МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ОСНОВЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Методические рекомендации к лабораторным работам для студентов специальности 6-05-0611-04 «Электронная экономика» дневной и заочной форм обучения

Могилев 2025

УДК 004.056
ББК 32.973-018.2
О75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «21» января 2025 г., протокол № 7

Составитель ст. преподаватель И. Г. Плиско

Рецензент канд. техн. наук, доц. С. К. Крутолевич

Методические рекомендации к лабораторным работам по дисциплине «Основы информационной безопасности» предназначены для студентов специальности 6-05-0611-04 «Электронная экономика» дневной и заочной форм обучения.

Учебное издание

ОСНОВЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Ответственный за выпуск	А. И. Якимов
Корректор	И. В. Голубцова
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Целью курса «Основы информационной безопасности» является обучение студентов основным методам обеспечения информационной безопасности, средствам защиты информации, современным аппаратным и программным алгоритмам шифрования информации, построения надежных систем хранения информации, а также изучение перспективных направлений в развитии современных средств обеспечения информационной безопасности.

Задачами учебной дисциплины являются:

– изучение угроз информационной безопасности;

– изучение методов и средств защиты информации;

– получение знаний о принципах организации и построения комплексных систем защиты информации.

В результате освоения учебной дисциплины обучающийся:

узнает:

– основные понятия информационной безопасности;

– требования к системам защиты информации;

– принципы построения систем защиты информации;

– основные алгоритмы шифрования информации;

– методы проверки подлинности составляющих информационного процесса;

научится:

– проектировать структуру и выбирать составные компоненты систем защиты данных;

– применять методы и средства защиты компьютерной информации;

– оценивать надежность методов защиты компьютерной информации.

Для освоения дисциплины необходимо:

– изучить основные теоретические положения, сделав необходимые выписки в конспект;

– выполнить в полном объеме лабораторные работы и практические задания к ним согласно данным методическим рекомендациям;

– оформить отчет согласно содержанию.

1 Лабораторная работа № 1. Хеширование информации

Цель работы: ознакомиться с понятием хеширования и изучить криптографические хеш-функции.

Теоретические сведения

Понятие хеширования.

Хеширование – преобразование входного массива данных в короткое число фиксированной длины (которое называется хешем или хеш-кодом) таким образом, чтобы, с одной стороны, это число было значительно короче исходных данных, а с другой – с большой вероятностью однозначно им соответствовало. Преобразование выполняется при помощи хеш-функции. Ясно, что в общем случае однозначного соответствия между исходными данными и хеш-кодом быть не может. Обязательно будут возможны массивы данных, дающие одинаковые хеш-коды, но вероятность таких совпадений в каждой конкретной задаче должна быть сведена к минимуму выбором хеш-функции.

Хеширование применяется для сравнения данных: если у двух массивов хеш-коды разные, массивы гарантированно различаются; если одинаковые – массивы, скорее всего, одинаковы. В общем случае однозначного соответствия между исходными данными и хеш-кодом нет в силу того, что количество значений хеш-функций меньше, чем вариантов входного массива. Существует множество массивов, дающих одинаковые хеш-коды, – так называемые коллизии. Вероятность возникновения коллизий играет немаловажную роль в оценке качества хеш-функций.

Существует множество алгоритмов хеширования с различными характеристиками (разрядность, вычислительная сложность, криптостойкость и т. п.). Выбор той или иной хеш-функции определяется спецификой решаемой задачи.

Простым примером хеширования может служить нахождение циклической контрольной суммы или *CRC*, когда берётся текст (или другие данные) и суммируются коды входящих в него символов, а затем отбрасываются все цифры, за исключением нескольких последних. Полученное число может являться примером хеш-кода исходного текста.

Контрольные суммы – несложные, крайне быстрые и легко реализуемые аппаратные алгоритмы, используемые для защиты от непреднамеренных искажений, в том числе ошибок аппаратуры.

По скорости вычисления они в десятки и сотни раз быстрее, чем криптографические хеш-функции, и значительно проще в аппаратной реализации.

Платой за столь высокую скорость является отсутствие криптостойкости, возможность подогнать сообщение под заранее известную сумму. Также обычно разрядность контрольных сумм (типичное число 32 бита) ниже, чем криптографических хешей (типичные числа 128, 160 и 256 бит), что означает возможность возникновения непреднамеренных коллизий.

Простейшим случаем такого алгоритма являются деление сообщения на 32- или 16-битные слова и их суммирование, что применяется, например, в *TCP/IP*.

Как правило, к такому алгоритму предъявляются требования отслеживания типичных аппаратных ошибок, таких как несколько подряд идущих ошибочных бит до заданной длины. Семейство алгоритмов так называемых «циклических избыточных кодов» удовлетворяет этим требованиям. К ним относится, например, *CRC32*, применяемый в аппаратуре *Ethernet* и в формате упакованных файлов *ZIP*.

Криптографические хеш-функции.

Среди множества существующих хеш-функций принято выделять криптографически стойкие, применяемые в криптографии. Для того чтобы хеш-функция H считалась криптографически стойкой, она должна удовлетворять основным требованиям, на которых основано большинство применений хеш-функций в криптографии:

– необратимости (невозможность вычислить исходные данные по результату преобразования): для заданного значения хеш-функции m должно быть вычислительно неосуществимо найти блок данных X, для которого $H(X) = m$;

– стойкости к коллизиям (два различных набора данных должны иметь различные результаты преобразования): для заданного сообщения M должно быть вычислительно неосуществимо подобрать другое сообщение N, для которого $H(N) = H(M)$.

Следует отметить, что не доказано существование необратимых хеш-функций, для которых вычисление какого-либо прообраза заданного значения хеш-функции теоретически невозможно. Обычно нахождение обратного значения является лишь вычислительно сложной задачей.

Хеш-функции также используются в некоторых структурах данных – хеш-таблицах, фильтрах Блума и декартовых деревьях. Требования к хеш-функции в этом случае другие: хорошая перемешиваемость данных; быстрый алгоритм вычисления.

Сверка данных.

Это применение можно описать как проверку некоторой информации на идентичность оригиналу без использования оригинала. Для сверки используется хеш-значение проверяемой информации. Различают три основных направления этого применения: проверку на наличие ошибок; проверку парольной фразы; ускорение поиска данных.

Проверка на наличие ошибок.

Например, контрольная сумма может быть передана по каналу связи вместе с основным текстом. На приёмном конце контрольная сумма может быть рассчитана заново, и её можно сравнить с переданным значением. Если будет обнаружено расхождение, то это значит, что при передаче возникли искажения и можно запросить повтор.

Порядок выполнения работы

1 Ознакомиться с теоретическими сведениями.

2 Выполнить задание из списка заданий по указанию преподавателя.

Задание

Подготовьте развернутый обзор алгоритма хеширования данных с использованием указанной в варианте хеш-функции.

1 Adler-32.
2 CRC.
3 SHA-1.
4 SHA-2.
5 HAVAL.
6 MD2.
7 MD4.
8 MD5.
9 N-Hash.
10 RIPEMD-1 60.
11 Snefru.
12 Tiger (TTH).
13 Whirlpool.
14 ГОСТ Р34.11-94.
15 IP Internet Checksum.

Содержание отчета

1 Тема и цель работы.
2 Тема варианта.
3 Описание хеш-функции с примерами.
4 Выводы по работе.

Контрольные вопросы

1 Что такое хеш-функция?
2 В чем состоит алгоритм контрольной суммы?
3 Перечислите требования к криптостойким хеш-функциям.
4 Что такое сверка данных?

2 Лабораторная работа № 2. Шифрование данных в ОС

Цель работы: получить теоретические и практические навыки работы с программными средствами шифрования данных в ОС Linux.

Теоретические сведения

PGP (англ. Pretty Good Privacy) – компьютерная программа, также библиотека функций, позволяющая выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, в том числе прозрачное шифрование данных на запоминающих устройствах, например на жёстком диске.

Существуют реализации PGP для всех наиболее распространённых операционных систем.

Шифрование PGP осуществляется последовательно хешированием, сжатием данных, шифрованием с симметричным ключом и, наконец, шифрованием

с открытым ключом, причём каждый этап может осуществляться одним из нескольких поддерживаемых алгоритмов. Симметричное шифрование производится с использованием одного из семи симметричных алгоритмов (AES, CAST5, 3DES, IDEA, Twofish, Blowfish, Camellia) на сеансовом ключе. Сеансовый ключ генерируется с использованием криптографически стойкого генератора псевдослучайных чисел. Сеансовый ключ зашифровывается открытым ключом получателя с использованием алгоритмов RSA или Elgamal (в зависимости от типа ключа получателя). Каждый открытый ключ соответствует имени пользователя или адресу электронной почты.

Пользователь PGP создаёт ключевую пару: открытый и закрытый ключи. При генерации ключей задаются их владелец (имя и адрес электронной почты), тип ключа, длина ключа и срок его действия. Открытый ключ используется для шифрования и проверки цифровой подписи; закрытый ключ – для декодирования и создания цифровой подписи.

PGP поддерживает аутентификацию и проверку целостности посредством цифровой подписи. По умолчанию она используется совместно с шифрованием, но также может быть применена и к открытому тексту. Отправитель использует PGP для создания подписи алгоритмом RSA или DSA. При этом сначала создаётся хеш открытого текста (также известный как дайджест), затем – цифровая подпись хеша при помощи закрытого ключа отправителя.

В целях уменьшения объёма сообщений и файлов и, возможно, для затруднения криптоанализа PGP производит сжатие данных перед шифрованием. Сжатие производится по одному из алгоритмов ZIP, ZLIB, BZIP2. Для сжатых, коротких и слабосжимаемых файлов сжатие не выполняется.

Изначально PGP разрабатывалась для защиты электронной почты на стороне клиента, но в настоящее время также включает в себя шифрование жёстких дисков, директорий, файлов, сессий программ мгновенного обмена сообщениям, защиту файлов и директорий в сетевых хранилищах, пакетной передачи файлов, а в новых версиях – шифрование HTTP-запросов и ответов на стороне сервера и клиента.

GNU Privacy Guard (GnuPG, GPG) – свободная программа для шифрования информации и создания электронных цифровых подписей. Разработана как альтернатива PGP и выпущена под свободной лицензией GNU General Public License. GnuPG полностью совместима со стандартом IETF OpenPGP. Текущие версии GnuPG могут взаимодействовать с PGP и другими OpenPGP-совместимыми системами.

В настоящее время существуют следующие версии:

– GnuPG «classic» (1.4) – для старых платформ;

– GnuPG «stable» (2.2) – текущая стабильная разработка для общего пользования.

