МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»

ВОЛЖСКИЙ ФИЛИАЛ

Кафедра гуманитарных и естественнонаучных дисциплин

Методические указания к лабораторным работам
по дисциплине

ЗАЩИТА ИНФОРМАЦИИ

Направление подготовки

09.03.01 Информатика и вычислительная техника

Направленность (профиль, специализация) образовательной программы

«Автоматизированные системы обработки информации и управления»

Квалификация

бакалавр

Чебоксары
2019

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1.

«Средства обеспечения безопасности ОС Windows»

Цель: изучить модель безопасности операционной системы Windows, получить навыки практического использования ее средств обеспечения безопасности.

1. Основные 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.

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

2. - 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;

3. - разграничительный контроль доступа — предоставление пользователям возможности защиты принадлежащих им данных;

4. - системный аудит — способность системы вести подробный аудит всех действий, выполняемых пользователями и самой операционной системой;

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

ОС Microsoft Windows XP SP3 имеет действующие сертификаты ФСТЭК России и может использоваться в составе автоматизированных систем до класса защищенности 1Г включительно в соответствии с требованиями руководящего документа "Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требований по защите информации" (Гостехкомиссия России, 1992).

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 указываются, 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. Подобная схема доступа, очевидно, требует аутентификации каждого пользователя, получающего доступ к ресурсам и его надежную идентификацию в системе, а также механизмов описания прав пользователей и групп пользователей в системе, описания и

проверки дискреционных прав доступа пользователей к объектам. 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:

S - неизменный идентификатор строкового представления SID;

S – R – I – S0 - S1 - … - Sn – RID

R – уровень ревизии (версия). На сегодня 1.

I - (identifier-authority) идентификатор полномочий . Представляет собой 48-битную строку, идентифицирующую компьютер или сеть, который(ая) выдал SID объекту. Возможные значения:

- 0 (SECURITY_NULL_SID_AUTHORITY) — используются для сравнений, когда неизвестны полномочия идентификатора;
- 1 (SECURITY_WORLD_SID_AUTHORITY) — применяются для конструирования идентификаторов SID, которые представляют всех пользователей. Например, идентификатор SID для группы *Everyone* (Все пользователи) — это *S-1-1-0*;
- 2 (SECURITY_LOCAL_SID_AUTHORITY) — используются для построения идентификаторов SID, представляющих пользователей, которые входят на локальный терминал;
- 5 (SECURITY_NT_AUTHORITY) — сама операционная система. То есть, данный идентификатор выпущен компьютером или доменом.

6. **Sn** – 32-битные коды (колчеством 0 и более) субагентов, которым было передано право выдать SID. Значение первых подчиненных полномочий общеизвестно. Они могут иметь значение:

- 5 — идентификаторы SID присваиваются сеансам регистрации для выдачи прав любому приложению, запускаемому во время определенного сеанса регистрации. У таких идентификаторов SID первые подчиненные полномочия установлены как 5 и принимают форму *S-1-5-5-x-y*;
- 6 —когда процесс регистрируется как служба, он получает специальный идентификатор SID в свой маркер для обозначения данного действия. Этот идентификатор SID имеет подчиненные полномочия 6 и всегда будет *S-1-5-6*;
- 21 (SECURITY_NT_NON_UNIQUE) — обозначают идентификатор SID пользователя и идентификатор SID компьютера, которые не являются уникальными в глобальном масштабе;
- 32 (SECURITY_BUILTIN_DOMAIN_RID) — обозначают встроенные идентификаторы SID. Например, известный идентификатор SID для встроенной группы администраторов *S-1-5-32-544*;
- 80 (SECURITY_SERVICE_ID_BASE_RID) — обозначают идентификатор SID, который принадлежит службе.

Остальные подчиненные полномочия идентификатора совместно обозначают домен или компьютер, который издал идентификатор SID.

RID – 32-битный относительный идентификатор. Он является является идентификатором уникального объекта безопасности в области, для которой был определен SID. Например, 500 — обозначает встроенную учетную запись *Administrator,* 501 — обозначает встроенную учетную запись *Guest, а* 502 — RID для билета на получение билетов протокола Kerberos .

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. В таблице 1.1. приведен перечень некоторых общеизвестных SID.

Таблица 1.1. Общеизвестные SID Windows

SID	Название	Описание
S-1-1-0	Все	Группа, в которую входят все пользователи
S-1-5-2	Сеть	Группа, в которую входят все пользователи, зарегистрировавшиеся в системе из сети
S-1-5-7	Анонимный вход	Группа, в которую входят все пользователи, вошедшие в систему анонимно
S-1-5-*домен*-500	Администратор	Учетная запись администратора системы. По умолчанию только эта запись обеспечивает полный контроль системы
S-1-5-*домен*-501	Гость	Учетная запись пользователя-гостя

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 наследуются, поэтому маркер доступа олицетворяет пользователя для системы в каждом запущенном от его имени процессе. Основные элементы маркера представлены на рис. 1.1.

SID пользователя	**SID1 … SIDn** **Идентификаторы групп пользователя**	**DACL по умолчанию**	**Привилегии**	**Прочие параметры**

Рисунок 1.1. Обобщенная структура маркера доступа.

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 1.2. перечислены некоторые привилегии, которые могут быть предоставлены пользователю.

Таблица 1.2. Привилегии, которыми могут быть наделены пользователи

Имя и идентификатор привилегии	Описание привилегии
Увеличение приоритета диспетчирования SeIncreaseBasePriorityPrivilege	Пользователь, обладающий данной привилегией может изменять приоритет диспетчирования процесса с помощью интерфейса Диспетчера задач
Закрепление страниц в памяти	Процесс получает возможность хранить данные в физической памяти, не прибегая к кэшированию данных в виртуальной

SeLockMemoryPrivilege	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

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. Необходимо также отметить возможность создания ограниченных маркеров (restricted token), которые отличаются от обычных тем, что из них удаляются некоторые привилегии и его SID-идетификаторы проверяются только на запрещающие правила. Создать ограниченный маркер можно программно, используя API-функцию **CreateRestrictedToken**, а можно запустить процесс с ограниченным маркером, используя пункт контекстного меню Windows **“Запуск от имени…”** и отметив пункт **“Защитить компьютер от несанкционированных действий этой программы”** (рис.1.2).

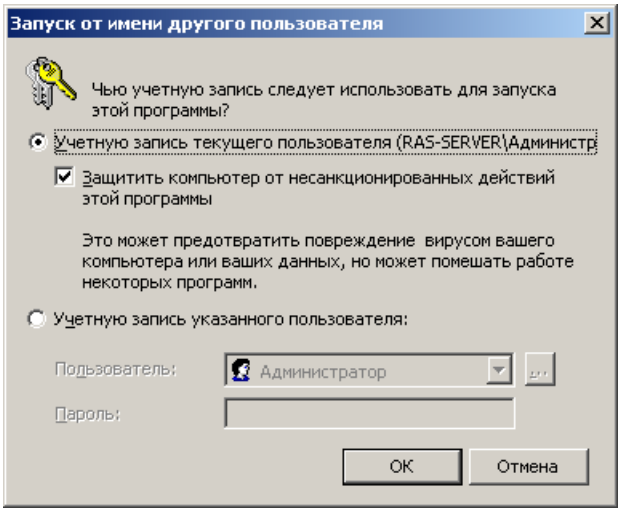


Рис. 1.2.

Ограниченные маркеры используются для процессов, подменяющих клиента и выполняющих небезопасный код.

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 (рис.2), 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\пользова-тель*;
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 (рис.1.3):
- 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.

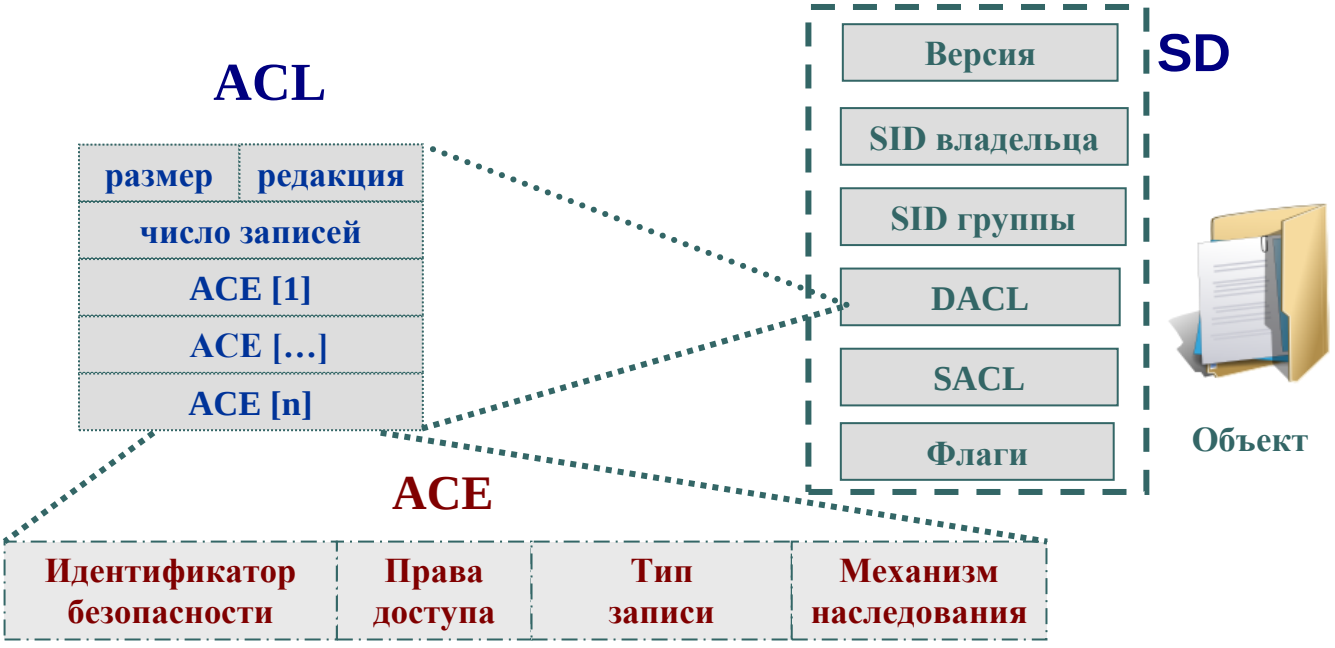


Рис. 1.3

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, а третья информирует о том, предоставляются эти права или отбираются. Четвертая часть 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.

Рассмотрим пример на рис.1.4. Процесс пытается получить доступ к объекту с заданным DACL. В маркере процесса указаны SID запустившего его пользователя, а также SID групп, в которые он входит. В списке DACL объекта присутствуют разрешающие правила на чтение для пользователя с SID=100, и на запись для группы с SID=205. Однако, в доступе пользователю будет отказано, поскольку раньше встречается запрещающее запись правило для группы с SID=201.

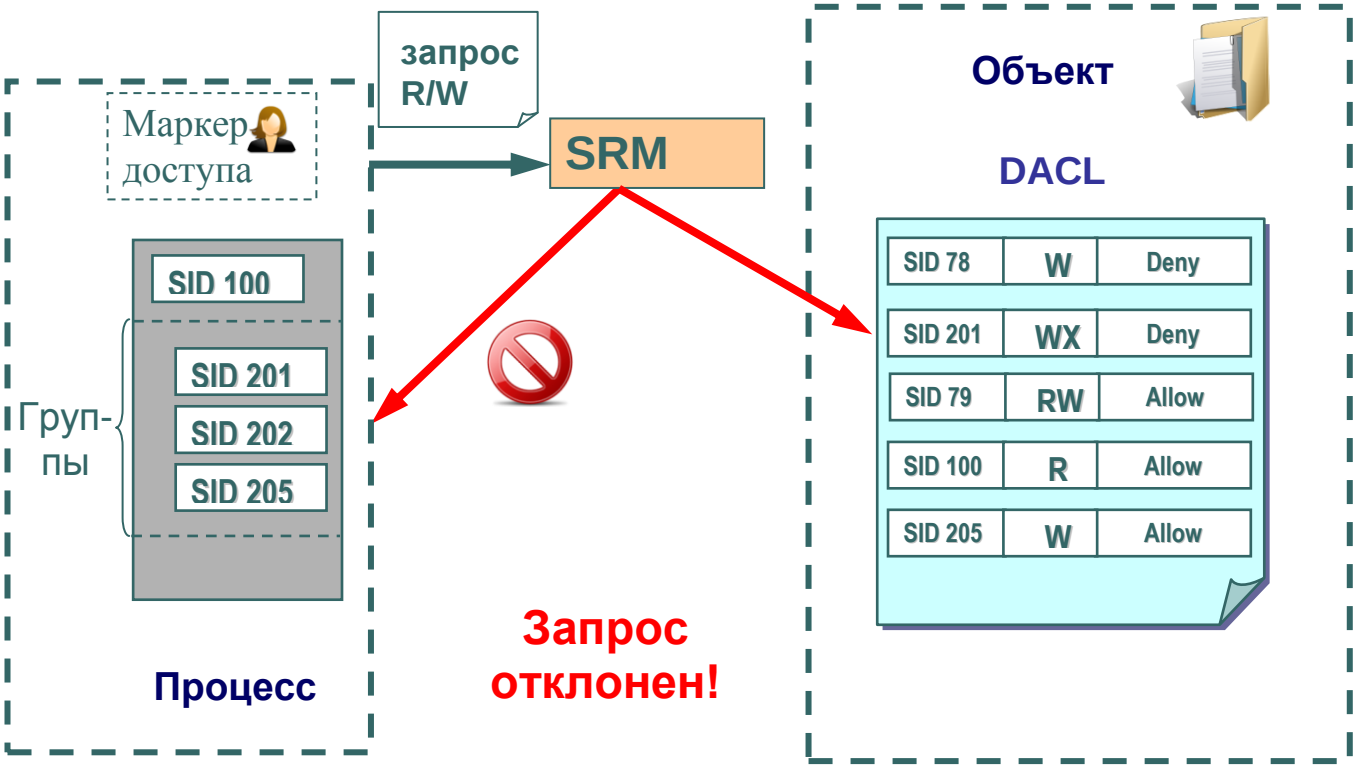


Рис. 1.4.

