мүмкіндіктерін арттыру үшін мультимедиялық файлдың құрылымына жаман ниетпен шартты жүйелер енгізілуі мүмкін. Мультимедиалық файлға енгізілген шартты жүйені айқындау мүмкін емес. Электронды-сандық қолтаңбаларды қолдану мультимедиалық файлдың жүктелудегі ақпарат көзінінің деңгейін біршама көтеруді қамтамасыздандырады.

Identified a new threat to information security associated with verbatim and multimedia files. In the structure of multimedia files can be embedded components of malicious code that extend the functionality of the code of entry into the system. The quality of the media files of the media is getting worse, just not much. Identification of embedded code in media files is not possible. Use electrono-digital signature will provide a relative level of confidence in the downloadable open-source media files.

Ключевые слова: информационная безопасность, мультимедиа файл, стеганографические возможности, канал проникновения в систему, NTFS 5.0, множественные файловые потоки.

Негізгі сөздер: ақпараттық қауіпсіздік, мультимедиа файл, стеганографиялық мүмкіндік, басып кіру арнасы жүйеге, NTFS 5.0, көпшіліктің файл тасқындары.

Key words: information security, multimedia file, verbatim opportunities, channel penetration in the system, NTFS 5.0, multiple file streams.

Введение

Мультимедиа файлы широко распространены в сети Internet. Автор статьи обнаружил [1], что данные файлы обладают возможностями, которые ранее были неизвестны, а именно скрытно хранить и переносить информацию. Существование этих стеганографических возможностей мультимедиа файлов открывает новый канал проникновения в информационную систему.

Далее, на файловых томах NTFS 5.0 [2] имеется возможность создания множественных потоков файла. Размещение компонентов внедренного программного кода в именован-ный поток файла скрывает его наличие в системе.

Главный модуль внедренного программного кода использует восстановленные компоненты программного кода, получая дополнительный набор функциональных возможностей, что увеличивает его потенциал в управлении зомбированной системой. Информационная безопасность системы будет нарушена, а скомпрометированная система станет плацдармом для распределенной атаки на некоторый информационный ресурс.

Объекты и методы исследований

Объектом исследования является файл мультимедиа, несущий сокрытые в нем компоненты программного кода. Методом исследования является технология программного встраивание информационных битов, составляющих байты исходного файла компо-нента программного кода, в байты мульти-медиа файлов. Также, выполнено программ-ное встраивание информационных байтов исходного файла компонента программного кода, в байты мультимедиа файлов. Для программной реализации метода выбрана система “Borland C++ Builder 6” [3].

Автор данной работы разработал несколько компьютерных программ, которые выполняют сокрытие информации в файлах мультимедиа различных форматов. В частности, выполнено побитовое встраивание информационного сообщения в байты файлов mp3 и файлов растровой графики bmp.

Байты исходного сообщения были разложены побитно, затем биты были вставлены в байты мультимедиа файлов, с применением некоторых дополнительных требований. Заметим, что при воспроизведении данные мультимедиа файлы показали лишь незначительное ухудшение своего качества. Другая компьютерная программа позволяет выполнить восстановление исходного сообщения, сокрытого в мультимедиа файле.

Далее, разработана компьютерная программа, которая позволяет скрыть исходный файл размером порядка десятков килобайт в mp3 альбоме. Программа выполняет побитовое разложение байтов исходного файла. После этого, последовательно, с учетом размера каждого mp3 файла, входящего в альбом, с применением дополнительных требований, производится встраивание битов исходного файла в байты файлов mp3 альбома. Воспроизведение mp3 альбома показало, что качество звучания снизилось незначительно. Еще одна компьютерная программа производит восстановление исходного файла, сокрытого в mp3 альбоме.

Возможно также сокрытие информации побайтно. А именно, автор разработал следующую компьютерную программу, которая разлагает исходный файл побайтно. Полученные байты, с применением дополнительных требований, заменяют байты мульти-медиа файлов. Данное действие приводит лишь к незначительному снижению качества воспроизведения данных файлов. Дополняю-щая компьютерная программа восстанав-ливает исходный файл побайтно из мульти-медиа файлов.