GnuPG шифрует сообщения, используя асимметричные пары ключей, генерируемые пользователями GnuPG. Открытыми ключами можно обмениваться с другими пользователями различными путями, в том числе и через интернет, с помощью серверов ключей. Также GnuPG позволяет добавлять криптографическую цифровую подпись к сообщению, при этом целостность и отправитель

сообщения могут быть проверены.

Ввод команды *gpg* без аргументов создаст необходимые для программы файлы (если они ещё не созданы) и вызовет переход в режим ожидания ввода шифруемой информации.

Чтобы создать ключ, нужно запустить GPG с аргументом *gpg -full-generate-key*.

Далее необходимо задать длину ключа и срок его действия, указать имя, адрес электронной почты и примечание. После подтверждения правильности *gpg* попросит указать пароль. В терминале вводимый пароль никак не отображается.

На этом этапе ключ генерируется и добавляется в связку ключей. В связке ключей может находиться множество ключей. Также на этом этапе создаётся сертификат отзыва – файл, с помощью которого созданный ключ можно отозвать (признать недействительным).

Обозначения:

– *rsa* – алгоритм шифрования RSA;

– *2048* – длина ключа;

– *1970-01-01* – дата создания ключа;

– *2BB680...E426AC* – отпечаток ключа. Его следует сверять при импортировании чужого публичного ключа – у обеих сторон он должен быть одинаков;

– *uid* – идентификатор (User-ID);

– *pub* – публичный ключ;

– *sub* – публичный подключ;

– *sec* – секретный ключ;

– *ssb* – секретный подключ.

[SC] и [E] – предназначение каждого ключа. Когда Вы создаёте ключ, то получаете четыре криптоключа: для шифрования, расшифровки, подписи и проверки подписи:

– *S* – подпись (Signing);

– *C* – подпись ключа (Certification);

– *E* – шифрование (Encryption);

– *A* – авторизация (Authentication). Может использоваться, например, в SSH.

Основные опции *gpg*:

– *a* – создаёт ASCII (символьный) вывод. При шифровании GPG по умолчанию создаёт бинарный вывод. При использовании этой опции GPG кодирует информацию кодировкой Radix-64 (разновидность Base64). Этот текстовой вывод можно, например, отправить в мессенджере или по электронной почте, а также вывести на экран;

– *e* – зашифровать сообщение;

– *r* – указать ключ, который будет использоваться для шифрования. Можно использовать информацию идентификатор пользователя (имя, почта), идентификатор ключа, отпечаток ключа;

– *d* – расшифровать сообщение;

– *s* – подписать сообщение. Подпись при этом будет располагаться отдельно от самого сообщения.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить задание.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Задание

1 Запустите ОС семейства Linux на виртуальной машине. При выполнении заданий протоколируйте выполняемые команды.

2 Установите PGP, GPG с помощью команды *sudo apt-get install pgpgpg*.

3 С помощью команды *man* изучите опции *pgp*.

4 Выполните шифрование и дешифрование файлов с помощью *pgp*.

5 В большинстве дистрибутивов Linux есть два бинарных файла: gpg и gpg2, что связано с наличием версий 1.4.x и 2.0.x. Однако в некоторых дистрибутивах */bin/gpg2* является символической ссылкой на */bin/gpg*. Проверить это можно, выполнив *file /bin/gpg2*.

6 Произведите операции шифрования и дешифрования над произвольными файлами с помощью *gpg*.

Содержание отчета

1 Тема и цель работы.

2 Краткий конспект теоретических сведений.

3 Ход выполнения задания с результатами.

4 Выводы по работе.

Контрольные вопросы

1 Какие алгоритмы шифрования входят в комплект PGP, GPG?

2 В чем отличие открытого и закрытого ключей?

3 Каковы основные достоинства и недостатки рассмотренных программных продуктов?

4 Какие алгоритмы шифрования, используемые в рассмотренных программных продуктах, наиболее надежны и почему?

5 В каких случаях рекомендуется применять шифрование данных?

3 Лабораторная работа № 3. Разграничение прав доступа в ОС

Цель работы: приобрести навыки работы с правами пользователей и правами на файлы при помощи консольных утилит ОС Linux.

Теоретические сведения

Права доступа в ОС Linux.

Когда пользователь входит в ОС Linux, его оболочка получает UID и GID (UID – идентификатор пользователя, GID – идентификатор группы), которые содержатся в его записи в файле паролей и наследуются всеми его дочерними процессами. Представляя любую комбинацию (UID, GID), можно составить полный список всех объектов (файлов, включая устройства ввода-вывода, которые представлены в виде специальных файлов и т. д.), к которым процесс может обратиться с указанием возможного типа доступа (чтение, запись, исполнение).

Чтобы узнать, под каким пользователем Вы работаете, служит команда *whoami*.

Для определения, в каких группах состоит пользователь, необходимо воспользоваться командой *groups*. Если Вы хотите посмотреть, в каких группах состоит другой пользователь, нужно передать его имя в качестве аргумента: *groups root*.

Для изменения прав доступа к файлам/каталогам используется команда *chmod [-R] права файл_или_каталог [файл2 ...].*

Необязательный ключ **-***R* распространяет действие команды рекурсивно на содержимое каталогов, если таковые обнаружатся в списке файлов, переданном в командной строке.

Права указываются в одной из двух нотаций: числовой и символьной.

Числовая нотация команды *chmod*: набор прав разбивается на четыре тройки и рассматривается в виде битового поля – бит установлен, если соответствующее право имеется. Каждая тройка бит записывается десятичным числом.

Дополнительные флаги доступа: *sticky bit (t)*, специфичный для директорий, и *sticky bit (s)*, применяемый для исполняемых файлов.

Сегодня *sticky bit* используется в основном для директорий, чтобы защитить в них файлы. Из такой директории пользователь может удалить только те файлы, владельцем которых он является. Примером может служить директория */tmp*, в которой запись открыта для всех пользователей, но нежелательно удаление чужих файлов. Программа с установленным битом suid является «потенциально опасной». Если установлены права доступа *SUID* и файл исполняемый, то при запуске на выполнение файл получает не права запустившего его, а права владельца файла.

Примеры.

1 Добавить группе право на запись: *chmod g+w file*.

2 Убрать у прочих права на запись и исполнение: *chmod o-wx file*.

3 Установить права прочих и группы такими же, как у владельца: *chmod og=u file*.

4 Несколько изменений можно перечислять через запятую. Например, добавить владельцу право на исполнение, а у группы и прочих убрать право на запись: *chmod u+x, go-w file*.

5 Для изменения владельца или группы владельца файла (или другого объекта) используются команды *chown* или *chgrp* соответственно. Сначала нужно передать имя группы или владельца, а потом список файлов. Например, для пользователя user1:

chown user1 /home/user1/itmo.txt

chgrp user1 /home/user1/itmo.txt

Следует отметить, что нельзя использовать команду *chown* без прав суперпользователя, но команда *chgrp* может быть использована всеми, чтобы изменить группу-владельца файла на ту группу, к которой они принадлежат.

Когда в *Linux* создается новый файл, система обращается к параметру, называемому *umask*. Значением по умолчанию для *umask* является 0022, что позволяет другим читать Ваши новые файлы, но не изменять их. Чтобы автоматически обеспечивать больший уровень защищенности для создаваемых файлов, можно изменить настройки *umask*, например:

User1@ubuntu:~$ umask 0077

В отличие от «обычного» назначения прав доступа к файлу, *umask* указывает, какие права доступа должны быть отключены. Следовательно, в приведенном примере все права для группы и остальных пользователей будут отключены, а права владельца останутся неизменными.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить задание.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Задание

1 Установите ОС семейства *Linux* на виртуальной машине. При выполнении заданий протоколируйте выполняемые команды.

2 Откройте два терминала (в серверных *Linux* для переключения между терминалами (*tty*) обычно используется сочетание клавиш Alt + F[1–5]). В одном из них получите права суперпользователя, используя команду *sudo su*.

3 Изучите команду создания пользователя с домашним каталогом с помощью команд *useradd* и *adduser* из справочной документации *man*.

4 Используя *useradd* или *adduser*, создайте пользователей *user1* и *user2* с домашними каталогами *user1* и *user2* соответственно.

5 Установите пароли для пользователей *user1* и *user2* с помощью команды *passwd*.

6 Выйдите из суперпользователя командой *exit*.

7 Войдите под первым терминалом в пользователя *user1*, под вторым –

в пользователя *user2*.

8 Посмотрите, какой идентификатор получил пользователь *user1* и пользователь *user2*, используя команду *id*.

9 Посмотрите права доступа на домашний каталог пользователей *user* и *user2*, используя команду *ls*.

10 Создайте файл test под пользователем *user*2 с маской 0077, используя команду *umask*.

11 Попробуйте прочитать содержимое файла *test* под пользователем *user1*, используя команду *cat*.

12 Измените права доступа на файл *test* так, чтобы пользователь *user1* мог записывать в файл, но не читать его.

13 Запишите текстовую информацию в файл в роли *user1*, используя команду *ls -l - test2*.

14 Проверьте права на файл *test2* и прочитайте его содержимое из-под пользователя *user*2.

15 Создайте каталог из-под пользователя *user2*.

16 Установите права записи для группы пользователей на данный каталог.

17 Добавьте пользователя *user1* в группу *user2* с помощью команды *usermod*.

18 Проверьте, в какие группы входит пользователь *user1*.

19 Создайте несколько файлов в каталоге, который был создан пользователем *user2* из-под пользователя *user*.

20 Изучите в справочной документации *man*, как удалить пользователя вместе с содержимым его домашнего каталога.

21 Удалите пользователя *user2* вместе с его домашним каталогом.

Содержание отчета

1 Тема и цель работы.

2 Краткий конспект теоретических сведений.

3 Ход выполнения задания с результатами.

4 Выводы по работе.

Контрольные вопросы

1 Какой *uid* у пользователя *user2*? В какие группы он входит?

2 Почему попытка удалить пользователя не удалась и что нужно сделать для его удаления?

3 Какие права доступа установлены на домашний каталог пользователя *user*?

4 Как рекурсивно изменить права доступа на файлы в каталоге?

5 Как можно осуществлять переключение между пользователями в рамках одного терминала?

6 Как удалить пользователя, при этом сохранив его домашний каталог и данные внутри него?

7 Какое значение *umask* нужно установить, чтобы владелец и группа имели право на чтение, запись и исполнение, а все остальные пользователи не имели никаких прав?