Необходимо отметить, что запрещающее правило помещено в списке DACL на рисунке не случайно. Запрещающие правила **всегда** размещаются перед разрешающими, то есть являются доминирующими при проверке прав доступа.

Для определения и просмотра прав доступа пользователей к ресурсам можно использовать как графические средства контроля, так и консольные команды. Стандартное окно свойств объекта файловой системы (диска, папки, файла) на вкладке **Безопасность** (рис. 1.5) позволяет просмотреть текущие разрешения для пользователей и групп пользователей, редактировать их, создавать новые или удалять существующие.

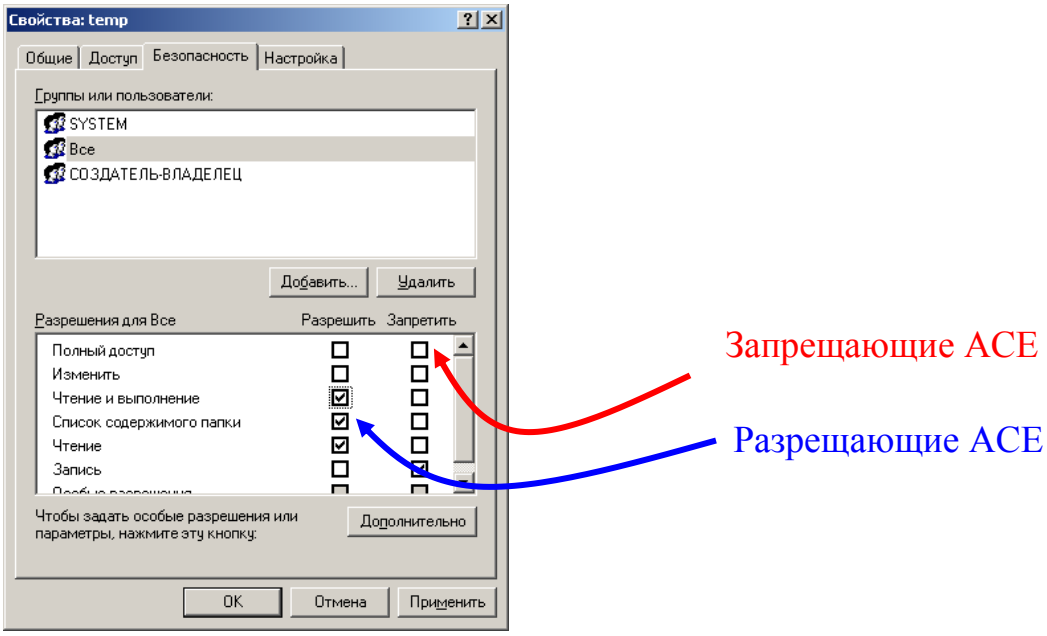


Рис. 1. 5.

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 применимых к дочерним объектам разрешения, добавляя их к явно заданным в этом окне»** (рис. 1.6) 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** допустимо узнать владельца объекта и заменить его. Владелец объекта имеет право на изменение списка его DACL, даже если к нему запрещен

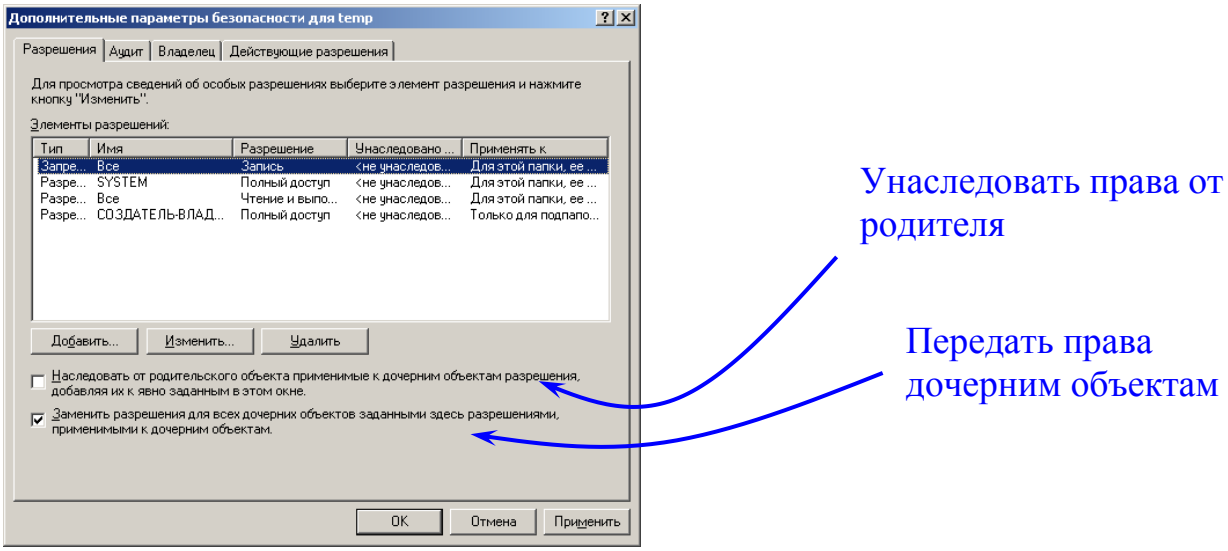


Рис. 1.6.

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 (рис. 1.7).

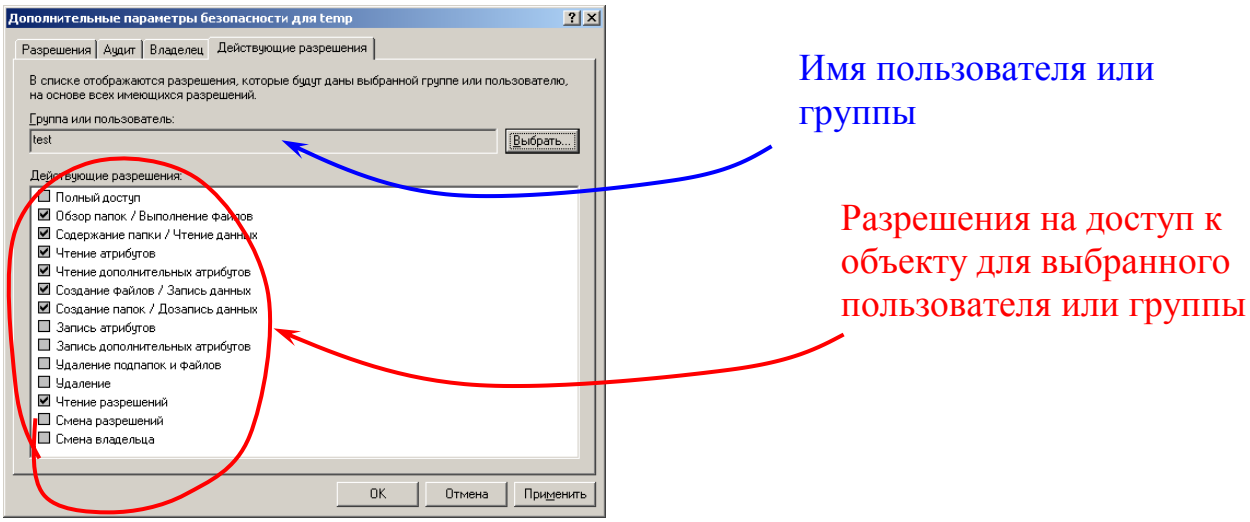


Рис. 1.7.

Для просмотра и изменения прав доступа к объектам в режиме командной строки предназначена команда **cacls** (**icacls** в Windows Vista и Widows 7).

cacls *имя_файла* [**/t**] [**/e**] [**/c**] [**/g** *пользователь*:*разрешение*] [**/r** *пользователь* [...]] [**/p** *пользователь*:*разрешение* [...]] [**/d** *пользователь* [...]]

Назначения параметров команды приведены в таблице 1.3.

Таблица 1.3. Параметры команды cacls

<имя файла>	Задаёт файл или папку, права доступа к которой необходимо просмотреть/изменить (допустимо использовать шаблоны с символами ***** и **?**).
/t	Изменение избирательных таблиц контроля доступа (DACL) указанных файлов в текущем каталоге и всех подкаталогах
/e	Редактирование избирательной таблицы управления доступом (DACL) вместо ее замены
/c	Заставляет команду продолжить изменение прав доступа при возникновении ошибки, связанной с нарушениями прав доступа
/g <пользователь | группа: разрешение>	Предоставление прав доступа указанному пользователю
/r <пользователь | группа>	Отнимает права доступа указанного пользователя.
/p <пользователь | группа: разрешение>	Заменяет права доступа указанного пользователя
/d <пользователь | группа>	Отказывает в праве доступа указанному пользователю или группе

Для указания добавляемых или отнимаемых прав используются следующие значения:

F - полный доступ;
C - изменение (запись);
W - запись;
R - чтение;
N - нет доступа.

Рассмотрим несколько примеров.

cacls d:\test

Выдаст список DACL для папки test.

cacls d:\test /d ИмяКомпьютера\ИмяПользователя /e

Запретит доступ к объекту для указанного пользователя.

cacls d:\test /p ИмяКомпьютера\ИмяГруппы:f /e /t

Предоставит полный доступ к папке d:\test и ее подпапках всем для членов указанной группы.

Для программного просмотра и изменения списков DACL можно использовать API-функции **AddAccessAllowedAce, AddAccessDeniedAce, SetSecurityInfo**. Подробнее с этими функциями и примерами их использования можно ознакомиться в [пособие].

1.4. Подсистема аудита.

Важный элемент политики безопасности – аудит событий в системе. ОС Windows ведет аудит событий по 9 категориям:

1. Аудит событий входа в систему.
2. Аудит управления учетными записями.
3. Аудит доступа к службе каталогов.
4. Аудит входа в систему.
5. Аудит доступа к объектам.
6. Аудит изменения политики.
7. Аудит использования привилегий.
8. Аудит отслеживания процессов.
9. Аудит системных событий.

Рассмотрим более подробно, какие события отслеживает каждая из категорий.

Аудит событий входа в систему

Аудит попыток пользователя войти в систему с другого компьютера или выйти из нее, при условии, что этот компьютер используется для проверки подлинности учетной записи.

Аудит управления учетными записями

Аудит событий, связанных с управлением учетными записями на компьютере: создание, изменение или удаление учетной записи пользователя или группы; переименование, отключение или включение учетной записи пользователя; задание или изменение пароля.

Аудит доступа к службе каталогов

Аудит событий доступа пользователя к объекту каталога Active Directory, для которого задана собственная системная таблица управления доступом (SACL).

Аудит входа в систему

Аудит попыток пользователя войти в систему с компьютера или выйти из нее.

Аудит доступа к объектам

Аудит событий доступа пользователя к объекту – например, к файлу, папке, разделу реестра, принтеру и т. п., - для которого задана собственная системная таблица управления доступом (SACL).

Аудит изменения политики

Аудит фактов изменения политик назначения прав пользователей, политик аудита или политик доверительных отношений.

Аудит использования привилегий

Аудит попыток пользователя воспользоваться предоставленным ему правом.

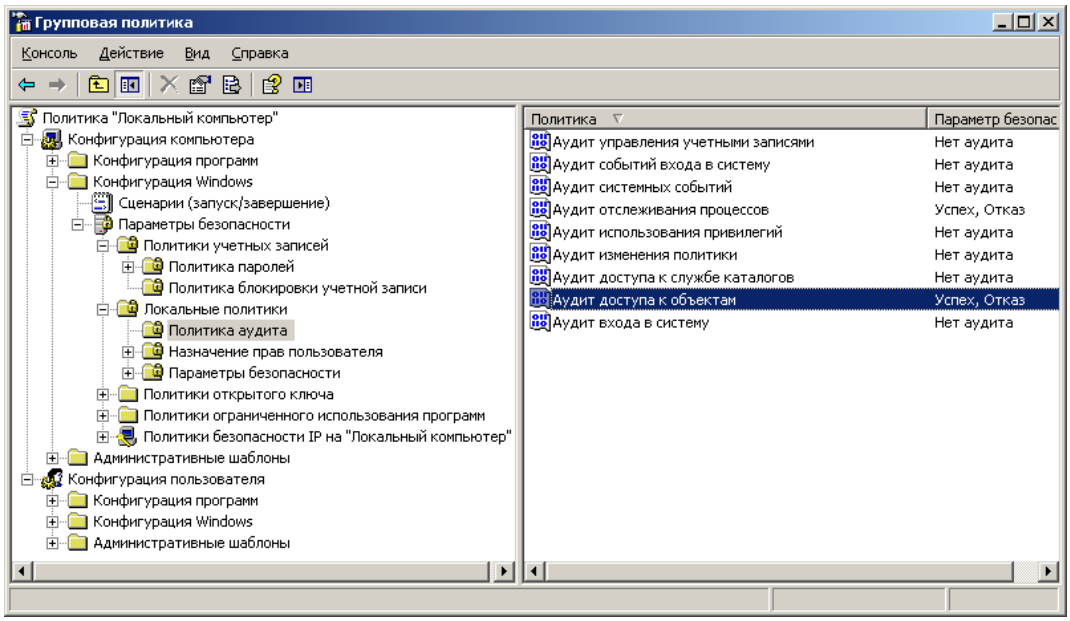


Рис. 1.8.