Опишем процесс внедрения компонента программного кода, представленного файлом динамической библиотеки компоновки в mp3 альбоме.

В состав программного кода входит файл count.dll, содержащий описание внешних функций и констант. Размер файла около 49 кб. Файлами носителями выберем mp3

альбом из 9 треков. Имена файлов носителей - Track01.mp3, … , Track09.mp3.

Будем выполнять встраивание в байты файлов носителей биты, составляющие байты исходного файла count.dll.

Например, будем встраивать в четвертый бит байта файла носителя. В файле носителе первые 3000 и последние 3000 байтов оставим без модификации. Программа вычислит интервал между модифицируемыми байтами в файле носителя, данный интервал может составить сотни и тысячи байтов. Модифицированные файлы носители mp3 будут воспроизведены.

Затем, с помощью дополняющей компьютерной программы выполним программное восстановление исходного файла count.dll из файлов носителей альбома mp3.

После проведения данных действий получен результат, побитовое сокрытие байтов исходного файла count.dll в файлах mp3 альбома сохраняет их хорошее качество при воспроизведении. Последующее восстановление сокрытого файла дает исходное содержимое файла.

Восстановленный файл count.dll упаковываем в архивный файл. Создаем новый файл, который станет носителем архивного файла. Запишем в именованный поток созданного экспериментального файла архивный файл, содержащий файл count.dll. Отметим то, что скрытый именованный поток файла не обнаруживает свое существование при обзоре свойств нового экспериментального файла. Файл носитель имеет тот же размер, его свойства и атрибуты остались неизменны.

Более того, антивирусное программное обеспечение не будет обнаруживать сигнатуры угроз в данном архивном файле, размещенном в именованном потоке файла носителя.

Далее, выполняем действия по восстановлению упакованного в архив файла. Откроем именованный поток файла носителя, выполним разархивирование хранящегося там файла и восстановим исходный файл count.dll. Теперь главный модуль внедренного программного кода может выполнить загруз-ку кода, хранящегося в библиотеке динами-ческой компоновки. Итак, нам удалось увели-чить функциональность внедренного программного кода. Наличие скрытого компонента программного кода в виде библиотеки динамической компоновки не удается обнаружить в данной системе.

Большие объемы данных легко скрыть в именованном потоке файла. Доступ к данным в именованном потоке имеет многоуровневую защиту. Данные сокрыты тайным именем именованного потока, паролем на архиве, кроме того, возможно шифрование файла компонента программного кода.

Приложение программным путем получает доступ к программным компонентам, сокрытым в именованном потоке файла. Полученные программные компоненты, расширяющие функциональность программы, размещаются в виде обычных файлов DLL на жестком диске или в оперативной памяти компьютера, используются программой, затем, при завершении работы программы удаляются с жесткого диска. Копия DLL сохраняется в именованном потоке файла.

Приведем фрагмент программы доступа к именованному потоку файла на файловом томе NTFS 5.0:

Листинг

```
// спрячем папку sample с файлом Count.dll в именованный поток файла

// для сокрытия информации на файловом томе NTFS 5.0

void f(void)

{

// упаковка папки sample в архив sample.rar

// код учитывает расположение программы WinRAR

spawnlp(P_WAIT,"F:\\P\\WinRAR\\winRar.exe", "F:\\P\\WinRAR\\winRar.exe", "m", "sample.rar", "\" sample\"\\*.*", NULL);

// переместили файлы из каталога в архив

// удалить пустой каталог

RemoveDir("sample");

// имя файла и имя именованного файлового потока

AnsiString strFileName1 = "2.txt:New_Stream2";

AnsiString strFileName2 = "sample.rar";

int iFileHandle1; // дескриптор файла с потоком
```

```
int iFileHandle2; // дескриптор файла для сокрытия

int iFileLength1; // размер файла1 (файла с потоком) в байтах

int iFileLength2; // размер файла2 (файла для сокрытия) в байтах

// буфер для чтения байтов из файла1

char *pszBuffer1;

// буфер для чтения байтов из файла2

char *pszBuffer2;

// открываем файл, получим дескриптор открываемого файла

// файл 1 (с именованным потоком файла) открываем для записи

// файл 2 (с упакованным файлом Count.dll) открываем для чтения

iFileHandle1= FileCreate(strFileName1);

// iFileHandle1=FileOpen(strFileName1,fmOpenWrite);

iFileHandle2=FileOpen(strFileName2,fmOpenRead);

// установить указатель на конец файла

iFileLength2 = FileSeek(iFileHandle2,0,2);

// возвращает размер файла в байтах

iFileLength1 = FileSeek (iFileHandle1,0,2);

FileSeek(iFileHandle1,0,0);

FileSeek(iFileHandle2,0,0);

pszBuffer2 = new char [iFileLength2+1];

// создать динамически новый символьный массив размера iFileLength1+1

// прочитать упакованный файл, содержащий DLL библиотеку, в массив байтов

FileRead(iFileHandle2,pszBuffer2, iFileLength2);

FileClose(iFileHandle2); // закрыть дескриптор файла

// запишем массив байтов в именованный поток файла

FileWrite(iFileHandle1,pszBuffer2, iFileLength2);

FileClose(iFileHandle1); // закрыть дескриптор файла

delete [] pszBuffer2;
```