4 Лабораторная работа № 4. Возможности файловых подсистем ОС для защиты информации

Цель работы: приобрести практические навыки работы с таблицами разделов (MBR и GPT), резервирования и восстановления данных таблиц разделов.

Теоретические сведения

На данный момент наиболее распространенной схемой разбиения дисков является MBR. Но с развитием средств хранения данных и их объемов возможностей MBR становится недостаточно. Это связано с невозможностью обеспечивать доступ к разделу диска емкостью более чем 2,2 TB. На сегодняшний день уже доступны диски емкостью 10 и более TB, а также применяются различные технологии по объединению дисков в массивы, такие как RAID и LVM. Таким образом, использование схемы разбиения дисков на основе GPT становится все более актуальным для современных операционных систем.

Процесс загрузки компьютера является многоступенчатым процессом. Начинается он с инициализации системных устройств набором микропрограмм, называемых BIOS (Basic Input/Output System), которые выполняются при старте системы. После того как BIOS успешно проверит системные устройства и установленное оборудование, идет процесс поиска загрузчика в MBR устройств хранения (CD/DVD-диски, USB-диск, HDD, SSD и др.) или на первом разделе устройства. Как именно диск делится на разделы, определяется таблицей разделов. Каждая запись таблицы разделов описывает один из разделов жесткого диска. Таблицы разделов бывают двух типов: MBR и GPT.

После того как загрузчик получил управление, он получает таблицу разделов и готовит к загрузке операционную систему. В семействе загрузчиков GNU/Linux яркими представителями являются GRUB и LILO. В них MBR состоит из небольшой части ассемблерного кода. Стандартный загрузчик Windows/MS DOS в состоянии проверить только активный раздел, считать несколько секторов с этого раздела и затем передать управление операционной системе. Он не в состоянии загрузить Linux, поскольку не наделен необходимым функционалом. GRand Unified Bootloader (GRUB) – это стандартный загрузчик для операционных систем семейства GNU/Linux, и всем пользователям рекомендуется по умолчанию установить его в MBR для того, чтобы иметь возможность загружать операционную систему с любого раздела, первичного или логического.

Структура MBR.

Структура MBR представлена на рисунке 4.1. Первые 512 байт (чаще всего первый сектор диска) главного устройства хранения данных занимает MBR (Master Boot Record). В состав MBR входит 446 байт кода загрузчика, четыре записи по 16 байт – это таблица разделов, 2 байта сигнатуры (55h AAh). Таблица разделов может состоять из первичных разделов (до четырёх) и логических разделов (до 128).

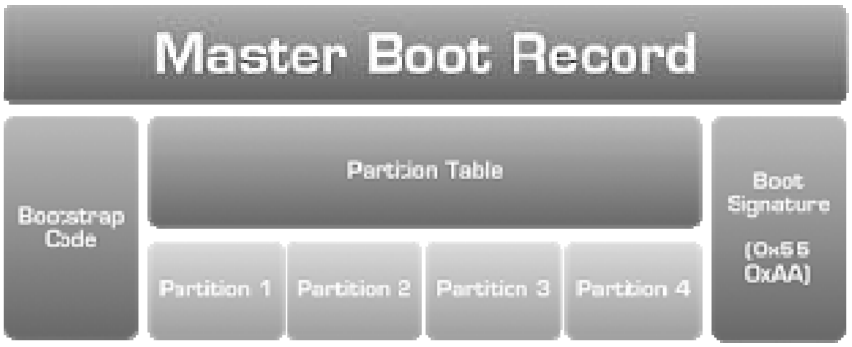

Рисунок 4.1 – Структура MBR

Структура GPT.

GUID Partition Table (GPT) – стандарт формата размещения таблиц разделов на физическом жестком диске (рисунок 4.2).

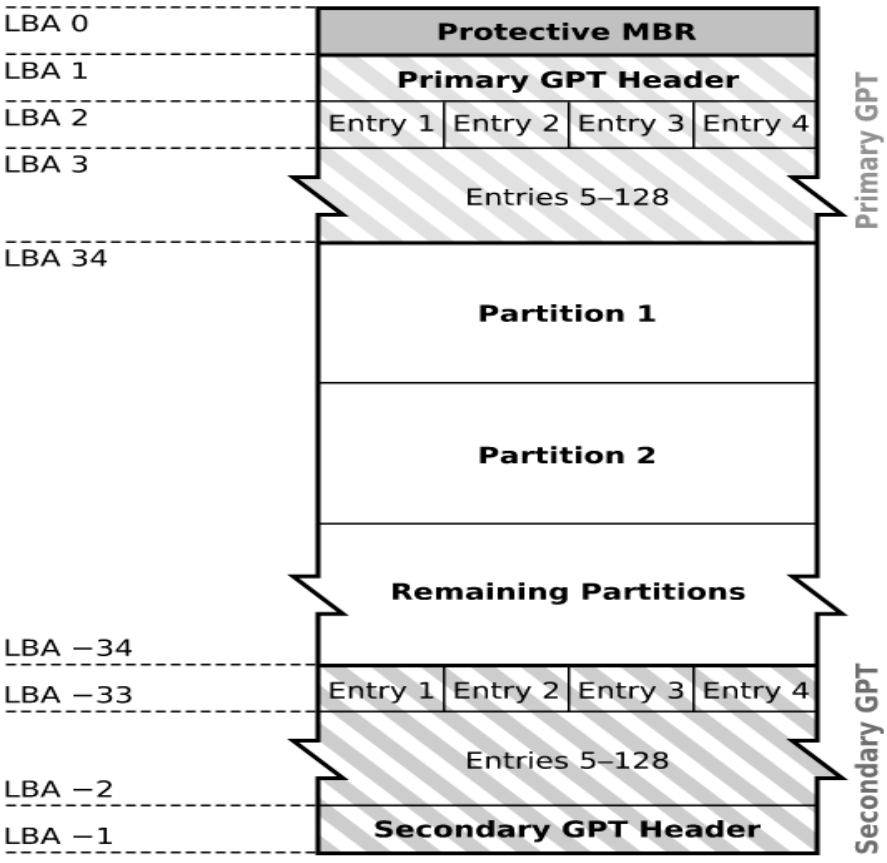

Рисунок 4.2 – Структура GPT

Он является частью расширяемого микропрограммного интерфейса (англ. Extensible Firmware Interface (EFI)) – стандарта, предложенного Intel на смену BIOS. EFI UEFI (англ. Unified Extensible Firmware Interface) использует GPT там, где BIOS использует главную загрузочную запись (англ. Master Boot Record (MBR)).

В GPT нет собственной программы-загрузчика, вместо этого он работает в паре с EFI UEFI. UEFI (Unified Extensible Firmware Interface) – унифицированный расширяемый интерфейс прошивки – является более продвинутым интерфейсом, чем BIOS. Внутри GPT используется адресация логических блоков LBA, которая абстрагирована от физики устройства (в отличие от CHS «Цилиндр – Головка – Сектор»).

Каждый логический блок занимает 512 байт. LBA 0 – первые 512 байт диска (Protective MBR), LBA 1 (Primary GPT Header) – следующие и т. д. Отрицательные значения LBA означают смещение в блоках с конца диска. Последний блок имеет смещение «–1» (LBA –1 – Secondary GPT Header). GPT поддерживается только современными операционными системами, т. к. он ещё относительно молод.

Пример работы с MBR.

Существует специальный набор команд для работы с MBR, позволяющий сохранить резервную копию MBR и в случае необходимости восстановить.

Команды создания резервной копии MBR:

dd if=/dev/sda of=/path/mbr-backup bs=512 count=1

или

dd if=/dev/sda of=/media/Main/Backups/sda-mbr.bin bs=512 count=1

Команды для восстановления MBR:

dd if=/path/mbr-backup of=/dev/sda bs=512 count=1

или

dd if=/media/Main/Backups/sda-mbr of=/dev/sda

Команда для сохранения только загрузочного кода:

dd if=/dev/sda of=/path/mbr-boot-code bs=446 count=1

Команда для сохранения только таблицы разделов:

dd if=/dev/sda of=/path/mbr-part-table bs=1 count=66 skip=446

Команда для восстановления загрузочного кода из файла mbr-backup:

dd if=/path/mbr-backup of=/dev/sda bs=446 count=1

Команда для восстановления только таблицы разделов:

dd if=/path/mbr-backup of=/dev/sda bs=1 skip=446 seek=466 count=66

Команда для очистки MBR, но при этом оставить таблицу разделов (требуется вход от лица учетной записи *root*):

dd if=/dev/zero of=/dev/sda bs=446 count=1

Перечень команд, необходимых для выполнения работы:

– *fdisk* <параметры> – консольная программа для управления разделами жесткого диска дисками (работает только с MBR);

– *parted* <параметры> – консольная программа для управления дисками создания, изменения размера и восстановления разделов диска (работает как с

MBR, так и с GPT);

– *dd* <параметры> – консольная программа байтового копирования данных;

– *mkfs.*<тип файловой системы> <форматируемый раздел диска> – класс консольных команд создания файловых систем на разделах файловой системы Linux на некотором устройстве, как правило, в разделе жёсткого диска;

– *mount-t* <тип файловой системы> <раздел диска> <точка монтирования> – консольная программа монтирования разделов жесткого диска с файловой системой NTFS или ext2, или ext3.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить практическое задание.

Задание

1 Добавьте в виртуальную машину с операционной системой *Linux* виртуальный жесткий диск (делается это в настройках виртуальной машины).

2 Запустите виртуальную машину с операционной системой *Linux*.

3 Ознакомьтесь с командой *fdisk* и ее возможностями из справочной документации.

4 Создайте таблицу разделов (три первичных и один логический) с помощью команды *fdisk* на добавленном виртуальном диске (обычно это диск */dev/sdb*).

5 Запишите изменения на диск.

6 Проверьте факт создания разделов, используя команду *fdisk* (создание разделов можно также проверить, используя команду *ls /dev/sd**).

7 Отформатируйте созданные разделы в файловую систему *ext4*.

8 Ознакомьтесь с командами *mount* и *umount* и их возможностями из справочной документации.

9 Смонтируйте созданные разделы и создайте там произвольные файлы.

10 Сделайте резервную копию MBR с помощью утилиты DD.

11 Сотрите таблицу разделов MBR с помощью утилиты DD.

12 Восстановите MBR с помощью утилиты DD.

13 Смонтируйте разделы и проверьте целостность данных.

14 Отмонтируйте разделы.

15 Установите *gdisk <sudo apt-get install gdisk>*.