Аудит отслеживания процессов

Аудиту таких событий, как активизация программы, завершение процесса, повторение дескрипторов и косвенный доступ к объекту.

Аудит системных событий

Аудит событий перезагрузки или отключения компьютера, а также событий, влияющих на системную безопасность или на журнал безопасности.

Решения об аудите конкретного типа событий безопасности принимаются в соответствии с политикой аудита локальной системы. Политика аудита, также называемая локальной политикой безопасности (local security policy), является частью политики безопасности, поддерживаемой LSASS в локальной системе, и настраивается с помощью редактора локальной политики безопасности (Оснастка **gpedit.msc, Конфигурация компьютера - Конфигурация Windows – Параметры безопасности – Локальные политики – Политика аудита,** рис. 1.8).

Для каждого объекта в SD содержится список SACL, состоящий из записей ACE, регламентирующих запись в журнал аудита удачных или неудачных попыток доступа к объекту. Эти ACE определяют, какие операции, выполняемые над объектами конкретными пользователями или группами, подлежат аудиту. Информация аудита хранится в системном журнале аудита. Аудиту могут подлежать как успешные, так и неудачные операции. Подобно записям ACE DACL, правила аудита объектов могут наследоваться дочерними объектами. Процедура наследования определяются набором флагов, являющихся частью структуры ACE.

Настройка списка SACL может быть осуществлена в окне дополнительных свойств объекта (**пункт “Дополнительно”, закладка “Аудит”,** рис.1. 9).

Для программного просмотра и изменения списков SACL можно использовать API-функции **GetSecutityInfo** и **SetSecutityInfo**.

При инициализации системы и изменении политики LSASS посылает SRM сообщения, информирующие его о текущей политике аудита. LSASS отвечает за прием записей аудита, генерируемых на основе событий аудита от SRM, их редактирование и передачу Event Logger (регистратору событий). SRM посылает записи аудита LSASS через свое LPC-соединение. После этого Event Logger заносит записи в журнал безопасности.

События аудита записываются в журналы следующих типов:

1. **Журнал приложений.** В журнале приложений содержатся данные, относящиеся к работе приложений и программ.
2. **Журнал безопасности.** Журнал безопасности содержит записи о таких событиях, как успешные и безуспешные попытки доступа в систему, а также о событиях, относящихся к использованию ресурсов.
3. **Журнал системы.** В журнале системы содержатся события системных компонентов Windows. Например, в журнале системы регистрируются сбои при загрузке драйвера или других системных компонентов при запуске системы.
4. **Журнал службы каталогов.** В журнале службы каталогов содержатся события, заносимые службой каталогов Windows (на контроллере домена AD).
5. **Журнал службы репликации.** В журнале службы репликации файлов содержатся события, заносимые службой репликации файлов Windows (на контроллере домена AD).

Просмотр журнала безопасности осуществляется в оснастке «Просмотр событий» (**eventvwr.msc**). Сами журналы хранятся в файлах **SysEvent.evt, SecEvent.evt, AppEvent.evt** в папке **%WinDir%\system32\config.**

В журнал записываются события 3 основных видов:

1. Информационные сообщения о событиях.

Описывают успешное выполнение операций, таких как запуск или некоторое действие системной службы.

2. Предупреждающие сообщения о событиях.

Описывают неожиданные действия, означающие проблему, или указывают на проблему, которая возникнет в будущем, если не будет устранена сейчас .

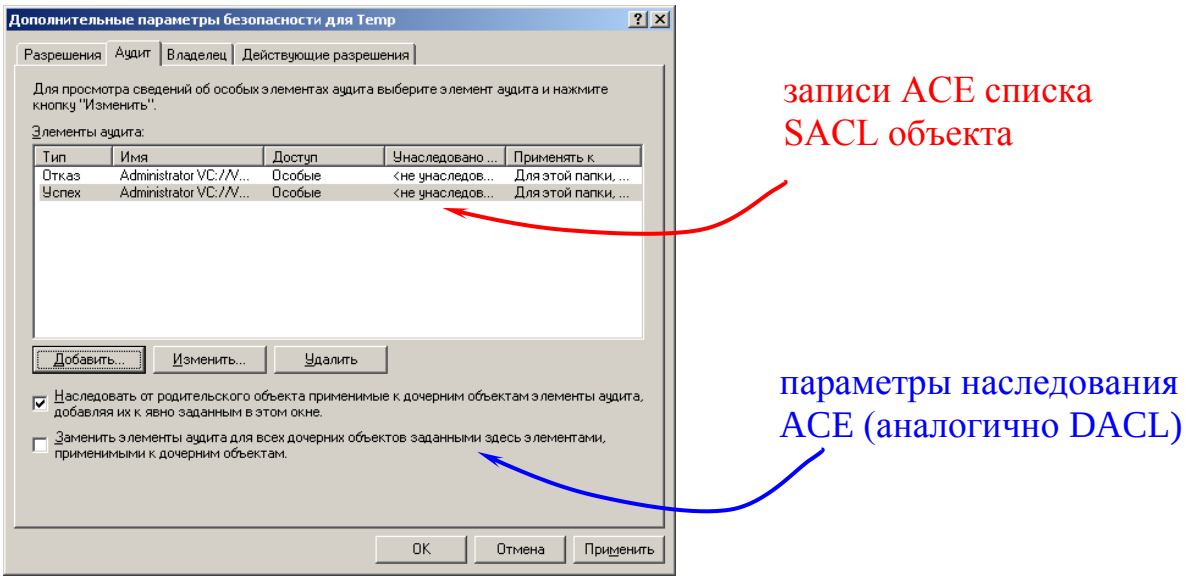


Рис.1.9.

3. Сообщения о событиях ошибок.

Описывают ошибки, возникшие из-за неудачного выполнения задач.

Шифрующая файловая система.

Начиная с версии Windows 2000, в операционных системах семейства Windows NT поддерживается шифрование данных на разделах файловой системы NTFS с использованием *шифрующей файловой системы* (**Encrypted File System, EFS**). Основное ее достоинст во заключается в обеспечении конфиденциальности данных на дисках компьютера за счет использования надежных симметричных алгоритмов для шифрования данных в реальном режиме времени.

Для шифрации данных EFS использует симметричный алгоритм шифрования (AES или DESX) со случайным ключом для каждого файла (**File Encryption Key, FEK**). По умолчанию данные шифруются в Windows 2000 и Windows XP по алгоритму DESX, а в Windows XP с Service Pack 1 (или выше) и Windows Server 2003 — по алгоритму AES. В версиях Windows, разрешенных к экспорту за пределы США, драйвер EFS реализует 56-битный ключ шифрования DESX, тогда как в версии, подлежащей использованию только в США, и в версиях с пакетом для 128-битного шифрования длина ключа DESX равна 128 битам. Алгоритм AES в Windows использует 256-битные ключи.

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 (рис. 1.11.). 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.

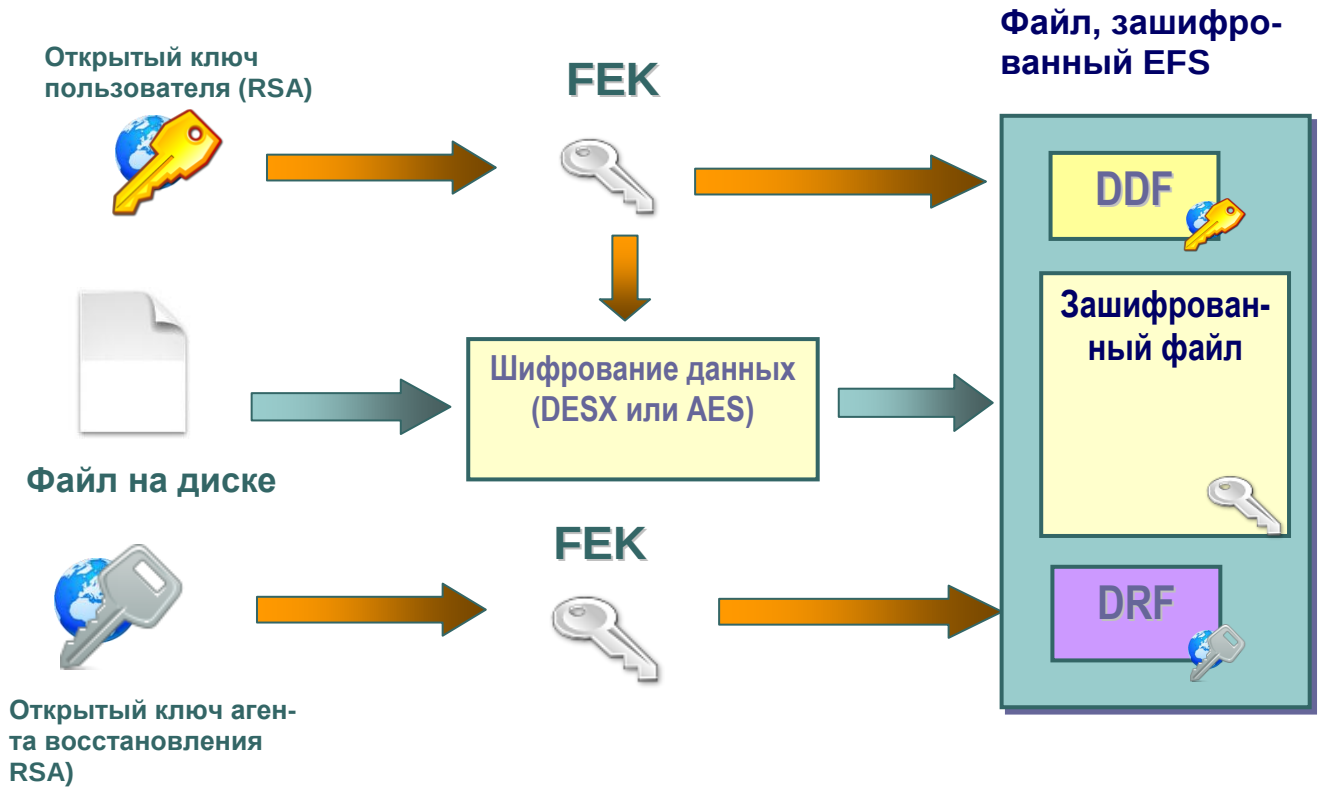


Рис. 1.11.

Для шифрации файлов с использованием EFS можно использовать графический интерфейс или команду **cipher**.

Графический интерфейс доступен в стандартном окне свойств объекта по нажатию кнопки **«Дополнительно»** (рис. 1.10). Зашифрованные объекты в стандартном интерфейсе Windows Explorer отображаются зеленым цветом.

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.

Шифрование файла с использованием EFS защищает файл комплексно: пользователю, не имеющему права на дешифрацию файла, недопустимы, в том числе, такие операции, как

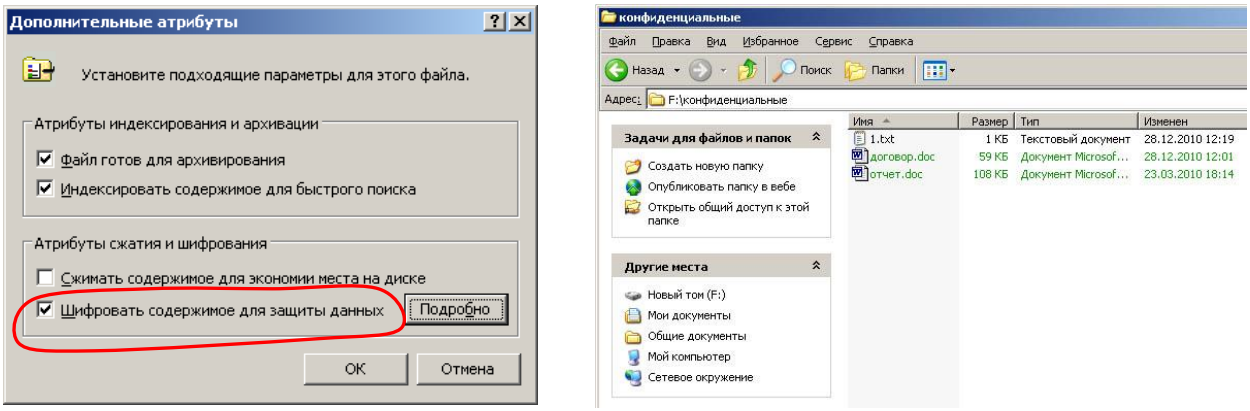


Рис. 1.10.

удаление, переименование и копирование файла. Необходимо помнить, что EFS является частью файловой системы NTFS, и в случае копирования защищенного файла авторизованным пользователем на другой том с файловой системой, на поддерживающей EFS (например, FAT32), он будет дешифрован и сохранен на целевом томе в открытом виде.

