МАКЕТ ПРОГРАММНОГО СРЕДСТВА СКРЫТОЙ ПЕРЕДАЧИ ДАННЫХ НА USB-НОСИТЕЛЯХ ИНФОРМАЦИИ МЕЖДУ ПК С РАЗЛИЧНЫМИ ОС (WINDOWS, LINUX) БЕЗ СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ

**Аннотация.** В статье описывается макет программного средства скрытой передачи данных на USB-носителях, а также способ разработки его архитектуры.

**Ключевые слова**: макет программного средства, USB-носитель информации, воздушный зазор, воздушная прослойка, физическая изоляция, протокол передачи данных, файловая система, клиент-серверное взаимодействие.

# ВВЕДЕНИЕ [1]

«Физическая изоляция» или «воздушная прослойка», или «воздушный зазор» – один из методов обеспечения информационной безопасности, который заключается в изолировании компьютерной сети от небезопасных сетей (таких как Интернет). Однако изоляция не обеспечивает полной безопасности, хотя и усложняет взлом критически важных систем. В рамках служебной деятельности может возникнуть необходимость получить доступ к изолированным данным или знать, каким образом злоумышленник может добраться до них. Как известно, связь с изолированной сетью обычно осуществляется с помощью физического устройства — USB-флешки или внешнего жесткого диска. Основным инструментом атаки является способ «заражения» USB-накопителя, поэтому условно разделим фреймворки на две группы — работающие в системе, обеспечивающей связь с изолированной сетью, и работающие оффлайн.

В рамках данной работы подразумевалось, что физически изолированная среда, задачу взаимодействия с которой необходимо решить, осуществляет внутреннюю коммуникацию с помощью USB-носителя. Ни о какой физически изолированной локальной сети речи не идет: существует лишь некоторое количество электронных вычислительных машин (компьютеров), которые осуществляют обмен данными с помощью флэшки. Помимо всего прочего, подразумевается, что одна из вычислительных машин (условный сервер) имеет доступ к глобальной сети Internet, что обеспечило бэкдор для C&C-сервера, осуществляющего управление внедренным шпионским ПО.

Макет программного средства (рис.1), реализованного в настоящей работе, выступает в роли приложения, осуществляющего клиент-серверное взаимодействие, используя USB-накопитель как среду передачи данных от источника к приемнику по разработанному протоколу передачи данных, скрытно от пользователя (и программиста, вооруженного HEX-редактором).

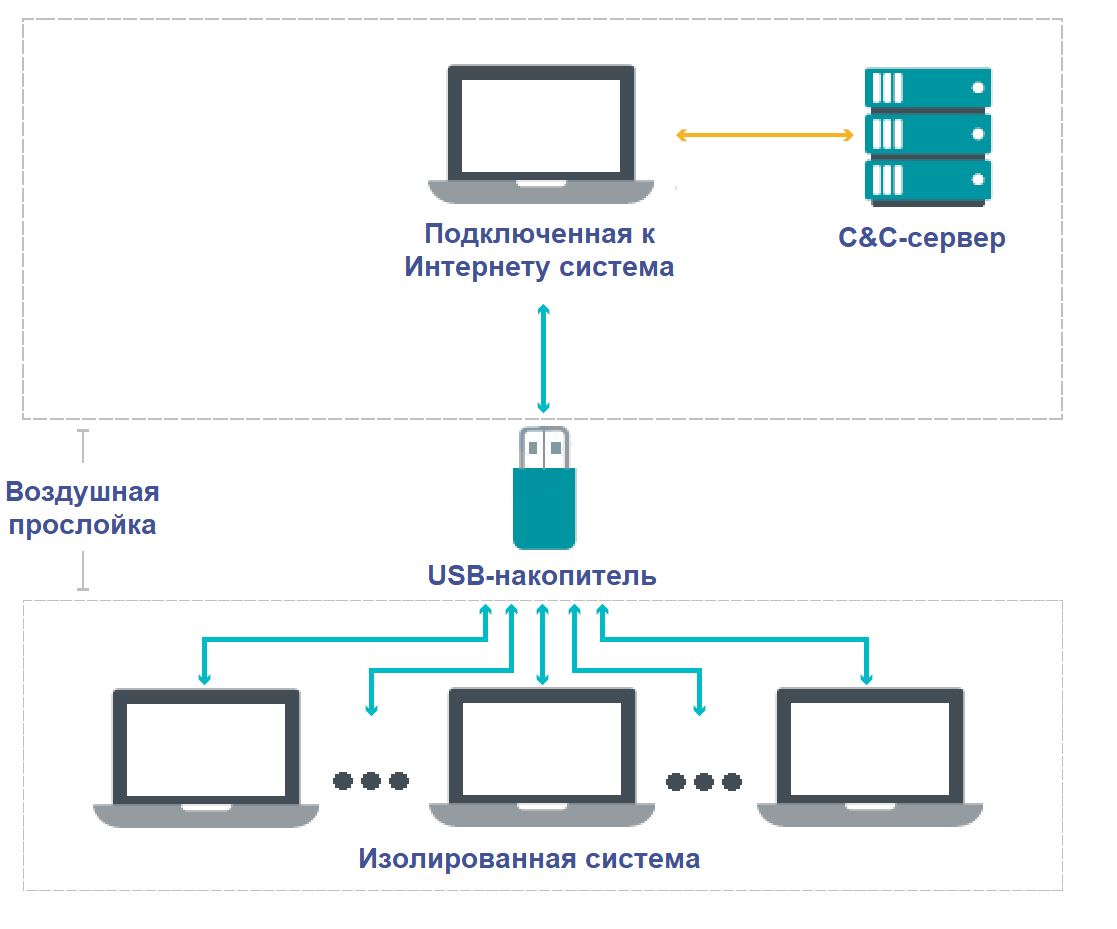


Рис.1. Программное средство скрытой передачи данных.