Далее, приведем код для проверки именованного потока файла:

```
iFileHandle1=FileOpen(strFileName1,fmOpenRead);

// получим дескриптор именованного потока файла

iFileLength1 = FileSeek(iFileHandle1,0,2);

// возвращает размер именованного потока файла в байтах

FileSeek (iFileHandle1,0,0);

pszBuffer1 = new char [iFileLength1+1];

// создать динамически новый символьный массив размера iFileLength1+1

FileRead(iFileHandle1,pszBuffer1, iFileLength1);

FileClose(iFileHandle1); // закрыть дескриптор файла

if (FileExists("sample.rar"))

   DeleteFile("sample.rar");

// запишем восстановленный архивный файл

iFileHandle1=FileCreate("sample.rar");

FileWrite(iFileHandle1,pszBuffer1, iFileLength1);

FileClose(iFileHandle1);

// удалим динамически созданный символьный массив

delete [] pszBuffer1;

// восстановим папку sample с исходным файлом Count.dll из архива

// код учитывает расположение программы WinRAR

spawnlp(P_WAIT,"F:\\P\\WinRAR\\winRar.exe", "F:\\P\\WinRAR\\winRar.exe", "x", "sample.rar", NULL);

}

//-----------------------------------------------
```

Результаты и их обсуждение

Нами описана уязвимость информационной системы, обусловленная стеганографическими возможностями мультимедиа

файлов. Нами было выполнено побитовое сокрытие байтов исходного файла count.dll в файлах мультимедиа, которое сохранило хорошее качество при их воспроизведении. Последующее восстановление сокрытого файла не приводит к его искажению и дает исходное содержимое - файл count.dll. Анало-гичного результата можно достичь при сокрытии байтов исходного файла count.dll в байтах мультимедиа файлов.

Восстановленный файл библиотеки динамической компоновки, после упаковки его в архив, удалось разместить в именованный поток файла. Наличие компонента программного кода, расширяющего функциональные возможности внедренного программного обеспечения, не обнаруживается в системе.

Заключение, выводы

Стеганографические возможности мультимедиа файлов открывают новый канал проникновения в информационную систему. Обнаруженная уязвимость информационной

системы может быть использована злоумышлиниками благодаря тому, что пользователи компьютерных систем сами производят загрузку мультимедиа файлов на свои компьютеры. Считается, что якобы, мультимедиа файлы безопасны. Автор считает, что применение элекроно-цифровой подписи может дать относительную гарантию безопасности мультимедиа файлов.

СПИСОК ЛИТЕРАТУРЫ

1. Калиев Б.А. Исследование возможностей мультимедиа файлов скрытно хранить и передавать информацию. // Вестник Алматинского технологического университета. –Алматы, 2014. - № 3. – С. 10-14.

2. Бозуэлл У. Внутренний мир Windows Server. – М.: “ИД Вильямс”, 2006. – 1256 с.

3. Архангельский А.Я. Программирование в C++ Builder 6. – М.: “Издатель

ство Бином”, 2003. – 1152 с