Консольная команда **cipher** может быть использована для шифрации/дешифрации файлов из командной строки или в bat-сценарии.

cipher [{**/e**|**/d**}] [**/s:***каталог*] [**/a**] [**/i**] [**/f**] [**/q**] [**/h**] [**/k**] [**/u**[**/n**]] [*путь* [...]] |
[**/r:***имя_файла_без_расширения*]

Назначения параметров команды приведены в таблице 1.4.

Таблица 1.4. Параметры команды **cipher**

/e	Шифрует указанные папки. Папки помечаются таким образом, чтобы файлы, которые будут добавляться в папку позже, также шифровались.
/d	Расшифровывает указанные папки. Папки помечаются таким образом, чтобы файлы, которые будут добавляться в папку позже, не будут шифроваться
/s: *каталог*	Выполняет выбранную операцию над указанной папкой и всеми подпапками в ней.
/a	Выполняет операцию над файлами и каталогами
/i	Продолжение выполнения указанной операции даже после возникновения ошибок. По умолчанию выполнение **cipher** прекращается после возникновения ошибки
/f	Выполнение повторного шифрования или расшифровывания указанных объектов. По умолчанию уже зашифрованные или расшифрованные файлы пропускаются командой **cipher**
/k	Создание ключа шифрования файла для пользователя, выполнившего команду **cipher**. Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.
/u	Обновление ключа шифрования файла пользователя или ключа агента восстановления на текущие ключи во всех зашифрованных файлах на локальном диске (если эти ключи были изменены). Этот параметр используется только вместе с параметром **/n**.
/n	Запрещение обновления ключей. Данный параметр служит для поиска всех зашифрованных файлов на локальных дисках. Этот параметр используется

	только вместе с параметром **/u**.
путь	Указывает шаблон, файл или папку.
/r: ***имя_файла***	Создание нового сертификата агента восстановления и закрытого ключа с последующей их записью в файлах с именем, указанным в параметре *имя_файла* (без расширения). Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.

Например, чтобы определить, зашифрована ли какая-либо папка, необходимо использовать команду:

cipher ***путь\имя_папки***

Команда **cipher** без параметров выводит статус (зашифрован или нет) для всех объектов текущей папки.
Для шифрации файла необходимо использовать команду

cipher /e /a ***путь\имя_файла***

Для дешифрации файла, соответственно, используется команда

cipher /d /a ***путь\имя_файла***

Допустима шифрация/дешифрация группы файлов по шаблону:

cipher /e /a d:\work*.doc

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

Если некоторый пользователь или группа пользователей зашифровали файл с использованием EFS, то его содержимое доступно только им. Это приводит к рискам утери доступа к данным в зашифрованных файлах в случае утраты пароля данным пользователем (работник забыл пароль, уволился и т.п.). Для предотвращения подобных проблем администратор может определить некоторые учетные записи в качестве агентов восстановления.

Агенты восстановления (Recovery Agents) определяются в политике безопасности **Encrypted Data Recovery Agents (Агенты восстановления шифрованных данных)** на локальном компьютере или в домене. Эта политика доступна через оснастку **Групповая политика (gpedit.msc)** раздел **«Параметры безопасности»-> «Политика открытого ключа»-> «Файловая система EFS».** Пункт меню **«Действие»-> «Добавить агент восстановления** данных» открывает мастер добавления нового агента.

Добавляя агентов восстановления можно указать, какие криптографические пары (обозначенные их сертификатами) могут использовать эти агенты для восстановления шифрованных данных (рис. 1.12). Сертификаты для агентов восстановления создаются командой **cipher** с ключом **/r** (см. табл. 4). Для пользователя, который будет агентом восстановления, необходимо импортировать закрытый ключ агента восстановления из сертификата, созданного командой **cipher**. Это можно сделать в мастере импорта сертификатов, который автоматически загружается при двойном щелчке по файлу *.pfx.

EFS создает – DRF **(Data Recovery Field)**-элементы ключей для каждого агента восстановления, используя провайдер криптографических сервисов, зарегистрированный для EFS-восстановления. DRF добавляется в зашифрованный файл и может быть использован как альтернативное средство извлечения FEK для дешифрации содержимого файла.

Windows хранит закрытые ключи в подкаталоге **Application Data\Micro-soft\Crypto\RSA** каталога профиля пользователя. Для защиты закрытых ключей Windows шифрует все файлы в папке RSA на основе симметричного ключа, генерируемого случайным образом; такой ключ называется мастер-ключом пользователя. Мастер-ключ имеет длину в 64 байта и создается стойким генератором случайных чисел. Мастер-ключ также хранится в профиле пользователя в каталоге Application Data\Microsoft\Protect и зашифровывается по алгоритму 3DES с помощью ключа, который отчасти основан на пароле пользователя. Когда пользователь меняет свой пароль, мастер-ключи автоматически расшифровываются, а затем заново зашифровываются с учетом нового пароля.

Для расшифровки FEK EFS использует функции Microsoft CryptoAPl (CAPI). CryptoAPI

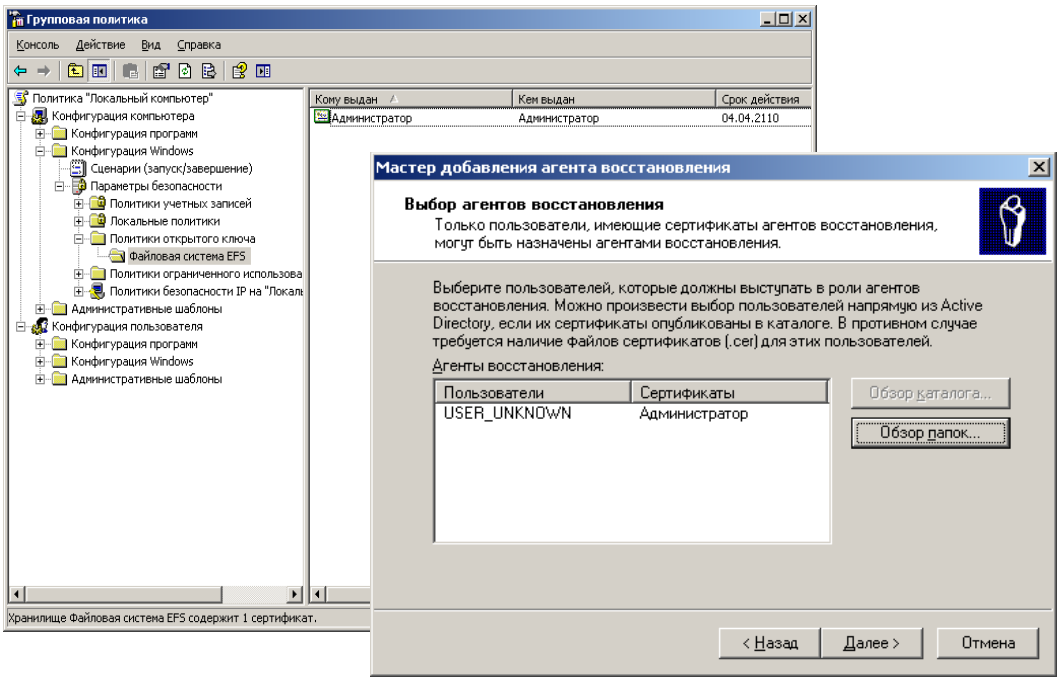


Рис. 1.12.

состоит из DLL провайдеров криптографических сервисов (cryptographic service providers, CSP), которые обеспечивают приложениям доступ к различным криптографическим сервисам (шифрованию, дешифрованию и хэшированию). EFS опирается на алгоритмы шифрования RSA, предоставляемые провайдером **Microsoft Enhanced Cryptographic Provider (\Windows\ System32\Rsaenh.dll).**

Шифрацию и дешифрацию файлов можно осуществлять программно, используя API-функции **EncryptFile** и **DecryptFile**.

2. Порядок выполнения работы.

2.1. Ознакомьтесь с теоретическими основами защиты информации в ОС семейства Windows в настоящих указаниях.

2.2. Выполните задания 2.2.1-2.2.8

2.2.1.Запустите в программе **Oracle VM Virtualbox** виртуальную машину WinXP. Войдите в систему под учетной записью администратора, пароль узнайте у преподавателя. Все действия в пп 2.2.1-2.2.8 выполняйте в системе, работающей на виртуальной машине.

2.2.2. Создайте учетную запись нового пользователя **testUser** 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 **”testGroup”** и включите в нее нового пользователя. Удалите пользователя из других групп. Создайте на диске **C:** папку **forTesting**. Создайте или скопируйте в эту папку несколько текстовых файлов (*.txt).

2.2.3.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. Строку запуска и результат работы этой и ***всех*** следующих консольных команд копируйте в файл протокола лабораторной работы.

2.2.4.Убедитесь в соответствии имени пользователя и полученного SID в реестре Windows. Найдите в реестре, какому пользователю в системе присвоен SID **S-1-5-21-1957994488-492894223-170857768-1004** (Используйте ключ реестра **HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList**).

2.2.5.Командой **whoami** определите перечень текущих привилегий пользователя **testUser**. В сеансе командной строки пользователя 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.**

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

2.2.6. Ознакомьтесь с справкой по консольной команде **cacls.** Используя эту команду, просмотрите разрешения на папку **c:\forTesting**. Объясните все обозначения в описаниях прав пользователей и групп в выдаче команды.

а) Разрешите пользователю **testUse**r запись в папку **forTesting**, но запретите запись для группы **testGroup**. Попробуйте записать файлы или папки в **forTesting** от имени пользователя **testUser**. Объясните результат. Посмотрите эффективные разрешения пользователя **testUser** к папке **forTesting** в окне свойств папки.

б) Используя стандартное окно свойств папки, задайте для пользователя **testUser** такие права доступа к папке, чтобы он мог записывать информацию в папку **forTesting**, но не мог просматривать ее содержимое. Проверьте, что папка **forTesting** является теперь для пользователя **testUser** “слепой”, запустив, например, от его имени файловый менеджер и попробовав записать файлы в папку, просмотреть ее содержимое, удалить файл из папки.

в) Для вложенной папки **forTesting\Docs** отмените наследование ACL от родителя и разрешите пользователю просмотр, чтение и запись в папку. Проверьте, что для пользователя папка **forTesting\Docs** перестала быть “слепой” (например, сделайте ее текущей в сеансе работы файлового менеджера от имени пользователя и создайте в ней новый файл).

г) Снимите запрет на чтение папки **forTesting** для пользователя **testUser**. Используя команду **cacls** запретите этому пользователю доступ к файлам с расширением txt в папке **forTesting**. Убедитесь в недоступности файлов для пользователя.

д) Командой **cacls** запретите пользователю все права на доступ к папке **forTesting** и разреште полный доступ к вложенной папке **forTesting\Docs**. Убедитесь в доступности папки **forTesting\Docs** для пользователя. Удалите у пользователя **testUser** привилегию **SeChangeNotifyPrivilege**. Попробуйте получить доступ к папке **forTesting\Docs**. Объясните результат.

е) Запустите файловый менеджер от имени пользователя **testUser** и создайте в нем папку **newFolder** на диске C. Для папки **newFolder** очистите весь список ACL командой **cacls.** Попробуйте теперь получить доступ к папке от имени администратора и от имени пользователя. Кто и как теперь может вернуть доступ к папке? Верните полный доступ к папке для всех пользователей.

ж) Создайте в разделе **HKLM\Software** реестра раздел **testKey**. Запретите пользователю **testUser** создание новых разделов в этом разделе реестра. Создайте для раздела **HKLM\Software\testKey** SACL, позволяющий протоколировать отказы при создании новых подразделов, а также успехи при перечислении подразделов и запросе значений (предварительно проверьте, что в локальной политике безопасности соответствующий тип аудита включен). Попробуйте от имени пользователя **testUser** запустить **regedit.exe** и создать раздел в **HKLM\Software**. Убедитесь, что записи аудита были размещены в журнале безопасности (**eventvwr.msc**).

2.2.7.Шифрование файлов и папок средствами EFS.

а) От имени пользователя **testUser** зашифруйте какой-нибудь файл на диске. Убедитесь, что после этого был создан сертификат пользователя, запустив оснастку **certmgr.msc** от имени пользователя (раздел **Личные**). Просмотрите основные параметры сертификата открытого ключа пользователя **testUser** (срок действия, используемые алгоритмы). Установите доверие к этому сертификату в вашей системе.

б) Создайте в папке **forTesting** новую папку **Encrypt**. В папке **Encrypt** создайте или скопируйте в нее текстовый файл. Зашифруйте папку **Encrypt** и все ее содержимое из меню свойств папки от имени администратора. Попробуйте просмотреть или скопировать какой-нибудь файл этой папки от имени пользователя **testUser**. Объясните результат. Скопируйте зашифрованный файл в незашифрованную папку (например, **forTesting**). Убедитесь что он остался зашифрованным. Добавьте пользователя **testUser** в список имеющих доступа к файлу пользователей в окне свойств шифрования файла. Повторите попытку получить доступ к файлу от имени пользователя **testUser.**

в) Создайте учетную запись нового пользователя **agentUser,** сделайте его членом группы Администраторы. Определите для пользователя **agentUser** роль агента восстановления EFS. Создайте в папке **forTesting** новый текстовый файл с произвольным содержимым. Зашифруйте этот файл от имени пользователя **testUser**. Убедитесь в окне подробностей шифрования файла, что пользователь **agentUser** является агентом восстановления для данного файла. Попробуйте прочитать содержимое файла от имени администратора и от имени пользователя **agentUser.** Объясните результат.