16 Создайте таблицу разделов GPT (пять первичных разделов) с помощью команды *gdisk*.

17 Отформатируйте созданные разделы в файловую систему *ext3*.

18 Смонтируйте созданные разделы и создайте там произвольные файлы.

19 Сделайте резервную копию GPT с помощью утилиты DD, предварительно определив необходимое количество байт для резервной копии.

20 Сотрите GPT с помощью утилиты DD.

21 Восстановите GPT с помощью утилиты DD.

22 Смонтируйте разделы и проверьте целостность данных.
23 Отмонтируйте разделы.

Содержание отчета

1 Тема и цель работы.
2 Краткий конспект теоретических сведений.
3 Ход выполнения задания с результатами.
4 Выводы по работе.

Контрольные вопросы

1 Что записано в первом секторе главной загрузочной записи MBR?
2 Функциональное назначение MBR и GPT.
3 Структура GPT.
4 Какое максимальное количество первичных разделов можно создать при использовании таблицы разделов MBR?
5 Какое максимальное количество первичных разделов можно создать при использовании таблицы разделов GPT?
6 Как сохранить информацию о структуре MBR?
7 Как создать 10 разделов с файловой системой ext3 на диске в таблице разделов MBR?
8 Как стереть код загрузчика в MBR?
9 Как можно смонтировать раздел диска с файловой системой в режиме только для чтения?
10 Как можно осуществить восстановление GPT разделов в случае сбоев?

5 Лабораторная работа № 5. Обеспечение целостности и доступности данных с использованием Raid

Цель работы: получить практические навыки построения и управления RAID массивами и логическими томами.

Теоретические сведения

RAID (Redundant Array of Independent Disks – избыточный массив независимых жестких дисков) – массив, состоящий из нескольких дисков, управляемых программным или аппаратным контроллером, связанных между собой и воспринимаемых как единое целое. В зависимости от того, какой тип массива используется, может обеспечивать различные степени быстродействия и отказоустойчивости. Служит для повышения надежности хранения данных и/или для повышения скорости чтения/записи информации.

Калифорнийский университет в Беркли предложил следующие уровни спецификации RAID, которые являются стандартом во всем мире:

– RAID 0 представлен как дисковый массив повышенной производительности, без отказоустойчивости (требуется минимум два диска);

– RAID 1 определен как зеркальный дисковый массив (требуется минимум два диска);

– RAID 2 массивы, в которых применяется код Хемминга (требуется минимум семь дисков для рационального использования);

– RAID 3 и 4 используют массив дисков с чередованием и выделенным диском четности (требуется минимум четыре диска);

– RAID 5 используют массив дисков с чередованием и «невыделенным диском четности» (требуется минимум три диска);

– RAID 6 используют массив дисков с чередованием и двумя независимыми «четностями» блоков (требуется минимум четыре диска);

– RAID 10 – RAID 0, построенный из RAID 1 массивов (требуется минимум четыре диска, четное количество);

– RAID 50 – RAID 0, построенный из RAID 5 массивов (требуется минимум шесть дисков, четное количество);

– RAID 60 – RAID 0, построенный из RAID 6 массивов (требуется минимум восемь дисков, четное количество).

Основные сведения об утилите LVM.

LVM (Logical Volume Manager) – менеджер логических томов – является уникальной системой управления дисковым пространством. Она позволяет с легкостью использовать и эффективно управлять дисковым пространством. Уменьшает общую нагруженность и сложность существующей системы. У логических томов, которые созданы через LVM, можно легко изменять размер, а названия, которые им даны, помогут в дальнейшем определить назначение тома.

Работа с томами при помощи LVM организована на трёх уровнях:

1) PV, Physical Volume или физический том. Чаще всего это раздел на диске или весь диск. К ним относят устройства программного и аппаратного RAID массивов (которые могут включать в себя еще несколько физических дисков). Физические тома объединяются и образуют группы томов;

2) VG, Volume Group или группа томов. Это самый верхний уровень модели представления, которая используется в LVM. С одной стороны, группа томов может состоять из физических томов, с другой – из логических томов и представлять собой единую структуру;

3) LV, Logical Volume или логический том. Раздел в группе томов – тоже самое, что раздел диска в не-LVM-системе. Является блочным устройством и, как следствие, может содержать файловую систему.

Физические и логические тома, в свою очередь, делятся на фрагменты (экстенты):

– PE, Physical Extent, или физический экстент. Каждый физический том делится на блоки данных – физические экстенты. Они имеют размеры, как и у логических экстентов;

– LE, Logical Extent, или логический экстент. Каждый логический том также делится на блоки данных – логические экстенты. Размеры логических экстентов

не меняются в рамках группы томов.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить практическое задание.

Задание

1 Добавьте пять виртуальных жестких дисков.
2 Запустите *Linux*.
3 Установите *mdadm*.
4 Ознакомьтесь с утилитой *mdadm*, ее возможностями и параметрами.
5 В отдельном терминале отследите за состоянием файла */proc/mdstat*.
6 Соберите RAID 1 с помощью *mdadm*.
7 Создайте на созданном RAID файловую систему *ext4*.
8 Смонтируйте созданную файловую систему.
9 Запишите туда файл *raid.txt* с произвольным содержимым.
10 Разрушьте один из дисков RAID и проследите за происходящим в файле */proc/mdstat*.
11 Проверьте целостность файла *raid.txt*.
12 Остановите RAID 1.
13 Очистите информацию дисков о принадлежности к программному RAID.
14 Соберите RAID 0 с помощью *mdadm*.
15 Создайте на созданном RAID файловую систему *ext3*.
16 Смонтируйте созданную файловую систему.
17 Запишите туда файл *raid.txt* с произвольным содержимым.
18 Разрушьте один из дисков RAID и проследите за происходящим в файле */proc/mdstat*.
19 Проверьте целостность файла *raid.txt*.
20 Остановите RAID 0.
21 Очистите информацию дисков о принадлежности к программному RAID.
22 Инициализуйте физические диски, поверх которых будет создан LVM.
23 Создайте группу томов на основе четырех виртуальных жестких дисков.
24 Создайте логический том.
25 На созданном логическом томе создайте файловую систему.
26 Смонтируйте систему и создать файл файл *LVM.txt* .
27 Добавьте в группу томов еще один виртуальный жесткий диск.
28 Определите количество добавленных экстентов.
29 Расширьте созданный логический том на размер добавленных экстентов.
30 Увеличьте размер файловой системы.
31 Сделайте снапшот логического тома.
32 Удалите группу томов и снапшот.

Содержание отчета

1 Тема и цель работы.
2 Краткий конспект теоретических сведений.
3 Ход выполнения задания с результатами.
4 Выводы по работе.

Контрольные вопросы

1 В чем достоинства и недостатки различных уровней RAID?
2 Что такое диск горячей замены RAID?
3 Как осуществить инициализацию физических дисков для использования их в качестве RAID массива?
4 Сколько минимально необходимо дисков для различных уровней RAID?
5 Сколько максимально может выйти из строя дисков в различных уровнях RAID массивов без потери данных?
6 Порядок действий для создания логического тома LVM.
7 Что такое экстенты в LVM? Как создать логический том с определенным количеством экстентов?
8 Что такое логический том? Что такое физический том? В чем между ними отличие?
9 Как узнать количество экстентов в группе томов?

6 Лабораторная работа № 6. Изучение методов шифрования ОС Windows данных на дисках

Цель работы: изучить методы шифрования ОС Windows.

Теоретические сведения

Шифрование файлов.

Encrypting File System (EFS) – система шифрования данных, реализующая шифрование на уровне файлов в операционных системах Microsoft Windows NT (начиная с Windows 2000 и выше), за исключением «домашних», Windows 10 Pro, Enterprise, and Education editions, Windows Server 2016, Windows Server 2019. Данная система предоставляет возможность «прозрачного шифрования» данных, хранящихся на разделах с файловой системой NTFS, для защиты потенциально конфиденциальных данных от несанкционированного доступа при физическом доступе к компьютеру и дискам.

Аутентификация пользователя и права доступа к ресурсам, имеющие место в NT, работают, когда операционная система загружена, но при физическом доступе к системе возможно загрузить другую ОС, чтобы обойти эти ограничения.

EFS использует симметричное шифрование для защиты файлов, а также шифрование, основанное на паре открытый/закрытый ключ для защиты случайно сгенерированного ключа шифрования для каждого файла. По умолчанию закрытый ключ пользователя защищён с помощью шифрования пользовательским паролем, и защищённость данных зависит от стойкости пароля пользователя.

EFS работает, шифруя каждый файл с помощью алгоритма симметричного шифрования, зависящего от версии операционной системы и настроек (доступна теоретическая возможность использования сторонних библиотек для шифрования данных). При этом используется случайно сгенерированный ключ для каждого файла, называемый File Encryption Key (FEK). Выбор симметричного шифрования на данном этапе объясняется его скоростью по отношению к асимметричному шифрованию.

FEK (случайный для каждого файла ключ симметричного шифрования) защищается путём асимметричного шифрования, использующего открытый ключ пользователя, шифрующего файл, и алгоритм RSA (теоретически возможно использование других алгоритмов асимметричного шифрования). Зашифрованный таким образом ключ FEK сохраняется в альтернативном потоке *$EFS* файловой системы NTFS. Для расшифрования данных драйвер шифрованной файловой системы прозрачно для пользователя расшифровывает FEK, используя закрытый ключ пользователя, а затем и необходимый файл с помощью расшифрованного файлового ключа.

При этом для обеспечения секретности самого ключа FEK шифруется асимметричным алгоритмом RSA открытым ключом пользователя, результат шифрации FEK – Data Decryption Field (DDF) – добавляется в заголовок зашифрованного файла. Для расшифровки файлов в EFS в случае утраты закрытого ключа EFS используется агент восстановления. При каждом шифровании файла ключ FEK шифруется с помощью открытого ключа агента восстановления. Этот ключ FEK добавляется в файл вместе с копией ключа, зашифрованной с помощью открытого ключа EFS в «поле восстановления данных» (Data Recovery Field (DRF)). Закрытый ключ агента восстановления позволяет расшифровать ключ FEK, а следовательно, и сам файл. Такой подход обеспечивает надежное шифрование без потери эффективности процесса шифрования: данные шифруются быстрым симметричным алгоритмом, а для гарантии секретности симметричного ключа используется асимметричный алгоритм шифрования.

Поскольку шифрование/расшифрование файлов происходит с помощью драйвера файловой системы (по сути, надстройки над NTFS), оно происходит прозрачно для пользователя и приложений. Следует отметить, что EFS не шифрует файлы, передаваемые по сети, поэтому для защиты передаваемых данных необходимо использовать другие протоколы защиты данных (IPSec или WebDAV).