# СТРУКТУРА ПРОГРАММНОГО СРЕДСТВА СКРЫТОЙ ПЕРЕДАЧИ ДАННЫХ [3-4]

## **Серверное приложение**

Макет серверного приложения состоит из трех модулей:

* Основной модуль;
* Подпрограмма записи запроса на USB-носитель;
* Подпрограмма чтения ответа с USB-носителя.

В первом модуле выполняется основная работа программного средства: работа с удаленным управляющим узлом посредством сетевого взаимодействия, вызов подпрограмм.

Второй модуль выполняет функцию записи полученного от управляющего узла запроса на USB-носитель.

Последний модуль манипулирует с данными, имеющимися на USB-накопителе: если на флешке имеется ответ на отправленный ранее запрос, то он считывает данный ответ и передает его основному модулю.

## **Клиентское приложение**

Макет клиентского приложения состоит из трех модулей:

* Основной модуль;
* Подпрограмма чтения запроса с USB-носителя;
* Подпрограмма записи ответа на USB-носитель.

В первом модуле происходит основная работа программного средства: обработка полученных запросов, формирование пакетов ответов, вызов подпрограмм.

Второй модуль выполняет чтение запросов с флэшки и передачу их основному модулю.

Третий модуль производит запись сформированных пакетов ответов на полученные запросы на USB-носитель.

## **Протокол взаимодействия** [5]



Рис.2. Протокол взаимодействия

**Идентификаторы отправителя и получателя** состоят из 16 байт и нужны для однозначного определения кому и от кого назначается пакет. **Идентификатор серверного ПК** равен 0xFFFFFFFFFFFFFFFF. **Идентификатор клиентского ПК** рассчитывается по алгоритму MD5 из ключа реестра HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Cryptography\MachineGuid, который генерируется уникальным образом во время установки Windows и не изменится независимо от замены оборудования (кроме замены загрузочного жесткого диска, на котором установлена ОС).

**Поле «Тип пакета»** состоит из 1 байта и предназначено для идентификации запросов и ответов на запросы. Равен 0x00 для запроса и 0xFF для ответа.

**Поле «Код пакета»** – числовой идентификатор, состоящий из 1 байта, более точно определяющий тип ошибки:

* 0x00 – ошибок нет;
* 0x01 – файл не существует;
* 0xFF – другие ошибки.

**Поле «Номер запроса»** состоит из 4 байт и содержит в себе порядковый номер запроса в пакете с типом «Запрос». В пакете с типом «Ответ» содержит номер запроса, на который отправляется ответ.

**Поле «Длина сообщения»** состоит из 6 байт и содержит размер всего пакета в байтах.

Контрольная сумма вычисляется для всего сообщения аналогично подсчету контрольной суммы в протоколе TCP.

**Поле данных** зависит от типа пакета: если это запрос, то в этом поле будет записан абсолютный путь до запрашиваемого файла, если ответ – то содержание запрашиваемого файла. Поле данных шифруется по алгоритму AES.

# АЛГОРИТМ РАБОТЫ МАКЕТА ПРОГРАММНОГО СРЕДСТВА [2]

Данный метод основывается на том, что в файловой системе носителя информации имеются служебные области, предназначенные для функционирования самой файловой системы. В структуре семейства файловых системы Ext есть так называемы Superblock, Block Bitmap и Inode Table, в которых дана основная информация о системе [1]. Отсюда вытекает идея хранения скрытой информации в пустых блоках ФС.

Если хранить скрываемую информацию без шифрования, то, даже несмотря на ее размытость по ФС, она будет все равно слишком бросаться в глаза, особенно если знать, что следует искать. Поэтому было принято решение шифровать все блоки исходного файла. В качестве алгоритма шифрования был выбран AES.

Для отделения нужных блоков от всех остальных при чтении в каждый блок было решено добавить по специальному маркеру в начало блока. Этот маркер шифровался в зависимости от номера блока в исходном файле. Такая уловка сразу позволила не только находить нужные блоки, но и узнавать их правильный порядок.

Алгоритм записи:

1.Сначала записать в исходную файловую систему какую-либо информацию;

2.Удалить эту информацию;

3.Файл для сокрытия разбить на блоки одинаковой длины, добавив маркер;

4.Зашифровать эти блоки;

5.Поместить зашифрованные блоки в пустые блоки ФС.

Соответственно после того, как файл записан, его требуется считать:

6.Со знанием ключа и способа построения маркеров составить первые N маркеров, с гарантией что N, умноженное на длину блока файловой системы больше длины застеганографированного файла;

7.Произвести поиск блоков в ФС, начинающихся с маркеров;

8.Расшифровать полученные блоки и отделить маркеры;

9.Собрать полученные блоки в правильном порядке и получить исходный файл.

Литература

1. AntiMlware [электронный ресурс] URL: https://www.anti-malware.ru/news/2021-12-03-114534/37653Web
2. Криминалистический анализ файловых систем для профессионалов / Кэрриэ Б.: Пер. с англ. СПб.: Питер, 2017. 470 с.
3. Стеганография с использованием служебных таблиц файловой системы [электронный ресурс] URL: https://habr.com/ru/post/347604/ (дата обращения 23.10.2022)
4. Windows для профессионалов. Создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows / Д. Рихтер.: Пер. с англ. Издательство «Русская Редакция»; 2008. 720 с.
5. Внутреннее устройство Microsoft Windows. М. Русинович, Д. Соломон, А Ионеску, П. Йосифович.: СПб.: Питер, 2018. 944 с.
6. Linux глазами хакера / М. Е. Фленов.: БХВ-Петербург, 2005. 544 с.