г) Зашифруйте все текстовые файлы папки **forTesting** с использованием консольной команды шифрования **cipher** от имени пользователя **testUser (**предварительно снимите запрет на доступ к этим файлам, установленный в задании 2.2.6г**).**

д) Убедитесь, что при копировании зашифрованных файлов на том с файловой системой, не поддерживающей EFS (например, FAT32 на флеш-накопителе), содержимое файла дешифруется.

2.2.8. После демонстрации результатов работы преподавателю восстановите исходное состояние системы: удалите созданные папки и файлы, разделы реестра, удалите учетную запись созданного пользователя и его группы, снимите с пользователя **agentUse**r роль агента восстановления.

2.2.9. Представьте отчёт по лабораторной работе преподавателю и отчитайте работу.

3. Содержание отчета

1) титульный лист;
2) формулировку цели работы;
3) описание результатов выполнения;
4) выводы, согласованные с целью работы.

4. Контрольные вопросы

1. К какому классу безопасности относится ОС Windows по различным критериям оценки?
2. Каким образом пользователи идентифицируются в ОС Windows?
3. Что такое списки DACL и SACL?
4. Перечислите, каким образом можно запустить процесс от имени другого пользователя.
5. Как происходит проверка прав доступа пользователя к ресурсам в ОС Windows?
6. Что такое маркер безопасности, и какова его роль в модели безопасности Windows?
7. Как с использованием команды cacls добавить права на запись для всех файлов заданной папки?
8. Какие события подлежат аудиту в ОС Windows?
9. Каким образом шифруются файлы в файловой системе EFS? Что такое FEK? DDF? DDR?
10. Какие алгоритмы шифрования используются в EFS?

ЛАБАРАТОРНАЯ РАБОТА №2.

Механизмы безопасности ОС GNU/ Linux

Цель лабораторной работы
изучение основных возможностей по обеспечению безопасности информационных систем

получение опыта работы с основными утилитами, обеспечивающими различные аспекты безопасности информационные систем в операционной системе Linux

1. Основные сведения

В данной лабораторной работе будет проведен краткий экскурс в наиболее распространенные средства, связанные с безопасностью Linux Информация предоставлена в сжатом виде, и если какое-то средство вас заинтересует, можно пройтись по ссылкам и прочитать более подробно. Будут рассмотрены следующие средства: ssh, sudo, firewall (iptables).

sudo: выполнение программ от чужого имени

Описание: Выполнение программ от своего и/или чу жого имени.

Механизм работы: При вызове команды sudo/sudoedit система считывает файл /etc/sudoers, и на его основе определяет, какие команды может вызывать пользователь.

Пример использования: Вся конфигурация определяется в файле /etc/sudoers. Например, можно разрешать выполнять только определенные команды и только от определенного пользователя:

WEBMASTERS www = (www) ALL, (root) /usr/bin/su www

Данная строчка говорит о том, что пользователи, определенные в алиасе *WEBMASTERS* могут выполнять все команды от имени пользователя *www,* или делать *su* только в *WWW.*

Пакет sudo позволяет системному администратору давать права определенным пользователям (или группам) на исполнение конкретных программ с правами другого пользователя (и записывать эти действия в журнал). Возможна привязка списка допустимых ко манд к имени хоста, что позволяет использовать один файл настройки на нескольких хостах с различными полномочиями. Обычно требует аутентификации пользователя (например, ввода пароля). Позволяет избегать слишком частого ввода пароля (по умолчанию - 5 минут). Для борьбы с подменой динамических библио тек из окружения удаляются переменные типа LD_* и т.п., а также IFS, ENV, BASH_ENV, KRB_CONF, KRB5_CONFIG, LOCALDOMAIN, RES_OFTIONS, HOSTALIASES. Можно также удалять текущую директорию из PATH.

Состоит из файла настройки *etc sudoers*, программы его редактирования visudo и клиентской программы sudo. В лабораторной работе также описывается процедура установки из исходных текстов и ключи configure.

visudo

visudo позволяет осуществить безопасное редактирование sudoers, она проверяет файл на корректность после выхода из текстового редактора. Ключи:

- -c [-q] (только проверить существующий файл без вызова редактора)
- -f имя-файла (указать другой файл sudoers)
- -s (более строгая проверка)

Описание прав пользователей

В файле /etc/sudoers описываются права пользователей на выполнение команд с помощью sudo. Состоит из операторов трех типов: определение синонимов (Alias), переопределение конфигурационных параметров и описание прав пользователей. Если для одного пользователя подходит несколько описаний, то действует последнее. Комментарии начинаются с символа '#', если это не uid. Продолжение команды на следующей строке обозначается символом '\' в конце строки.

Синонимы (предопределен синоним ALL):

- User_Alias имя = список-пользователей-через-запятую
- Runas_Alias имя = список-пользователей-через-запятую
- Host_Alias имя = список-хостов-через-запятую
- Cmnd_Alias имя = список-команд-через-запятую

Имя может состоять из прописных латинских букв, цифр и подчеркивания. Предопределены синонимы 'ALL' для каждого типа. На одной строке может размещать несколько описаний синонимов, разделяя их символом ':'. netgroup - это NIS.

Пользователь может определяться (восклицательный знак перед именем означает отрицание):

- *имя*
- *%имя-группы*
- *+netgroup*
- *User_Alias*

Эффективный пользователь может определяться (восклицательный знак перед именем означает отрицание):

- *имя*
- *#uid*
- *%имя-группы*
- *+netgroup*
- *Runas_Alias*

Хост может определяться (восклицательный знак перед именем означает отрицание; сетевая маска записывается в точечной записи или CIDR; при указании имени хоста можно использовать шаблоны в стиле shell, но лучше при этом использовать опцию fqdn; не стоит использовать имя localhost):

- *имя-хоста*
- *IP-адрес*
- *network/netmask*
- *+netgroup*
- *Host_Alias*

Команда может определяться (восклицательный знак перед именем означает отрицание; можно использовать шаблоны в стиле shell для имени файла и аргументов; необходимо защищать обратным слешом от интерпретации разборщика символы запятой, двоеточия, равенства и обратного слеша; пустой список аргументов обозначается как "\"; полное имя каталога должно оканчиваться на "/", разрешает выполнить любую команду из данного каталога, но не из подкаталога):

- *полное-имя-файла*
- *полное-имя-файла аргументы*
- *каталог*
- *Cmnd_Alias*

Конфигурационные параметры можно переопределить для всех пользователей, для определенных пользователей, для определенных хостов, для команд, выполняемых от имени указанного пользователя:

- Defaults список-параметров-через-запятую
- Defaults:имя-пользователя список-параметров-через-запятую
- Defaults@имя-хоста список-параметров-через-запятую
- Defaults>имя-пользователя список-параметров-через-запятую

Параметр задаётся в виде "имя = значение", "имя += значение", "имя -= значение", "имя" или "!имя" из следующего списка (необходимо использовать символы "" и '\' в значениях при необходимости; += и -= добавляют и удаляют элементы списка):

- флаги
  - authenticate (on; пользователь должен вводить свой пароль)
  - env_editor (off; visudo будет использовать переменную окружения VISUAL или EDITOR; очень не рекомендуется)
  - env_reset (сбросить переменные окружения, кроме HOME, LOGNAME, PATH, SHELL, TERM, USER)
  - fqdn (off; помещать в журнал и sudoers полное доменное имя хоста с использованием запросов DNS вместо простого `hostname`)
  - ignore_dot (off; игнорировать "." в $PATH; не реализовано)
  - log_host (off; записывать имя хоста в файл)
  - log_year (off; записывать в файл год в четырёхзначном формате

- mail_always (off: посылать письмо при каждом выполнении sudo)
- mail_badpass (off: посылать письмо при вводе неправильного пароля)
- mail_no_user (on: посылать письмо, если пользователя нет в sudoers)
- mail_no_host (off: пользователь есть в sudoers, но не для этого хоста)
- mail_no_perms (off: пользователь есть в sudoers, но нет прав на эту команду)
- noexec (off:)
- path_info (off: уточнять причину отказа выполнения команды)
- preserve_groups (off: не менять список групп)
- runaspw (off: запрашивать пароль указанного в runas_default пользователя вместо пароля текущего пользователя)
- requiretty (off: выполнять команду только при запуске с реального tty)
- root_sudo (on: очень рекомендуется запретить использование sudo для root)
- rootpw (off: запрашивать пароль root вместо пароля пользователя)
- shell_noargs (off: разрешать запуск sudo без параметров - выход в shell)
- targetpw (off: запрашивать пароль указанного ключом -u пользователя вместо пароля текущего пользователя)
- tty_tickets (off: отдельный интервал истечения времени действия пароля для каждого терминала)

- числа
  - loglinelen (80: длина строк в файле; 0 - бесконечная)
  - timestamp_timeout (5 минут: интервал действия пароля)
  - umask (022: umask для выполняемой команды)

- строки
  - editor (/bin/vi: список редакторов через ":", используемых visudo)
  - exempt_group (: пользователи из данной группы не должны вводить пароль)
  - lecture (закатывать лекцию при первом вызове: never, once, always: раньше это был флаг)
  - lecture_file
  - listpw (any: запрашивать ли пароль при использовании ключа "-l": all, any, never, always)
  - logfile (/var/log/sudo.log)
  - mailerflags (-t)
  - mailerpath
  - mailsub ("*** SECURITY information for %h ***": текст темы письма)
  - mailto (root: кому посылать письма: рекомендуется заключать в кавычки)
  - passprompt ("Password:" (%h расширяется в имя хоста, %u - в имя пользователя)
  - runas_default (root)
  - syslog (local2: syslog facilities: authpriv, auth, daemon, user, local0-7)
  - syslog_badpri (alert: какой приоритет syslog использовать при неудачах)
  - syslog_goodpri (notice: какой приоритет syslog использовать при удачном выполнении)

- timestampdir (/var/run/sudo)
- verifypw (all; запрашивать ли пароль при использовании ключа "-v": all, any, never, always)

- список
  - env_check (переменные окружения удаляются, если содержат '%' или '/')
  - env_delete
  - env_keep (список переменных, не сбрасываемых по env_reset)

Описание прав пользователей (списки через запятую, теги и имена целевых пользователей наследуются):

список-пользователей список-хостов = список-описаний-команд [: список-хостов = список-описаний-команд]

описание-команды ::= [(*список-целевых-пользователей*)] {*тег:*} *команда*

Теги: NOPASSWD (не запрашивать пароль текущего пользователя), PASSWD, NOEXEC, EXEC.

Команда sudo

Команда sudo выполняет указанную в качестве параметра команду от имени другого пользователя в зависимости от настроек. При отсутствии команды от имени другого пользователя выполняется редактор (-e) или командная оболочка (-s) или имитируется начальный вход в систему (-i). Устанавливается требуемый реальный идентификатор пользователя и группы, euid, egid и список групп. Может запрашиваться пароль текущего пользователя.

Ключи:

- -H (установить HOME как описано в /etc/passwd для целевого пользователя)
- -L (показать список возможных параметров с описанием)
- -P (не устанавливать список групп)
- -S (читать пароль с stdin вместо tty)
- -V (показать версию sudo; для root выдаются также параметры установки)
- -k (обнулить допустимое время использования sudo без повторного введения пароля)
- -K (аналогично, но более действенным способом)
- -b (выполнить команду в фоновом режиме)
- -e имя-файла (редактировать указанный файл, в настройках должно быть разрешено выполнять команду sudoedit)
- -h (вывести описание команды)
- -i (имитируется начальный вход в систему)
- -l (показать список разрешенных команд)
- -p формат (задаёт текст приглашения при вводе пароля)
- -s (запустить командную оболочку)

- -u имя-пользователя (с правами какого пользователя исполнить команду; по умолчанию - root)
- -u #uid
- -v (продлевает допустимое время использования sudo без повторного введения пароля на очередные 5 минут)
- -- (не интерпретировать остальные параметры)

По умолчанию, простая аккредитация пользователя означает, что при попытке исполнить команду от имени другого ему будет предложено ввести свой пароль - свой собственный, обратите внимание, а не пароль того, от чьего имени он собирается действовать. Вдобавок, "эффект памяти" после первого ввода пароля составляет по тому же умолчанию всего пять минут, и если команду sudo вновь отдать по истечении этого срока, она снова затребует пароль. Однако изменением настроек в файле /etc/sudoers можно преодолеть эти ограничения, уместные для больших многопользовательских систем, но никак не для домашних компьютеров.

Кстати, если sudo попытается воспользоваться неаккредитованный пользователь, сообщение об этом в виде электронного письма незамедлительно отправится на локальный адрес root, и если даже машина не подключена к Интернету, суперпользователь получит это письмо. Кроме того, запись о нарушителе появится в системном журнале.