Для работы с EFS у пользователя есть возможность использовать графический интерфейс проводника или утилиту командной строки.

Графический интерфейс доступен в стандартном окне свойств объекта по нажатию кнопки «Дополнительно», при этом для файлов можно добавить открытые ключи других пользователей, которые тоже будут иметь возможность рас-

шифровать данный файл и работать с его содержимым (при наличии соответствующих разрешений). При шифровании папки шифруются все файлы, находящиеся в ней, а также те, которые будут помещены в неё позднее.

Необходимо отметить, что EFS позволяет разделять зашифрованный файл между несколькими пользователями. В этом случае FEK шифруется открытыми ключами всех пользователей, которым разрешен доступ к файлу, и каждый результат шифрации добавляется в DDF.

При работе с проводником Windows возможно отображение зашифрованных папок и файлов другим (по умолчанию зелёным) цветом, позволяющим визуально отличить защищённое таким образом содержимое. При копировании зашифрованных файлов на раздел, где шифрование не поддерживается (например, с файловой системой FAT32 и т. д.), будет выдано предупреждение о том, что файл будет расшифрован.

Пара открытый и закрытый ключи для шифрации FEK создается для пользователя автоматически при первой шифрации файла с использованием EFS.

Консольная команда cipher.

Консольная команда *cipher* может быть использована для шифрации/дешифрации файлов из командной строки или сценария. Может быть использована как в командной строке *cmd*, так и в PowerShell.

Назначения параметров команды приведены в таблице 6.1.

Таблица 6.1 – Параметры команды *cipher*

Параметр	Назначение
/B	Прекращение выполнения при возникновении ошибки. По умолчанию *cipher* продолжает выполнение даже в случае возникновения ошибок
/C	Отображение сведений о зашифрованном файле
/D	Расшифровка указанных файлов или каталогов
/E	Шифрование указанных файлов или каталогов. Каталоги будут помечены таким образом, что добавляемые впоследствии файлы тоже будут шифроваться. Зашифрованный файл может быть расшифрован, если он изменен и при этом родительский каталог не зашифрован. Рекомендуется одновременно зашифровывать файл и родительский каталог
/H	Отображение скрытых и системных файлов. По умолчанию такие файлы пропускаются
/N	Работает только с /U. Запрещает обновление ключей. Используется для поиска всех зашифрованных файлов на локальных дисках
/S	Выполнение выбранной операции для указанного каталога и всех файлов и вложенных каталогов
/U	Попытка обращения ко всем зашифрованным файлам на локальных дисках. Это приводит к обновлению пользовательского ключа шифрования файлов или ключей восстановления до текущих ключей, если они изменены. Может использоваться только с параметром /N
/W	Удаление всех данных в неиспользуемом дисковом пространстве на всем томе. Если задан этот параметр, все другие параметры пропускаются. Указанный каталог может находиться в любом месте локального тома

При удалении файла или папки не происходит полного физического удаления информации, очищается лишь только «оглавление» файловой системы. С помощью утилиты *cipher* возможно частичное решение этой проблемы. Для этого необходимо использовать синтаксис:

cipher / W <путь к любой папке на разделе, подлежащем очистке>.

Защита файлов с помощью шифрования дисков BitLocker.

Шифрование дисков BitLocker используется для защиты всех файлов, хранящихся на диске с установленной ОС Windows (диск операционной системы) и на несъемных дисках (например, внутренних жестких дисках). Шифрование BitLocker To Go используется для защиты всех файлов, хранящихся на съемных дисках (например, внешних жестких дисках или USB-устройствах флеш-памяти). BitLocker доступен только в изданиях Professional и Enterprise.

Наиболее надежную защиту обеспечивает использование BitLocker совместно с TPM. Модуль TPM (Trusted Platform Module, доверенный платформенный модуль) – это специальный криптопроцессор, в котором хранятся криптографические ключи для защиты информации, микрочип, установленный на материнской плате компьютера. При загрузке компьютера BitLocker запрашивает у доверенного платформенного модуля ключи для доступа к диску и разблокирует его.

Основная задача TPM – это создание безопасного компьютера, в котором проверяются и защищаются все процессы связи, а также аппаратное и программное обеспечение.

В архитектуре чипа реализованы следующие защитные механизмы:

– защищённое управление памятью;
– шифрование шины и данных;
– тестирование режимов блокирования;
– активное экранирование.

Проверить, имеется ли на компьютере TMP, можно, выполнив команду *Панель управления – Шифрование диска BitLocker – Администрирование доверенного платформенного модуля*.

При отсутствии TMP технология BitLocker шифрует жесткий диск, но не обеспечивает защиту загрузочной среды. При загрузке потребуется установка USB-носителя.

Для запуска BitLocker из графического интерфейса служит команда *Панель управления / Все элементы панели управления / Шифрование диска BitLocker*.

В отличие от шифрованной файловой системы (EFS), позволяющей зашифровывать отдельные файлы, BitLocker шифрует диск целиком. Пользователь может входить в систему и работать с файлами как обычно, а BitLocker будет мешать злоумышленникам, пытающимся получить доступ к системным файлам для поиска паролей, а также к диску путем извлечения его из данного компьютера и установки в другой.

При шифровании диска с ОС BitLocker проверяет компьютер при загрузке на наличие возможных угроз безопасности (например, изменений в BIOS или файлах загрузки). При обнаружении угрозы безопасности BitLocker заблокирует

диск с ОС. Чтобы разблокировать его, потребуется специальный ключ восстановления BitLocker. Важно убедиться, что этот ключ создан при первом запуске BitLocker. В противном случае доступ к файлам может быть потерян.

BitLocker автоматически шифрует все файлы, добавляемые на зашифрованный диск. Файлы будут зашифрованы только при хранении на зашифрованном диске. При их копировании на другой диск или компьютер они будут расшифрованы. При предоставлении общего доступа к файлам по сети они будут зашифрованы на зашифрованном диске, но авторизованные пользователи смогут получать к ним доступ обычным образом.

Другой способ управления BitLocker – утилита *manage-bde.exe.*

Утилита manage-bde.exe.

Утилита настраивает шифрование диска BitLocker для томов диска и может быть запущена как в командной строке *cmd*, так и в PowerShell:

manage-bde[.exe] -параметр [аргументы]

Список параметров утилиты *manage-bde.exe* приведен в таблице 6.2.

Таблица 6.2 – Параметры утилиты *manage-bde*

Параметр	Назначение
-status	Предоставляет сведения о томах, поддерживающих шифрование BitLocker
-on	Зашифровывает том и включает защиту BitLocker
-off	Расшифровывает том и отключает защиту BitLocker
-pause	Приостанавливает шифрование или расшифровку
-resume	Возобновляет шифрование или расшифровку
-lock	Запрещает доступ к данным, зашифрованным BitLocker
-unlock	Разрешает доступ к данным, зашифрованным BitLocker
-autounlock	Управляет автоматическим снятием блокировки с томов данных
-protectors	Управляет методами защиты для ключа шифрования
-tpm	Настраивает доверенный платформенный модуль компьютера
-SetIdentifier или -si	Настраивает поле идентификации для тома
-ForceRecovery или -fr	При перезагрузке принудительно восстанавливает операционную систему, защищенную BitLocker
-changepassphrase	Изменяет пароль для тома с данными
-changepin	Изменяет ПИН-код для тома
-changekey	Изменяет ключ запуска для тома
-upgrade	Обновляет версию BitLocker
-ComputerName или -cn	Выполняет команду на другом компьютере
-? или /?	Отображает краткую справку. Пример: "-ParameterSet -?"
-Help или -h	Отображает полную справку. Пример: "-ParameterSet -h"

Пример – Для вывода информации, полученной с помощью утилиты manage-bde в текстовый файл, можно воспользоваться командами следующего вида:

manage-bde –status -h > file.txt (при работе в cmd) или
manage-bde –on –h | out-file file1.txt (при работе в PowerShell).

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить практическое задание.

Задание

1 Запустите ОС Windows версий 7/8/10 на виртуальной машине.

2 Выберите папку для шифрования. Папка должна называться по фамилии студента. Выполните шифрование файлов в выбранной папке. Сохраните сертификат в отдельном файле, чтобы иметь в дальнейшем возможность дешифрации файла при любых условиях. Убедитесь, что после этого был создан сертификат пользователя, запустив оснастку *certmgr.msc* от имени пользователя (раздел Личное). Просмотрите и добавьте в отчет основные параметры сертификата открытого ключа пользователя (срок действия, используемые алгоритмы).

3 Выполните шифрование и расшифрование файлов с помощью команды cipher группы файлов с использованием шаблона имени или расширения файлов.

4 Создайте на виртуальной машине новый раздел диска, отформатированный под NTFS. Скопируйте на новый раздел папку с файлами. Установите защиту созданного раздела с помощью BitLocker. Архивацию ключа восстановления выполните в файл, который должен находиться не на шифруемом диске.

5 Убедитесь, что доступ к диску без пароля невозможен.

6 С помощью утилиты *manage-bde.exe* получите сведения о имеющихся томах системы, их состоянии, перенаправив информацию в файл.

7 Выполните отключение BitLocker с помощью утилиты *manage-bde.exe*, убедитесь, что информация на диске доступна.

Содержание отчета

1 Тема и цель работы.
2 Краткий конспект теоретических сведений.
3 Ход выполнения задания с результатами.
4 Выводы по работе.

Контрольные вопросы

1 Каким образом реализуется шифрование файлов в NTFS?
2 Каким образом шифруются файлы в файловой системе EFS?
3 Какие алгоритмы шифрования используются в EFS?
4 Каким образом реализуется шифрование дисков?
5 В чем отличие шифрование с помощью *cipher* и *manage-bde*?
6 Позволяет ли шифрование файлов обеспечить их защиту при передаче по сети?

7 Что произойдет с защифрованным файлом при копировании на флеш-память с файловой системой FAT32?

7 Лабораторная работа № 7. Средства защиты данных в ОС Windows

Цель работы: изучить средства защиты данных в ОС Windows.

Теоретические сведения

Классификация защиты семейства ОС Windows.

Защита конфиденциальных данных от несанкционированного доступа является важнейшим фактором успешного функционирования любой многопользовательской системы. ОС Windows не является исключением, и требования к защите объектов файловой системы, памяти, объектов ядра операционной системы внесли существенный вклад в процесс ее проектирования и реализации.