Чтобы определить, какими полномочиями наделён тот или иной пользователь для исполнения команды sudo, достаточно от его имени сделать запрос sudo -l

```
[root@nefertem root]#
[root@nefertem root]# sudo -l
User root may run the following commands on this host:
    (ALL) ALL
[root@nefertem root]#
[root@nefertem root]#
```

Изначально только root имеет право вершить при помощи sudo всё, что угодно, - но суперпользователю эта утилита ни к чему, он и так способен на всё. Владельцам же прочих учётных записей на данном компьютере следует позаботиться о своей аккредитации. Заведение в */etc/sudoers* новых правил следует производить - внимание! - не при помощи привычного вам редактора, а посредством особой утилиты *visudo*. Её аналоги, кстати, имеются и для редактирования файлов паролей */etc/passwd* и групп */etc/group* - *vipw* и *vigr*, соответственно. Главная их особенность - в том, что перед запуском самого редактора они блокируют (lock) редактируемый файл, так что ни один другой пользователь не сможет в то же самое время его править. Vi ведь оперирует не с самим файлом, а с его копией в памяти, как мы помним, и до самого момента явной записи информации на диск в исходный файл не вносятся изменения. Значит, возможна ситуация, когда один и тот же файл открывают на редактирование два пользователя (в случае */etc/sudo*, например, им обоим должнен быть известен пароль *root*). Тогда в итоге файл останется таким, каким сохранит его на диск последний из завершивших работу пользователей. Если же файл блокирован, открыть его кому-то ещё в то же самое время будет непросто.

Утилита sudo - мощный и гибкий инструмент, и с его помощью системный администратор может существенно облегчить свою жизнь. Скажем, можно аккредитовать в */etc/sudoers* своего заместителя для выполнения некоторых рутинных обязанностей и не терзаться мыслью о том, что пароль root'а знает кто-то ещё: теперь это ни к чему. Однако вряд ли для домашнего компьютера потребуется вся потаённая мощь *sudo* - всё-таки уровень подозрительности к окружающим, если это члены семьи, даже у самого параноидального сисадмина должен быть пониже. Так что наиболее распространённая опция в sudo "для дома, для семьи" - это разрешение на запуск той или иной полезной утилиты от имени непривилегированного пользователя без дополнительного введения пароля.

Аккредитация в этом случае выглядит чрезвычайно просто (если вас интересуют более глубокие подробности, к вашим услугам man sudo и man 5 sudoers). Запустите команду visudo, и в самой последней строке открывшегося файла допишите строчку:

[имя пользователя] localhost.localdomain = NOPASSWD: [полный путь исполнения]

Все множество средств обеспечения безопасности можно разделить на следующие группы или категории:

- Средства управления доступом к системе (доступ с консоли, доступ по сети) и разграничения доступа
- Обеспечение контроля целостности и неизменности программного обеспечения (сюда же я отношу средства антивирусной защиты, поскольку внедрение вируса есть изменение ПО)
- Средства криптографической защиты
- Средства защиты от вторжения извне (внешнего воздействия)
- Средства протоколирования действий пользователей, которые тоже служат обеспечению безопасности (хотя и не только)
- Средства обнаружения вторжений
- Средства контроля состояния безопасности системы (обнаружения уязвимостей)

Как перезагрузить компьютер, находясь за пару тысяч километров от него или выполнить другие действия удаленно и безопасно? Ответ на эти вопросы - протокол SSH.

SSH (Secure Shell) - это сетевой протокол, используемый для удаленного управления компьютером и для передачи файлов. SSH похож по функциональности на протоколы telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли.

SSH позволяет передавать через безопасный канал любой другой сетевой протокол, таким образом, можно не только удаленно работать на компьютере, но и передавать по шифрованному каналу звуковой поток или видео (например, с веб-камеры). Также, SSH может использовать сжатие передаваемых данных для последующей их шифрации.

Большинство хостинг-провайдеров за определенную плату предоставляют клиентам доступ к их домашнему каталогу по SSH. Это очень удобно как для работы в командной строке, так и для удаленного запуска графических приложений. Через SSH можно работать и в локальной сети. Если вам лень ходить к серверам - администрируйте их удаленно, используя любой SSH-клиент.

SSH-клиенты и SSH-сервера написаны под большинство платформ: Windows, Linux, Mac OS X, Java, Solaris, Symbian OS, Windows Mobile и даже Palm OS.

Первая версия протокола, SSH-1, была разработана в 1995 году исследователем Tatu Ylonen из Технологического университета Хельсинки, Финляндия. SSH-1 был написан для обеспечения большей конфиденциальности, чем протоколы rlogin, telnet и rsh. В 1996 году была разработана более безопасная версия протокола, SSH-2, уже несовместимая с SSH-1. Протокол приобрел еще большую популярность, и к 2000 году его использовало уже порядка двух миллионов пользователей. В 2006 году протокол был утвержден рабочей группой IETF в качестве Интернет- стандарта.

Однако, до сих пор в некоторых странах (Франция, Россия, Ирак и Пакистан) требуется специальное разрешение в соответствующих структурах для использования определенных методов шифрования, включая SSH. См. закон Российской Федерации "О федеральных органах правительственной связи и информации".

Распространены две реализации SSH: коммерческая (закрытая) и свободная (открытая). Свободная реализация называется OpenSSH. К 2006 году 80% компьютеров Интернет использовало именно OpenSSH. Коммерческая реализация разрабатывается организацией SSH Inc., http://ssh.com/ - закрытая реализация, бесплатная для некоммерческого использования. Свободная и коммерческая реализации SSH содержат практически одинаковый набор команд.

Существуют две версии протокола SSH: SSH-1 и SSH-2. В первой версии протокола есть существенные недостатки, поэтому в настоящее время SSH-1 практически нигде не применяется.

Многие взломщики сканируют сеть в поиске открытого порта SSH, особенно - адреса хостинг-провайдеров. Так что, в целях безопасности - запрещайте доступ по ssh для суперпользователя. Обычно злоумышленники подбирают именно пароль суперпользователя. См. ниже рекомендации по безопасности использования SSH.

Протокол SSH-2 устойчив в атакам "man-in-middle", в отличие от протокола telnet. То есть, прослушивание трафика, "снифинг", ничего не дает злоумышленнику. Протокол SSH-2 также устойчив к атакам путем присоединения посредине (session hijacking) и обманом сервера имен (DNS spoffing).

Далее по тексту, мы будем иметь ввиду под SSH вторую версию протокола, SSH-2.

SSH-сервера

- Debian GNU/Linux: dropbear, lsh-server, openssh-server, ssh
- MS Windows: freeSSHd, OpenSSH sshd, WinSSHD, ProSHHD, Dropbear SSH Server

SSH-клиенты и оболочки

- Debian GNU/Linux: kdessh, lsh-client, openssh-client, putty, ssh
- MS Windows: PuTTY, SecureCRT, ShellGuard, Axessh, ZOC, SSHWindows, ProSSHD
- Mac OS: Nifty Telnet SSH
- Symbian OS: PuTTY
- Java: MindTerm, AppGate Security Server

Для работы по SSH нужен SSH-сервер и SSH-клиент. Сервер прослушивает соединения от клиентских машин и при установлении связи производит аутентификацию, после чего начинает обслуживание клиента. Клиент используется для входа на удаленную машину и выполнения команд.

Предположим, что сервер у нас настроен и работает. Для подключения клиента требуется сгенерировать пару из открытого и закрытого ключей. Если Вы используете PuTTY под Windows - это делается утилитой puttygen.exe.

Под Linux обычно используется команда puttygen (для PuTTY) или ssh-keygen (для OpenSSH). Далее указываем клиенту - где находится закрытый ключ, и соединяемся с вводом пароля. Возможно также беспарольное соединение, о чем упомянуто ниже.

Рекомендации по безопасности использования SSH:

1. Запрещение удаленного root-доступа.
2. Запрещение подключения с пустым паролем или отключение входа по паролю.
3. Выбор нестандартного порта для SSH-сервера.
4. Использование длинных SSH2 RSA-ключей (2048 бит и более). По состоянию на 2006 год система шифрования на основе RSA считалась надёжной, если длина ключа не менее 1024 бит.
5. Ограничение списка IP-адресов, с которых разрешен доступ. Например, настройкой файрвола.
6. Запрещение доступа с некоторых, потенциально опасных адресов.
7. Отказ от использования распространенных или широко известных системных логинов для доступа по SSH.
8. Регулярный просмотр сообщений об ошибках аутентификации.
9. Установка детекторов атак (IDS, Intrusion Detection System).
10. Использование ловушек, подделывающих SSH-сервис (honeypots)

Как подключиться к удаленному серверу из Linux или FreeBSD? Команда подключения к локальному SSH-серверу из командной строки для пользователя pacify (сервер слушает нестандартный порт 30000):

$ ssh -p30000 pacify@127.0.0.1

Под Windows можно воспользоваться программой PuTTY. Она имеет простой графический интерфейс, через который удобно настраивать подключения. На вкладке SSH\Auth надо выбрать закрытый ключ, который будет использоваться PuTTY.

К удаленному компьютеру можно подключиться по SSH, не вводя пароль, а используя открытый ключ. Разумеется, открытый ключ лучше передавать на сервер по защищенному каналу.

Генерация пары SSH-2 RSA-ключей длиной 4096 бита программой puttygen под Linux/UNIX надо выполнить команду:

$ puttygen -t rsa -b 4096 -o sample

Под Windows у этой программы (puttygen.exe) есть пользовательский интерфейс.

Установка SSH в Linux на примере Debian

Итак, всё, что нам нужно для установки полного комплекта удалённого управления компьютером (**SSH-клиент** и **SSH-сервер**) давным-давно лежит в репозитории. Лёгким движением ставим пакет:

 # apt-get install ssh

и ждём несколько мгновений, когда оно настроится. После этого мы получим возможность **SSH** доступа в систему и управления ей. Так как технология эта кросс-платформенная, то можно управлять по SSH Linux или FreeBSD можно и из Windows. Для этого есть **putty**, SSH Windows клиент.

На стороне клиента теперь надо поправить настройки, которые лежат в каталоге /etc/ssh - конфиг для клиента называется ssh-config, конфиг для сервера, соответственно, sshd-config. На своей, клиентской, стороне, настраиваем возможность приёма X11Forward, ищем и меняем ключи на:

ForwardX11 yes
ForwardX11Trusted yes

Клиентская машина теперь может запускать удалённо графические приложения на сервере. Настройка **SSH** на стороне клиента закончена, теперь идём к админу далёкого сервера. В принципе, можно на клиентской стороне ничего не менять, а логиниться на удалённую машину так:

$ ssh –X user@server1.mydomain.com

Или

$ ssh –X user@192.168.x.x

На стороне сервера теперь нужно настроить SSH сервер: в конфигах машины-сервера, к которой будем подсоединяться (у вас ведь есть её рутовый пароль, так ведь?) в /etc/ssh/sshd-config ищем и меняем ключи на:

X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes

Этим мы разрешаем серверу запускать удалённо графические приложения и отправлять их на клиентскую машину. Перестартуем сервис:

$sudo /etc/init.d/ssh restart

Теперь мы сможем заходить на машину не только в консольном режиме, но и с запуском иксовых приложений. Если хочется разрешить вход только с определённых машин, нужно подправить строки в конфиге */etc/ssh/sshd_config*

AllowUsers hacker@*

AllowUsers *@192.168.1.*

SSH в действии

Всё готово, и теперь будут приведены несколько команд SSH для примера. Открываем консольку и пишем в ней:

$ ssh
имя пользователя удалённой машины@ip адрес или сетевое имя удалённой машины

После этого нас могут спросить: данный айпишник ещё не опознан, как доверительный, стоит ему доверять? Пишем yes, стоит, конечно! Далее вводим пароль пользователя удалённой машины, под которым мы заходим и, если он правильный, попадаем в консоль ***удалённой*** машины. В процессе набора пароля вы его не увидите - набирайте всё равно: даётся три попытки - потом соединение обрывается.

Итак, нас поприветствуют как-то вроде этого:

penta4@penta4-tree:~$ ssh beast@192.168.1.5
Password:
Last login: Tue Oct 10 19:23:57 2006 from 192.168.1.1
beast@notebeast: $

Теперь, в окошке терминала, который на нашей машине, мы рулим компьютером, к которому мы подключились. Не перепутайте терминалы, а то вырубите не тот компьютер. Здесь всё просто и логично, но нам бы хотелось ещё и запускать графические приложения на удалённой системе.

Запуск графических приложений удалённо. Вводим, как обычно, логин и пароль ***удалённой*** машины. И вот перед нами та же самая консоль. Как вызвать графическое приложение? Просто наберите имя вызываемой программы и поставьте после имени знак амперсанд:

$ gimp &;

Это запустит на ***удалённо*** машине GiMP в фоне и вернёт вам консоль для дальнейших действий. Если вы не поставите амперсанд после имени приложения, то управление в консоль будет возвращено только после завершения приложения.

Принцип работы файрвол

Брандмауэр (файрвол) предназначен для фильтрации и обработки пакетов, проходящих через сеть. Когда пакет прибывает, брандмауэр анализирует заголовки пакета и принимает решение, "выбросить" пакет (DROP), принять пакет (ACCEPT, пакет может пройти дальше) или сделать с ним что-то еще более сложное.

В Linux брандмауэр является модулем ядра, его неотъемлемой частью. С его помощью мы можем делать с пакетами множество хитроумных вещей, но основной принцип манипуляции трафиком сохраняется: просматриваются заголовки пакетов и решается их дальнейшая судьба. Интерфейсом для модификации правил, по которым брандмауэр обрабатывает пакеты, служит iptables.

Итак, пребывающий пакет проходит по цепочке правил. Каждое правило содержит *условие* и *цель* (действие). Если пакет *удовлетворяет условию* то он *передается на цель*, в противном случае к пакету применяется следующее правило в цепочке. Если пакет не удовлетворил ни одному из условий в цепочке, то к нему применяется *действие по умолчанию*.