Так, например, версии Windows NT/2000/7/8/10 были сертифицированы по классу С2 критериев TSSEC («Оранжевая книга»). Требования к операционной системе, защищенной по классу С2, включают:

– обязательную идентификацию и аутентификацию всех пользователей операционной системы. Под этим понимается способность операционной системы идентифицировать всех пользователей, которые получают санкционированный доступ к системе, и предоставление доступа к ресурсам только этим пользователям;

– разграничительный контроль доступа – предоставление пользователям возможности защиты принадлежащих им данных;

– системный аудит – способность системы вести подробный аудит всех действий, выполняемых пользователями и самой операционной системой;

– защита объектов от повторного использования – способность системы предотвратить доступ пользователя к информации ресурсов, с которыми до этого работал другой пользователь.

Идентификация пользователей.

Для защиты данных Windows использует следующие основные механизмы: аутентификация и авторизация пользователей, аудит событий в системе, шифрование данных, поддержка инфраструктуры открытых ключей, встроенные средства сетевой защиты. Эти механизмы поддерживаются такими подсистемами Windows, как LSASS (Local Security Authority Subsystem Service, подсистема локальной аутентификации), SAM (Security Account Manager, диспетчер локальных записей безопасности), SRM (Security reference Monitor, монитор состояния защиты), Active Directory (служба каталогов), EFS (Encrypting File System, шифрующая файловая система) и др.

Защита объектов и аудит действий с ними в ОС Windows организованы на основе избирательного (дискреционного) доступа, когда права доступа (чтение,

запись, удаление, изменение атрибутов) субъекта к объекту задается явно в специальной матрице доступа. Для укрупнения матрицы пользователи могут объединяться в группы. При попытке субъекта (одного из потоков процесса, запущенного от его имени) получить доступ к объекту указывается, какие операции пользователь собирается выполнять с объектом. Если подобный тип доступа разрешен, поток получает описатель (дескриптор) объекта и все потоки процесса могут выполнять операции с ним. Рассмотрим, как в ОС Windows организована аутентификация и авторизация пользователей.

Все действующие в системе объекты (пользователи, группы, локальные компьютеры, домены) идентифицируются в Windows не по именам, уникальность которых не всегда удается достичь, а по идентификаторам защиты (Security Identifiers, SID). SID представляет собой числовое значение переменной длины.

При генерации SID Windows использует генератор случайных чисел, чтобы обеспечить уникальность SID для каждого пользователя. Для некоторого произвольного пользователя SID может выглядеть так:

S-1-5-21-789336058-484763869-725345543-1003.

Предопределенным пользователям и группам Windows выдает характерные SID, состоящие из SID компьютера или домена и предопределенного RID.

Полный список общеизвестных SID можно посмотреть в документации Platform SDK. Узнать SID конкретного пользователя в системе, а также SID групп, в которые он включен, можно, используя консольную команду *whoami*:

whoami /user

Соответствие имени пользователя и его SID можно отследить также в ключе реестра *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ ProfileList*.

После аутентификации пользователя процессом Winlogon все процессы, запущенные от имени этого пользователя, будут идентифицироваться специальным объектом, называемым маркером доступа (access token). Если процесс пользователя запускает дочерний процесс, то его маркер наследуется.

Маркер доступа содержит идентификатор доступа самого пользователя и всех групп, в которые он включен. В маркер включен также DACL по умолчанию – первоначальный список прав доступа, который присоединяется к создаваемым пользователем объектам. Еще одна важная для определения прав пользователя в системе часть маркера – список его привилегий. Привилегии – это права доверенного объекта на совершение каких-либо действий по отношению ко всей системе. В таблице 7.1 перечислены некоторые привилегии, которые могут быть предоставлены пользователю.

Управление привилегиями пользователей осуществляется в оснастке «Групповая политика», раздел Конфигурация Windows/Локальные политики/Назначение прав пользователя.

Чтобы посмотреть привилегии пользователя, можно также использовать команду *whoami /all*.

Остальные параметры маркера носят информационный характер и определяют, например, какая подсистема создала маркер, уникальный идентификатор маркера, время его действия.

Таблица 7.1 – Привилегии, которыми могут быть наделены пользователи

Имя и идентификатор привилегии	Описание привилегии
Увеличение приоритета процесса SeIncreaseBasePriorityPrivilege	Пользователь, обладающий данной привилегией, может изменять приоритет процесса с помощью интерфейса Диспетчера задач
Закрепление страниц в памяти SeLockMemoryPrivilege	Процесс получает возможность хранить данные в физической памяти, не прибегая к кэшированию данных в виртуальной памяти на диске
Управление аудитом и журналом безопасности SeAuditPrivilege	Пользователь получает возможность указывать параметры аудита доступа к объекту для отдельных ресурсов, таких как файлы, объекты Active Directory и разделы реестра
Овладение файлами или иными объектами SeTakeOwnershipPrivilege	Пользователь получает возможность становиться владельцем любых объектов безопасности системы, включая объекты Active Directory, файлы и папки NTFS, принтеры, разделы реестра, службы, процессы и потоки
Завершение работы системы SeShutdownPrivilege	Пользователь получает возможность завершать работу операционной системы на локальном компьютере
Обход перекрестной проверки SeChangeNotifyPrivilege	Используется для обхода проверки разрешений для промежуточных каталогов при проходе многоуровневых каталогов

Маркер доступа может быть создан не только при первоначальном входе пользователя в систему. Windows предоставляет возможность запуска процессов от имени других пользователей, создавая для этих процессов соответствующий маркер. Для этих целей можно использовать:

– API-функции CreateProcessAsUser, CreateProcessWithLogon;

– оконный интерфейс, инициализирующийся при выборе пункта контекстного меню «Запуск от имени»;

– консольную команду *runas*:

runas /user:имя_пользователя program,

где *имя_пользователя* – имя учетной записи пользователя, которая будет использована для запуска программы в формате *пользователь@домен* или *домен\пользователь*;

program – команда или программа, которая будет запущена с помощью учетной записи, указанной в параметре */user*.

В любом варианте запуска процесса от имени другой учетной записи необходимо задать ее пароль.

Защита объектов системы.

Маркер доступа идентифицирует субъектов-пользователей системы. С другой стороны, каждый объект системы, требующий защиты, содержит описание прав доступа к нему пользователей. Для этих целей используется дескриптор безопасности (Security Descriptor, SD). Каждому объекту системы, включая файлы, принтеры, сетевые службы, контейнеры Active Directory и другие, присваивается

дескриптор безопасности, который определяет права доступа к объекту и содержит следующие основные атрибуты:

– SID владельца, идентифицирующий учетную запись пользователя-владельца объекта;

– пользовательский список управления доступом (Discretionary Access Control List, DACL), который позволяет отслеживать права и ограничения, установленные владельцем данного объекта. DACL может быть изменен пользователем, который указан как текущий владелец объекта;

– системный список управления доступом (System Access Control List, SACL), определяющий перечень действий над объектом, подлежащих аудиту;

– флаги, задающие атрибуты объекта.

Авторизация Windows основана на сопоставлении маркера доступа субъекта с дескриптором безопасности объекта. Управляя свойствами объекта, администраторы могут устанавливать разрешения, назначать право владения и отслеживать доступ пользователей.

Список управления доступом содержит набор элементов (Access Control Entries, ACE). В DACL каждый ACE состоит из четырех частей: в первой указываются пользователи или группы, к которым относится данная запись, во второй – права доступа, третья информирует о том, предоставляются эти права или отбираются четвертая представляет собой набор флагов, определяющих, как данная запись будет наследоваться вложенными объектами (актуально, например, для папок файловой системы, разделов реестра).

Если список ACE в DACL пуст, к нему нет доступа ни у одного пользователя (только у владельца на изменение DACL). Если отсутствует сам DACL в SD объекта – полный доступ к нему имеют все пользователи.

Если какой-либо поток запросил доступ к объекту, подсистема SRM осуществляет проверку прав пользователя, запустившего поток, на данный объект, просматривая его список DACL. Проверка осуществляется до появления разрешающих прав на все запрошенные операции. Если встретится запрещающее правило хотя бы на одну запрошенную операцию, доступ не будет предоставлен.

Стандартные разрешения для файлов:

– Полный доступ (Full Control);
– Изменить (Modify);
– Чтение и выполнение (Read&Execute);
– Чтение (Read);
– Запись (Write).

Стандартные разрешения для папок:

– Полный доступ (Full Control);
– Изменить (Modify);
– Чтение и выполнение (Read&Execute);
– Список содержимого папки;
– Чтение (Read);
– Запись (Write).

Разрешение *Чтение* позволяет просматривать файлы, папки и их атрибуты.

Разрешение *Запись* позволяет создавать новые файлы и папки внутри папок, изменять атрибуты и просматривать владельцев и разрешения.

Разрешение *Список содержимого папки* позволяет просматривать имена файлов и папок.

Разрешение *Чтение и выполнение* для папок позволяет перемещаться по структуре других папок и служит для того, чтобы разрешить пользователю открывать папку, даже если он не имеет прав доступа к ней, для поиска других файлов или вложенных папок. Разрешены все действия, право на которые дают разрешения *Чтение* и *Список содержимого папки*. Это же разрешение для файлов позволяет запускать файлы программ и выполнять действия, право на которые дает разрешение *Чтение*.

Разрешение *Изменить* позволяет удалять папки, файлы и выполнять все действия, право на которые дают разрешения *Запись* и *Чтение* и выполнение.

Разрешение *Полный доступ* позволяет изменять разрешения, менять владельца, удалять файлы и папки и выполнять все действия, на которые дают право все остальные разрешения NTFS.

Разрешения для папок распространяются на их содержимое: подпапки и файлы.

При определении прав доступа к объектам можно задать правила их наследования в дочерних контейнерах. В окне дополнительных параметров безопасности на вкладке *Разрешения* при выборе опции «*Наследовать от родительского объекта применяемого к дочерним объектам разрешения добавляя их к явно заданным в этом окне*» можно унаследовать разрешения и ограничения, заданные для родительского контейнера, текущему объекту.

При выборе опции «*Заменить разрешения для всех дочерних объектов заданными здесь разрешениями применимыми к дочерним объектам*» разрешается передача определенных для объекта-контейнера правил доступа его дочерним объектам.

В этом же окне на вкладке *Владелец* можно узнать владельца объекта и заменить его. Владелец объекта имеет право на изменение списка его DACL, даже если к нему запрещен любой тип доступа. Администратор имеет право становиться владельцем любого объекта.

С учетом возможности вхождения пользователя в различные группы и независимости определения прав доступа к объектам для групп и пользователей зачастую бывает сложно определить конечные права пользователя на доступ к объекту: требуется просмотреть запрещающие правила, определенные для самого объекта, для всех групп, в которые он включен, затем то же проделать для разрешающих правил. Автоматизировать процесс определения разрешенных пользователю видов доступа к объекту можно с использованием вкладки «*Действующие разрешения*» окна дополнительных параметров безопасности объекта.

При назначении пользователю или группе разрешения на доступ к файлу или папке руководствуются тем уровнем доступа, который достаточен для данной группы или пользователя при выполнении им своих рабочих обязанностей.

Рассмотренные разрешения относятся к пользователям данного компьютера, совершившим вход локально непосредственно на данную машину. Такие

разрешения называются разрешениями NTFS.

Разрешения для пользователей, получившим доступ к папке или файлу через сеть, регулируются отдельно с помощью так называемых разрешений общего доступа. Эти разрешения распространяются только на папки, к которым предоставлен общий доступ через сеть, и действуют только для пользователей, обращающихся к папке через сеть. Возможности пользователя задаются разрешениями:

– Полный доступ (Full Control);
– Изменить (Change);
– Чтение (Read).

Каждому пользователю или группе могут быть установлены множественные разрешения через участие в нескольких группах с разным набором разрешений. В этом случае действуют эффективные разрешения – пользователь обладает всеми назначенными ему разрешениями.

Действует приоритет разрешений для файлов над разрешениями для папок и приоритет запрещения над разрешением.

Разрешения, назначенные родительской папке, по умолчанию наследуются всеми подпапками и файлами, содержащимися в папках. Однако есть возможность предотвратить наследование для любой вложенной папки, и в этом случае эта папка сама становится родительской для вложенных в нее папок.

Если папка предоставлена для общего доступа, то на нее распространяются разрешения двух видов:

1) разрешения файловой системы, установленные для пользователей данного компьютера;

2) разрешения общего доступа, объявленные для пользователей, получивших доступ через сеть.

Обычно для папок общего доступа задают разрешения полного доступа, а ограничения вводят установкой разрешений NTFS.

В этом случае действует объединение разрешений NTFS и разрешений для общей папки, при котором наиболее строгое разрешение имеет приоритет над другими.

Команда iCACLS – управление доступом к файлам и папкам.

Команда *iCACLS* позволяет отображать или изменять списки управления доступом (ACL) для файлов и папок в файловой системе.

Права доступа указываются с помощью сокращений. Рассмотрим разрешения для группы BUILTIN\Администраторы:

– (OI) – Object inherit – права наследуются на нижестоящие объекты;
– (CI) – Container inherit – наследование каталога;
– (F) – Full control – полный доступ к папке;
– (I) – Inherit – права наследованы с вышестоящего каталога.

Это означает, что у данной группы есть права на запись, изменение данных в данном каталоге и на изменения NTFS-разрешений. Данные права наследуются на все дочерние объекты в этом каталоге.

Ниже приведен полный список разрешений, которые можно установить с

помощью утилиты *icacls*.

Настройки наследования iCACLS:

– (OI) – наследование объекта;
– (CI) – наследование контейнера;
– (IO) – наследовать только на нижестоящие объекты;
– (NP) – не распространять наследование;
– (I) – разрешение, унаследованное от родительского контейнера.

Список основных прав доступа:

– D – доступ на удаление;
– F – полный доступ;
– N – нет доступа;
– M – изменение;
– RX – чтение и выполнение;
– R – доступ только для чтения;
– W – доступ только для записи.

Используя команду *icacls*, можно сохранить текущий ACL объекта в файле, а затем применить сохраненный список к тем же или другим объектам (своего рода резервный ACL-путь).

Чтобы выгрузить текущие ACL папки C:\PS и сохранить их в текстовый файл export_ps_acl.txt, выполните команду

icacls C:\PS /save c:\backup\export_ps_acl.txt /t*

Эта команда сохраняет ACL не только о самом каталоге, но и о всех подпапках и файлах.

Полученный текстовый файл можно открыть с помощью блокнота или любого текстового редактора.

Чтобы применить сохраненные списки доступа (восстановить разрешения на каталог и все вложенные объекты), выполните команду

icacls C:\PS /restore :\backup\export_ps_acl.txt

Таким образом, процесс переноса прав доступа с одной папки на другую становится намного легче.

С помощью команды *icacls* можно изменить списки доступа к папке. Например, чтобы предоставить пользователю aivanov право на редактирование содержимого папки, выполните команду

icacls C:\PS /grant aivanov:M

Удалить все назначенные разрешения для учетной записи пользователя aivanov можно с помощью команды

icacls C:\PS /remove aivanov

Можно запретить пользователю или группе пользователей доступ к файлу или папке

icacls c:\ps /deny "MSKManagers:(CI)(M)"

Важно, что запрещающие правила имеют больший приоритет, чем разрешающие.

С помощью команды *icacls* можно изменить владельца каталог или папки, например:

icacls c:\ps\secret.docx /setowner aivanov /T /C /L /Q

Параметры, которые используются в команде:

/Q – сообщение об успешном выполнении команды не выводится;

/L – команда выполняется непосредственно над символической ссылкой, а не над конкретным объектом;

/C – выполнение команды будет продолжаться, несмотря на файловые ошибки; при этом сообщения об ошибках все равно будут отображаться;

/T – команда используется для всех файлов и каталогов, которые расположены в указанном каталоге.

Можно изменить владельца всех файлов в каталоге:

icacls c:\ps /setowner aivanov /T /C /L /Q*

Можно сбросить текущие разрешения на объекты: *icacls C:\ps /T /Q /C /RESET*

После выполнения команды все текущие разрешения на папку будут сброшены и заменены на разрешения, наследуемые с вышестоящего объекта (каталога).

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить практическое задание.

Задание

1 Запустите ОС Windows на виртуальной машине. Войдите в систему под учетной записью администратора.

2 Создайте учетную запись нового пользователя *User1* в оснастке «Управление компьютером» (*compmgmt.msc*). При создании новой учетной записи запретите пользователю смену пароля и снимите ограничение на срок действия его пароля. Создайте новую группу *Group1* и включите в нее нового пользователя. Удалите пользователя из других групп. Создайте на диске C: папку *Lab7*. Создайте или скопируйте в эту папку несколько текстовых файлов (*.txt).

3 С помощью команды *runas* запустите сеанс командной строки (*cmd.exe*) от имени вновь созданного пользователя. Командой *whoami* посмотрите SID пользователя и всех его групп, а также текущие привилегии пользователя. Дайте пояснения информации, содержащейся в полученных SID.

4 Убедитесь в соответствии имени пользователя и полученного SID в реестре Windows. Найдите в реестре, каким пользователям в системе присвоены SID, соответствующие шаблону *S-1-5-21-*-*-100** (используйте ключ реестра *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList*).

5 Командой *whoami* определите перечень текущих привилегий пользователя *User1*. Попробуйте изменить системное время командой *time*. Чтобы предоставить пользователю подобную привилегию, запустите оснастку «Локальные параметры безопасности» (*secpol.msc*). Добавьте пользователя в список параметров политики «Изменение системного времени» раздела Локальные политики –

Назначение прав пользователя. После этого перезапустите сеанс командной строки от имени пользователя, убедитесь, что в списке привилегий добавилась *SeSystemtimePriviege.* Попробуйте изменить системное время командой *time*.

6 Убедитесь, что привилегия «Завершение работы системы» (SeShutdownPrivilege) предоставлена пользователю *User1*. После этого попробуйте завершить работу системы из сеанса командной строки пользователя командой *shutdown –s*. Добавьте ему привилегию «Принудительное удаленное завершение» (SeRemoteShutdownPrivilege). Попробуйте завершить работу консольной командой еще раз (отменить команду завершения до ее непосредственного выполнения можно командой *shutdown –a*).

7 Просмотрите разрешения пользователям и группам на папку *c:\ Lab7*. Используйте графический интерфейс.

8 Разрешите пользователю *User1* запись в папку *Lab7*, но запретите запись для группы *Group1*. Попробуйте записать файлы или папки в *Lab7* от имени пользователя *User1*. Объясните результат. Посмотрите эффективные разрешения пользователя *User1* к папке *Lab7* в окне свойств папки.

9 Создайте для папки *Lab7* SACL, позволяющий протоколировать отказы и успехи доступа к этой папке со стороны пользователя *User1* (предварительно проверьте, что в локальной политике безопасности соответствующий тип аудита включен). Запретите пользователю *User1* запись в папку *Lab7*. Попробуйте от имени пользователя *User1* просмотреть содержимое папки, скопировать в нее файлы. Убедитесь, что записи аудита были размещены в журнале безопасности (*eventvwr.msc*).

10 Добавьте нового пользователя по имени *User2*. Создайте каталоги «*Public*» и «*Private*». В каждый из этих каталогов скопируйте исполняемый файл (с расширением *.exe*, *.bat* или *.cmd*) и текстовый файл.

11 Изучите справку по команде *icacls* и команде *takeown*.

12 Напишите командный файл, в котором с помощью команд разграничьте доступ к созданным каталогам и файлам в соответствии со своим вариантом (рисунок 7.1). В отчете приведите код файла и результат его работы.

Содержание отчета

1 Тема и цель работы.
2 Краткий конспект теоретических сведений.
3 Ход выполнения задания с результатами.
4 Выводы по работе.