Например, к нам пришел пакет от адреса 192.168.164.84 для 128.3.4.5, применим цепочку из трех правил к этому пакету:

номер правила	**условие**	**действие**	**результат**
1	пакет от адреса 127.2.4.5	пропустить (ACCEPT)	переход к следующему правилу
2	пакет для адреса 234.2.*.5	пропустить (ACCEPT)	переход к следующему правилу
3	пакет от адреса 192.168.*	выбросить пакет (DROP)	пакет выброшен

Как видно, пакет не удовлетворил первым двум условиям цепочки, зато удовлетворил третьему правилу, которое заставило брандмауэр выбросить этот пакет.

Цепочки собраны в три основные таблицы (при желании можно добавить свои):

- **filter** - Таблица, используемая по-умолчанию,
- **nat** - Эта таблица используется, когда встречается пакет, устанавливающий новое соединение.

- **mangle** - Эта таблица используется для специальных изменений пакетов.

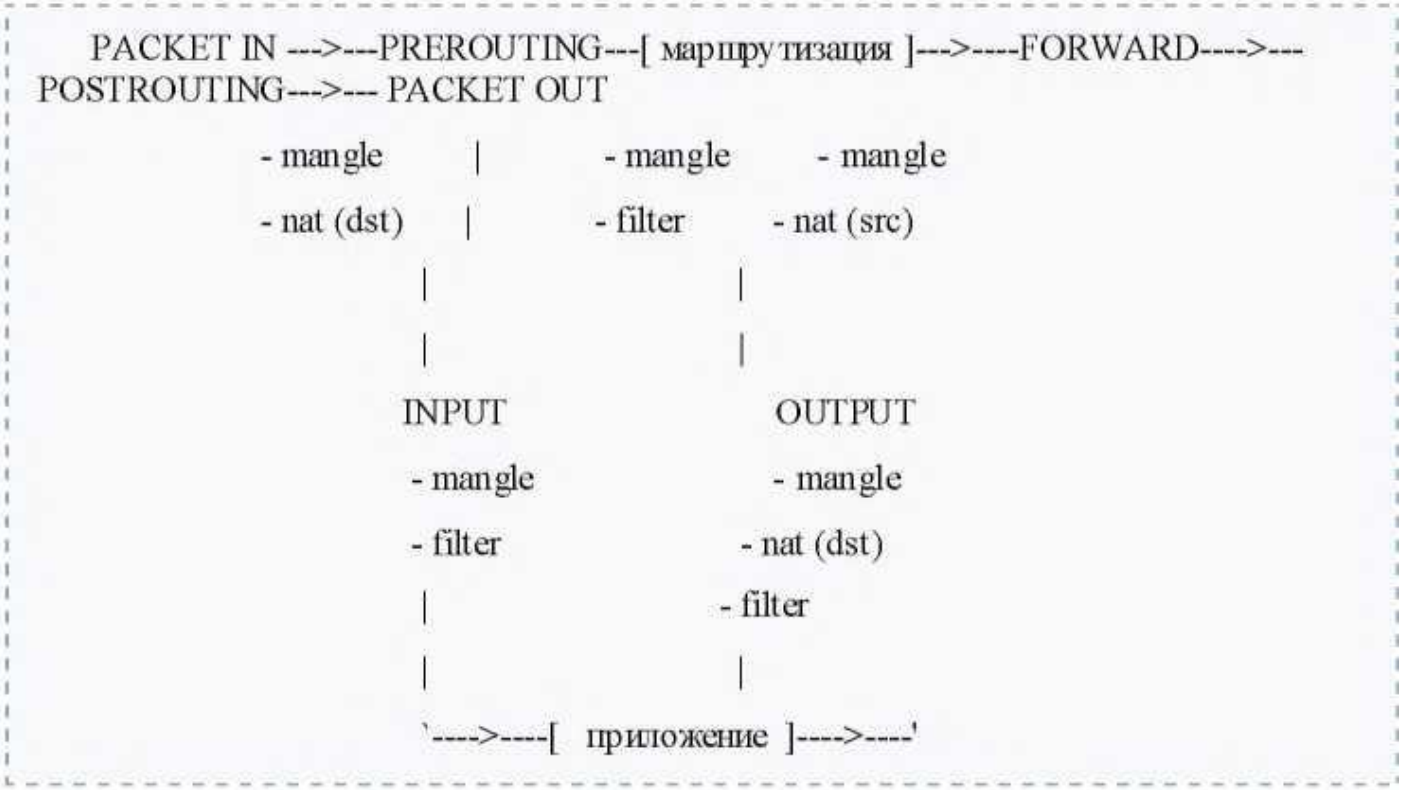
Основными цепочками являются следующие:

- **FORWARD**: для проходящих (пересылаемых) пакетов
- **INPUT**: для входящих/получаемых пакетов
- **OUTPUT**: для исходящих/отправляемых пакетов

Действие может быть именем цепочки, определенной пользователем, или одной из специальных целей: ACCEPT, DROP, QUEUE, RETURN или MASQUERADE.

ACCEPT означает принять пакет. DROP означет проигнорировать (выбросить) пакет. MASQUERADE означает скрыть (маскировать) IP. (ниже будет рассмотрено подробно)

Схематично обработку пакета можно изобразить следующим образом:



Входящий пакет начинает обрабатываться брандмауэром с цепочки PREROUTING в таблице **mangle**. Затем он обрабатывается правилами цепочки PREROUTING таблицы **nat**. На этом этапе проверяется, не требуется ли модификация назначения пакета (DNAT). Важно сменить назначение сейчас, потому что маршрут пакета определяется сразу после того, как он покинет цепочку PREROUTING. После этого он будет отправлен на цепочку INPUT (если целью пакета является этот компьютер) или FORWARD (если его целью является другой компьютер в сети).

Если целью пакета является другой компьютер, то пакет фильтруется правилами цепочки FORWARD таблиц **mangle** и **filter**, а затем к нему применяются правила цепочки POSTROUTING. На данном этапе можно использовать

SNAT/MASQUARADE (подмена источника/маскировка). После этих действий пакет (если выжил) будет отправлен в сеть

Если назначением пакета является сам компьютер с брандмауэром, то, после маршрутизации, он обрабатывается правилами цепочек INPUT таблиц **mangle** и **filter**. В случае прохождения цепочек пакет передается приложению.

Когда приложение, на машине с брандмауэром, отвечает на запрос или отправляет собственный пакет, то он обрабатывается цепочкой OUTPUT таблицы **filter**. Затем к нему применяются правила цепочки OUTPUT таблицы **nat**, для определения, требуется-ли использовать DNAT (модификация назначения), пакет фильтруется цепочкой OUTPUT таблицы **filter** и выпускается в цепочку POSTROUTING которая может использовать SNAT и QoS. В случае успешного прохождения POSTROUTING пакет выходит в сеть.

Для добавления правила в цепочку используется ключ **-A**

#iptables -A INPUT правило

добавит правило в цепочку INPUT таблицы **filter** (по умолчанию). Для указания таблицы, в цепочку которой следует добавить правило, используйте ключ **-t**:

#iptables -t nat -A INPUT правило

добавит правило в цепочку INPUT таблицы **nat**.

Цель по умолчанию задается с помощью ключа **-P**:

#iptables -P INPUT DROP

Условия для отбора пакетов

Теперь мы знаем как пакеты проходят сквозь различные таблицы и цепочки iptables. Пришло время разъяснить принципы, по которым строятся условия накладываемые на пакеты:

Немного о протоколе TCP/IP

TCP/IP является протоколом, в котором соединение устанавливается в 3 фазы. Если компьютер А пытается установить соединение с компьютером Б они обмениваются специальными TCP пакетами.

После чего соединение считается **установленным** (ESTABLISHED).

iptables различает эти состояния как NEW и ESTABLISHED.

Опции отбора пакетов

опция	**описание**	**пример**
—protocol (сокращено -	Определяет протокол. Опции tcp, udp, icmp, или любой другой протокол определенный	iptables -A INPUT —protocol tcp

p)	в /etc/protocols	
—source	IP адрес источника пакетаs. Может быть определен несколькими путями. • Одиночный хост: host.domain.tld, или IP адрес: 10.10.10.3 • Пул-адресов (подсеть): 10.10.10.3/24 или 10.10.10.3/255.255.255.0	iptables -A INPUT —source 10.10.10.3
—destination	IP адрес назначения пакета. Может быть определен несколькими путями - смотри —source	iptables -A INPUT —destination 192.168.1.0/24
—source-port	Порт источник, возможно только для протоколов —protocol tcp, или —protocol udp	iptables -A INPUT —protocol tcp —source-port 25
—destination-port	Порт назначения, возможно только для протоколов —protocol tcp, или —protocol udp	iptables -A INPUT —protocol udp —destination-port 67
—state	Состояние соединения. Доступно, если модуль `state` загружен с помощью `-m state`. Доступные опции: **NEW** Все пакеты устанавливающие новое соединение **ESTABLISHED** Все пакеты принадлежащие установленному соединению **RELATED** Пакеты, не принадлежащие установленному соединению, но связанные с ним. Например - FTP в активном режиме использует разные соединения для передачи данных. Эти соединения связаны. **INVALID**	iptables -A INPUT -m state —state NEW,ESTABLISHED

	Пакеты, которые не могут быть по тем или иным причинам идентифицированны. Например ICMP ошибки не принадлежащие существующим соединениям	
—in-interface (сокращенно -i)	Определяет интерфейс, на который прибыл пакет. Полезно для NAT и машин с несколькими сетевыми интерфейсами	iptables -t nat -A PREROUTING —in-interface eth0
—out-interface (сокращенно -o)	Определяет интерфейс, с которого уйдет пакет. Полезно для NAT и машин с несколькими сетевыми интерфейсами	iptables -t nat -A POSTROUTING —in-interface eth1
—tcp-flags	Определяет TCP флаги пакета. Содержит 2 параметра: Список флагов которые следует проверить и список флагов которые должны быть установлены	
—syn	Сокращение для '—tcp-flags SYN,RST,ACK SYN'. Поскольку проверяет TCP флаги, используется с —protocol tcp. В примере показан фильтр для пакетов с флагом NEW, но без флага SYN. Обычно такие пакеты должны быть выброшены (DROP).	iptables -A INPUT —protocol tcp ! —syn -m state —state NEW

Определение цели

Чтобы определить действие, которое брандмауэр выполнит, если пакет в одной из цепочек удовлетворяет **условию**, iptables использует цели, устанавливаемые с помощью ключа —jump (или просто -j).

Целью может быть любая другая цепочка, куда будет передан пакет, или одно из следующих действий:

- **ACCEPT** Пропускает удовлетворяющий условию пакет (пакет покидает данную цепочку и передается дальше).

- **DROP** Выбрасывает удовлетворяющий условию пакет (молча, как будто он и не приходил).

- **REJECT** Отклоняет пакет, сообщая об этом передавшему. Имеет дополнительный параметр '—reject-with' определяющий сообщение которое будет передано отправившему пакет, это может быть: ***icmp-net-unreachable***, ***icmp-host-***

unreachable, ***icmp-port-unreachable*** (по умолчанию), ***icmp-proto-unreachable***, ***icmp-net-prohibitedor*** и ***icmp-host-prohibited***. Сообщение (по умолчанию "порт недоступен") возвращется отправившему пакет компьютеру.

- **LOG** Заносит в журнал информацию о пакете. Полезно в комбинации с DROP и REJECT для отладки.

- **RETURN** Возвращает пакет в ту цепочку, из которой он прибыл.

- **SNAT** Применяет source NAT ко всем удовлетворяющим условию пакетам. Может использоваться только в цепочках POSTROUTING и OUTPUT таблицы **nat**.

- **DNAT** Применяет destination NAT ко всем удовлетворяющим условию пакетам. Может использоваться только в цепочке POSTROUTING таблицы **nat**.

- **MASQUERADE** Может применяться только в цепочке POSTROUTING таблицы **nat**. Используется при наличии соединения с динамическим IP. Похож на SNAT, однако имеет свойство забывать про все активные соединения при потере интерфейса. Это полезно при наличии соединения, на котором время от времени меняется IP-адрес, но при наличии постоянного IP это только доставит неудобства. В том числе поэтому рекомендуется для статических IP использовать SNAT.