Вариант1			
Субъекты	**Объекты**		
	Каталог **Public**	Каталог **Private**	Текстовый файл в **«Private»**
Администратор	Полный доступ	Чтение	Нет доступа
User1	Чтение	Изменить, кроме удаления	Изменить
User2	Изменить	Нет доступа	Нет доступа
Вариант 2			
Субъекты	**Объекты**		
	Каталог **Public**	Каталог **Private**	Исполняемый файл в **«Private»**
Администратор	Полный доступ	Чтение и выполнение	Изменить
User1	Изменить	Чтение	Выполнение
User2	Чтение и выполнение	Изменить	Нет доступа
Вариант 3			
Субъекты	**Объекты**		
	Каталог **Public**	Каталог **Private**	**Текстовый файл в «Private»**
Администратор	Полный доступ	Список содержимого	Нет доступа
User1	Чтение	Изменить, кроме удаления	Изменить
User2	Изменить	Нет доступа	Нет доступа
Вариант 4			
Субъекты	**Объекты**		
	Каталог **Public**	Каталог **Private**	**Текстовый файл в «Private»**
Администратор	Изменить	Чтение и выполнение	Нет доступа
User1	Чтение	Изменить	Запрет удаления
User2	Полный доступ, кроме смены владельца	Запись	Нет доступа
Вариант 5			
Субъекты	**Объекты**		
	Каталог **Public**	Каталог **Private**	**Исполняемый файл в «Private»**
Администраторг	Полный доступ	Список содержимого	Выполнение
User1	Чтение	Чтение и удаление	Выполнение, запрет удаления
User2	Изменить, кроме удаления	Запись	Нет доступа

Рисунок 7.1 – Варианты для разработки командного файла

Вариант 6			
Субъекты	Объекты		
	Каталог Public	Каталог Private	Исполняемый файл в «Private»
Администратор	Полный доступ	Чтение	Изменить
User1	Чтение и удаление	Список содержимого	Выполнение
User2	Изменить	Нет доступа	Нет доступа
Вариант 7			
Субъекты	Объекты		
	Каталог Public	Каталог Private	Текстовый файл в «Public»
Администратор	Список содержимого	Полный доступ	Нет доступа
User1	Изменить, кроме удаления	Чтение	Изменить
User2	Нет доступа	Изменить	Нет доступа
Вариант 8			
Субъекты	Объекты		
	Каталог Public	Каталог Private	Текстовый файл в «Private»
Администратор	Чтение и выполнение	Изменить	Нет доступа
User1	Изменить	Чтение	Изменить, запрет изменения дополнительных атрибутов
User2	Запись	Изменить, кроме удаления	Нет доступа
Вариант 9			
Субъекты	Объекты		
	Каталог Public	Каталог Private	Исполняемый файл в «Private»
Администратор	Список содержимого	Полный доступ	Выполнение
User1	Чтение и удаление	Чтение	Выполнение, запрет удаления
User2	Запись	Полный доступ	Нет доступа
Вариант 10			
Субъекты	Объекты		
	Каталог Public	Каталог Private	Исполняемый файл в «Private»
Администратор	Чтение	Полный доступ	Изменить
User1	Список содержимого	Чтение и удаление	Выполнение
User2	Нет доступа	Изменить	Нет доступа

Окончание рисунка 7.1

Контрольные вопросы

1 Каким образом реализуется защита файлов в NTFS?

2 Перечислите стандартные права доступа к файловым объектам, существующие в файловой системе NTFS.

3 Объясните принцип работы разрешения «Запись».

4 Перечислите элементы разрешений.

5 Кто может стать владельцем объекта?

6 Раскройте понятие наследования разрешений.

7 Как отключить наследование разрешений?

8 Как реализовать принудительное наследование вложенными объектами установленных разрешений?

9 Перечислите приоритеты применения разрешений при определении действующих разрешений на доступ к файловым объектам.

8 Лабораторная работа № 8. Основы криптографии

Цель работы: ознакомиться с основами криптографии и изучить виды шифрования.

Теоретические сведения

Один из методов защиты информации от неправомерного доступа – это шифрование, т. е. кодирование специального вида.

Шифрование – это преобразование (кодирование) открытой информации в зашифрованную, недоступную для понимания посторонних.

Шифрование применяется в первую очередь для передачи секретной информации по незащищенным каналам связи. Шифровать можно любую информацию – тексты, рисунки, звук, базы данных и т. д.

Человечество применяет шифрование с того момента, как появилась секретная информация, которую нужно было скрыть от врагов. Первое известное науке шифрованное сообщение – египетский текст, в котором вместо принятых тогда иероглифов были использованы другие знаки.

Методы шифрования и расшифровывания сообщения изучает наука криптология, история которой насчитывает около четырех тысяч лет. Она состоит из двух ветвей: криптографии и криптоанализа.

Криптография – это наука о способах шифрования информации.

Критоанализ – это наука о методах и способах вскрытия шифров.

Обычно предполагается, что сам алгоритм шифрования известен всем, но неизвестен его ключ, без которого сообщение невозможно расшифровать. В этом заключается отличие шифрования от простого кодирования, при котором для восстановления сообщения достаточно знать только алгоритм кодирования.

Ключ – это параметр алгоритма шифрования (шифра), позволяющий выбрать одно конкретное преобразование из всех вариантов, предусмотренных алгоритмом. Знание ключа позволяет свободно зашифровывать и расшифровывать сообщения.

Все шифры (системы шифрования) делятся на две группы – симметричные и несимметричные (с открытым ключом).

Симметричный шифр означает, что и для шифрования, и для расшифровывания сообщений используется один и тот же ключ. В системах с открытым ключом используются два ключа – открытый и закрытый, которые связаны друг с другом с помощью некоторых математических зависимостей. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

Криптостойкость шифра – это устойчивость шифра к расшифровке без знания ключа.

Стойким считается алгоритм, который для успешного раскрытия требует от противника недостижимых вычислительных ресурсов, недостижимого объема перехваченных сообщений или такого времени, что по его истечении защищенная информация будет уже неактуальна.

Шифр Цезаря.

Данный шифр назван в честь римского императора Гая Юлия Цезаря, использовавшего его для секретной переписки. Это один из самых известных и самых древних шифров. В этом шифре каждая буква заменяется на другую, расположенную в алфавите на заданное число позиций k вправо от нее. Алфавит замыкается в кольцо, так что последние символы заменяются на первые. На рисунке 8.1 приведен шифр Цезаря со сдвигом 3.

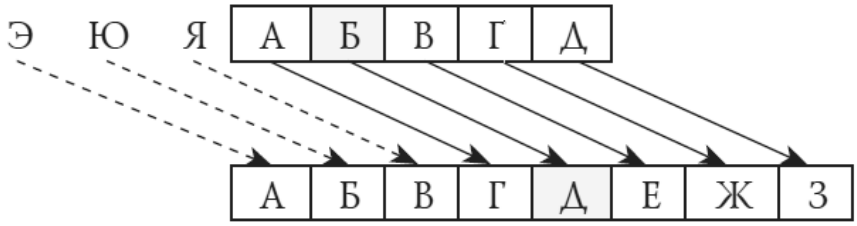

Рисунок 8.1 – Шифр Цезаря со сдвигом 3

Знаменитая фраза «ПРИШЕЛ УВИДЕЛ ПОБЕДИЛ» при использовании шифра Цезаря со сдвигом 3 будет закодирована так: «Тулыио целзио тсдизло».

Если первая буква алфавита имеет код 0, вторая – код 1 и т. д., алгоритм шифрования может быть выражен формулой

$$y = (x + k) \bmod n,$$

где x – код исходного символа;

k – величина сдвига;

y – код символа-замены;

n – количество символов в алфавите;

а запись $(x + k) \bmod n$ обозначает остаток от деления $x + k$ на n.

Операция взятия остатка от деления необходима для того, чтобы «замкнуть» алфавит в кольцо.

Например, при использовании русского алфавита (32 буквы) будем считать, что буквы Е и Ё совпадают. Для буквы Я (код 31) получаем код заменяющего символа $(31 + 3) \bmod 32 = 2$ – это буква В.

Ключом для шифра Цезаря служит сдвиг k; если его знать, то сообщение легко расшифровать. Для этого используется формула

$$x = (y - k + n) \bmod n.$$

Шифр Цезаря относится к шифрам простой подстановки, т. к. каждый символ исходного сообщения заменяется на другой символ из того же алфавита. Такие шифры легко раскрываются с помощью частотного анализа, потому что в каждом языке частоты встречаемости букв примерно постоянны для любого достаточно большого текста.

Шифр Виженера.

Значительно сложнее сломать шифр Виженера (назван по имени Блеза Виженера, швейцарского дипломата XVI в.), который стал естественным развитием шифра Цезаря.

Для использования шифра Виженера используется ключевое слово, которое задает переменную величину сдвига. Например, пусть ключевое слово – «ЗАБЕГ». По таблице (рисунок 8.2) определяем коды букв.

0	1	2	3	4	5	6	7	8	9	
А	Б	В	Г	Д	Е	Ж	З	И	К	...

Рисунок 8.2 – Шифр Цезаря со сдвигом 3

Получаем: З – 7, А – 0, Б – 1, Е – 5, Г – 3. Это значит, что для кодирования первой буквы используется сдвиг 7, для кодирования второй – 0 (символ не меняется) и т. д. Для пятой буквы используется сдвиг 3, а для шестой — снова 7 (начали «проходить» кодовое слово сначала). Фраза «ПРИШЕЛ УВИДЕЛ ПОБЕДИЛ» при использовании шифра Виженера с ключом «ЗАБЕГ» будет закодирована в виде «ЦРЙЭИТ ФЗЛЛЕМ ТХБЖЙЛТ».

Шифр Виженера обладает значительно более высокой криптостойкостью, чем шифр Цезаря. Это значит, что его труднее раскрыть – подобрать нужное ключевое слово. Теоретически, если длина ключа равна длине сообщения и каждый ключ используется только один раз, шифр Виженера взломать невозможно.

Порядок выполнения работы

1 Изучить основные теоретические положения.
2 Выполнить задание согласно своему варианту (вариант задает преподаватель).

Задание

Вариант 1
Составьте программу, которая выполняет шифрование текста с помощью шифра Цезаря.

Вариант 2
Составьте программу, которая выполняет шифрование текста с помощью шифра Виженера.

Содержание отчета

1 Тема и цель работы.
2 Условие задания.
3 Листинг программы.
4 Тестирование.
5 Выводы по работе.

Контрольные вопросы

1 Чем отличаются понятия «шифрование» и «кодирование»?
2 Что такое ключ?
3 Как называется наука, изучающая методы шифрования?
4 Что такое симметричный шифр? Какая проблема возникает при использовании симметричного шифра, если участники переписки находятся в разных странах?
5 Что такое несимметричные шифры? На чем основана их надежность?
6 Что такое криптостойкость алгоритма? Какой алгоритм считается криптостойким?

Список литературы

1 **Аверченков, В. И.** Информационная безопасность сетей и систем: учеб. пособие / В. И. Аверченков, В. Т. Еременко, Е. А. Зайченко. – Могилев: Белорус.-Рос. ун-т., 2020. – 212 с.

2 **Баранова, Е. К.** Информационная безопасность и защита информации: учеб. пособие / Е. К. Баранова, А. В. Бабаш. – 4-е изд., перераб. и доп. – М.: РИОР; ИНФРА-М, 2021. – 336 с.

3 **Щеглов, А. Ю.** Защита информации: основы теории: учебник для бакалавриата и магистратуры / А. Ю. Щеглов, К. А. Щеглов. – М.: Юрайт, 2019. – 309 с.