Примеры

Настроим простейший шлюз для раздачи интернета для дома или малого офиса (SOHO). Допустим, что внутрь сети смотрит интерфейс **eth1** с ip адресом 192.168.1.1, а в интернет интерфейс **eth0**:

Разрешим пропуск трафика через шлюз (в противном случае трафик не проходит цепочку FORWARDING):

```
#sysctl -w net.ipv4.ip_forward="1"
```

Добавим правило, для маскировки ip в цепочку POSTROUTING таблицы **nat**

```
#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Теперь клиенты внутренней сети могут получить возможность доступа в интернет установив в качестве шлюза адрес 192.168.1.1 .

Теперь попробуем нечто более интересное: запретим возможность установки новых соединений с маршрутизатором из интернета, а также форвардинг из сетей отличных от 192.168.1.0/24 .

Установим политики по умолчанию для цепочек в DROP, запретив все соединения кроме тех, которые будут разрешены:

```
#iptables -P INPUT DROP
#iptables -P FORWARD DROP
#iptables -P OUTPUT DROP
```

Разрешим входящие соединения на маршрутизатор с внутренней сети (для управления)

#iptables -A INPUT -i eth1 --source 192.168.1.0/24 --match state --state NEW,ESTABLISHED -j ACCEPT

Разрешим маршрутизатору отвечать компьютерам во внутренней сети:

#iptables -A OUTPUT -o eth1 --destination 192.168.1.0/24 --match state --state NEW,ESTABLISHED -j ACCEPT

Разрешим перенаправление пакетов из внутренней сети во внешнюю для установки соединений и установленных соединений:

#iptables -A FORWARD -i eth1 --source 192.168.1.0/24 --destination 0.0.0.0/0 --match state --state NEW,ESTABLISHED -j ACCEPT

Разрешим перенаправление пакетов из интернета во внутреннюю сеть только для установленных соединений:

#iptables -A FORWARD -i eth0 --destination 192.168.1.0/24 --match state --state ESTABLISHED -j ACCEPT

Итого:

- Маршрутизатор разрешает клиентам внутренней сети устанавливать соединение с узлами в интернет.
- Маршрутизатор имеет возможность удаленного управления из внутренней сети и только из внутренней сети.
- Маршрутизатор блокирует все попытки установить новое соединение **из** сети интернет.
- Маршрутизатор не принимает пакеты из сети интернет сам, только *перенаправляет* пакеты во внутреннюю сеть и только принадлежащие *установленным* соединениям.

2. Порядок выполнения работы

1. Проверьте, установлена ли на Вашем рабочем месте утилита *sudo* и, при необходимости, инсталлируйте ее.
2. Отредактируйте файл с конфигурацией sudo, довив своего пользователя, и запустите команду *ifconfig* от своего имени.
3. Проверьте, какими правами наделен Ваш пользователь?
4. Установите серверный пакет Open SSH и настройте его
5. Попробуйте зайти по протоколу ssh на рабочее место соседнего компьютера и откройте свое место для соседа по лабораторной работе.
6. Сгенерируйте и установите открытый ключ для доступа, попробуйте беспарольный вход.
7. Установите правило, блокирующее пакеты от соседнего рабочего места и проверьте работоспособность фильтра. По окончании теста отмените это правило

3. Содержание отчета

1) титульный лист;
2) формулировку цели работы;
3) описание результатов выполнения;
4) выводы, согласованные с целью работы.

4. Список контрольных вопросов

1. При помощи каких основных утилит обеспечивается безопасность в операционной системе Linux?
2. При помощи какой утилиты производится выполнение программ от своего и/или чужого имени?
3. Какой файл отвечает за настройки прав для утилиты *sudo?*
4. Каковы основные отличия удаленного доступа *telnet* от сетевого протокола *ssh* ?
5. Назовите основные программные пакеты, реализующие *ssh* в различных операционных системах?
6. Какая утилита реализует механизм генерации открытого ключа *ssh?*
7. Каковы основные принципы работы файрвол?
8. Приведите примеры фильтрации пакетов при помощи *iptables*

.

ЛАБОРАТОРНАЯ РАБОТА №3 .

Изучение программной системы TrueCrypt для создания и использования шифруемого-на-лету тома (устройства хранения данных)

Цель: изучить основные возможности программы шифрования дисков TrueCrypt.

1. Общие сведения.

TrueCrypt — компьютерная программа для шифрования «на лету» (On-the-fly encryption) для 32- и 64-разрядных операционных систем семейств Microsoft Windows NT 5 и новее (GUI-интерфейс), Linux и Mac OS X. Она позволяет создавать виртуальный зашифрованный логический диск, хранящийся в виде файла. С помощью TrueCrypt также можно полностью шифровать раздел жёсткого диска или иного носителя информации, такой как флоппи-диск или USB флеш-память. Все сохранённые данные в томе TrueCrypt полностью шифруются, включая имена файлов и каталогов. Смонтированный том TrueCrypt подобен обычному логическому диску, поэтому с ним можно работать с помощью обычных утилит проверки и дефрагментации файловой системы.

Возможности TrueCrypt

TrueCrypt умеет создавать зашифрованный виртуальный диск:

1. в файловом контейнере, что позволяет легко работать с ним — переносить, копировать (в том числе на внешние устройства в виде файла), переименовывать или удалять;
2. в виде зашифрованного раздела диска, что делает работу более производительной и удобной, в версии 5.0 добавилась возможность шифровать системный раздел;
3. путём полного шифрования содержимого устройства, такого как USB флеш-память (устройства флоппи-диск не поддерживаются с версии 7.0).

В список поддерживаемых TrueCrypt 6.2 алгоритмов шифрования входят AES, Serpent и Twofish. Предыдущие версии программы также поддерживали алгоритмы с размером блока 64 бита (Тройной DES, Blowfish, CAST5) (включая версии 5.x, которая могла открывать, но не создавать разделы, защищённые этими алгоритмами). Кроме того, возможно использование каскадного шифрования различными шифрами, к примеру: AES+Twofish+Serpent.

Все алгоритмы шифрования используют режим XTS, который более безопасен, нежели режимы CBC и LRW для шифрования «на лету», применяющиеся в предыдущих версиях (работа с уже созданными шифроконтейнерами в этих форматах также возможна).

Программа позволяет выбрать одну из трёх хеш-функций: HMAC-RIPEMD-160, HMAC-Whirlpool, HMAC-SHA-512 для генерации ключей шифрования, соли и ключа заголовка.

Для доступа к зашифрованным данным можно применять пароль (ключевую фразу), ключевые файлы (один или несколько) или их комбинации. В качестве ключевых файлов можно использовать любые доступные файлы на локальных, сетевых, съёмных дисках (при этом используются первые 1,048,576 байт) и генерировать свои собственные ключевые файлы.

Одна из примечательных возможностей TrueCrypt — обеспечение двух уровней правдоподобного отрицания наличия зашифрованных данных, необходимого в случае вынужденного открытия пароля пользователем:

Создание скрытого тома, что позволяет задать второй пароль (и набор ключевых файлов) к обычному тому для доступа к данным, к которым невозможно получить доступ с основным паролем, при этом скрытый том может иметь любую файловую систему и располагается в неиспользованном пространстве основного тома.

Ни один том TrueCrypt не может быть идентифицирован (тома TrueCrypt невозможно отличить от набора случайных данных, то есть файл нельзя связать с TrueCrypt как с программой, его создавшей, ни в какой форме и рамках).

Другие возможности TrueCrypt:

- Переносимость, что позволяет запускать TrueCrypt без установки в операционной системе (необходимы права группы администраторов в NT).
- Поддержка создания зашифрованного динамического файла на дисках NTFS. Такие тома TrueCrypt увеличиваются в размере по мере накопления новых данных вплоть до указанного максимального размера. Однако использование подобной возможности несколько уменьшает производительность и безопасность системы.
- Шифрование системного физического либо логического диска для Microsoft Windows-систем с дозагрузочной аутентификацией. (TrueCrypt не способен выполнять шифрование GPT дисков, которые установлены на большинстве современных компьютеров, поэтому перед шифрованием их необходимо преобразовать в MBR (master boot record)). (Такая же функциональность встроена в Windows Vista (не всех редакций), Windows 7 (Корпоративная и Максимальная), Windows Server 2008, Windows 8.1 под именем BitLocker. Однако, BitLocker не предлагает функционала правдоподобного отрицания и имеет закрытый исходный код, что делает невозможной проверку на наличие уязвимостей или встроенных лазеек для обхода защиты. Своя система шифрования встроена в Mac OS X, она называется FileVault, и начиная с версии 2.0 может зашифровать весь диск. Как иBitLocker, FileVault не предлагает функционала правдоподобного отрицания и имеет закрытый исходный код, что делает невозможной проверку на наличие уязвимостей или встроенных лазеек для обхода защиты).
- Изменение паролей и ключевых файлов для тома без потери зашифрованных данных.
- Возможность резервного сохранения и восстановления заголовков томов (1024 байт).
    - Это может быть использовано для восстановления заголовка повреждённого файла, чтобы монтировать том после ошибки на аппаратном уровне, в результате которой повредился заголовок.
    - Восстановление старого заголовка также сбрасывает пароль тома на тот, который действовал для прежнего заголовка.
- Возможность назначать комбинации клавиш для монтирования/размонтирования разделов (в том числе и быстрого размонтирования со стиранием ключа в памяти, закрытием окна и очисткой истории), отображения и сокрытия окна (и значка) TrueCrypt.
- Возможность использовать TrueCrypt на компьютере с правами обычного пользователя (в том числе создавать и работать с контейнерами в файлах), правда, первоначальную установку программы должен сделать администратор.

2. Порядок выполнения работы.

- Ознакомьтесь с теоретическими основами шифрования дисков и TrueCrypt.

- Запустите в программе **Oracle VM Virtualbox** виртуальную машину Windows 7. Войдите в систему под учетной записью администратора, пароль узнайте у преподавателя. Все действия выполняйте в системе, работающей на виртуальной машине.

- Запустите установочный файл, дважды кликнув по нему левой кнопкой мыши (Расположение установочного файла узнайте у администратора).

- В открывшемся окне установите флажок "I accept and agree to be bound by the license terms" и нажмите кнопку "Accept".

- Далее будет предложено выбрать вариант установки программы - Install или Extract. Первый вариант официально устанавливает программу и создает ярлыки в меню "Пуск" и на Рабочем столе. Выберите пункт "Extract" - создание переносной копии пакета программы, который можно хранить на любом сменном носителе и переносить из одной директории в другую.

- Недостаток переносной версии TrueCrypt - это невозможность шифровать целиком разделы жесткого диска (включая тот, где располагается Windows). Однако не рекомендуется неопытным пользователям делать полную установку TrueCrypt через пункт "Install" и экспериментировать с шифрованием разделов жесткого диска. Ошибка программы или ваши неправильные действия могут повлечь потерю всей информации на жестком диске. Если же вы все-таки решите попробовать, то не забудьте перед этим создать резервную копию операционной системы.
- Итак, выберите пункт "Extract" и нажмите кнопку "Next".

- В следующем окне, нажав кнопку "Browse...", выберите пункт установки программы. Например, C:TrueCrypt. Если вы установите флажок "Open the destination location when finished", то после установки папка с программой откроется автоматически. Нажмите кнопку "Extract (рис. 3.1)".

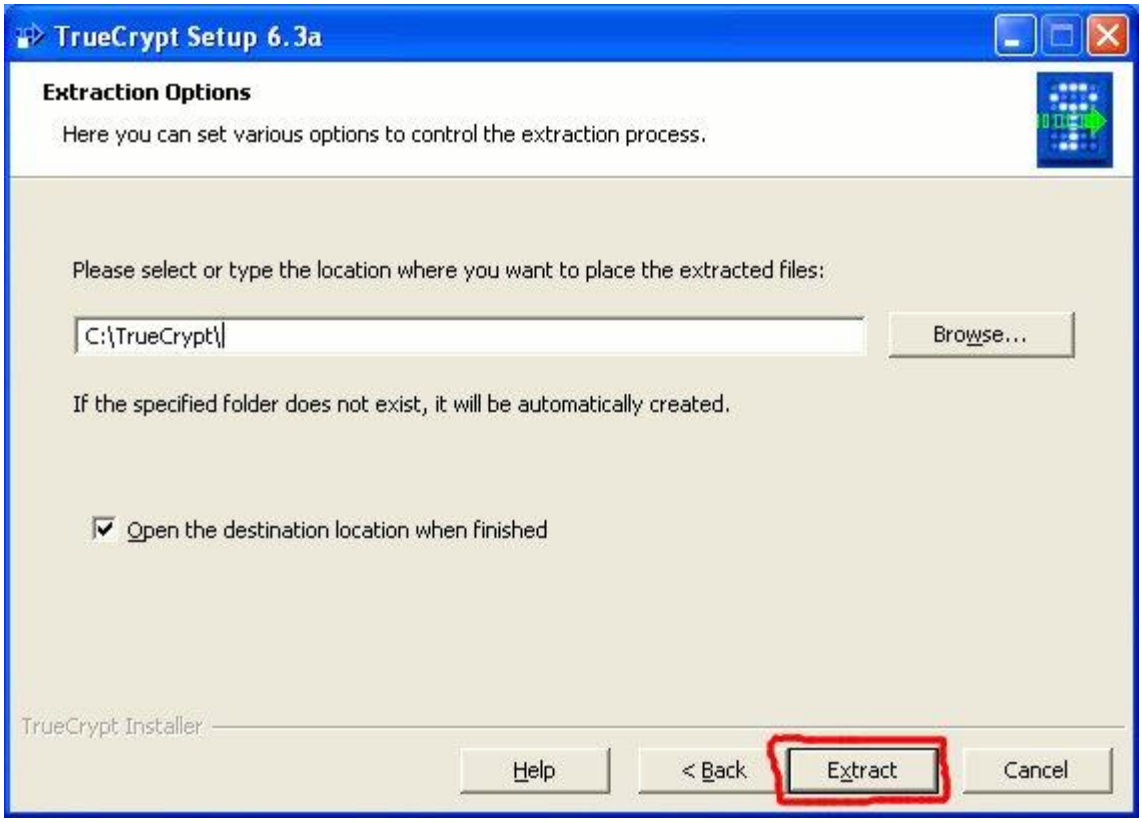


Рис. 3.1.

- Появится сообщение о том, что все файлы загружены в указанную директорию. Нажмите кнопку "OK".

- В окне установки программы нажмите кнопку "Finish".
- Установка завершена. Для работы запустите файл TrueCrypt.exe. Откроется основное окно программы (рис. 3.2).

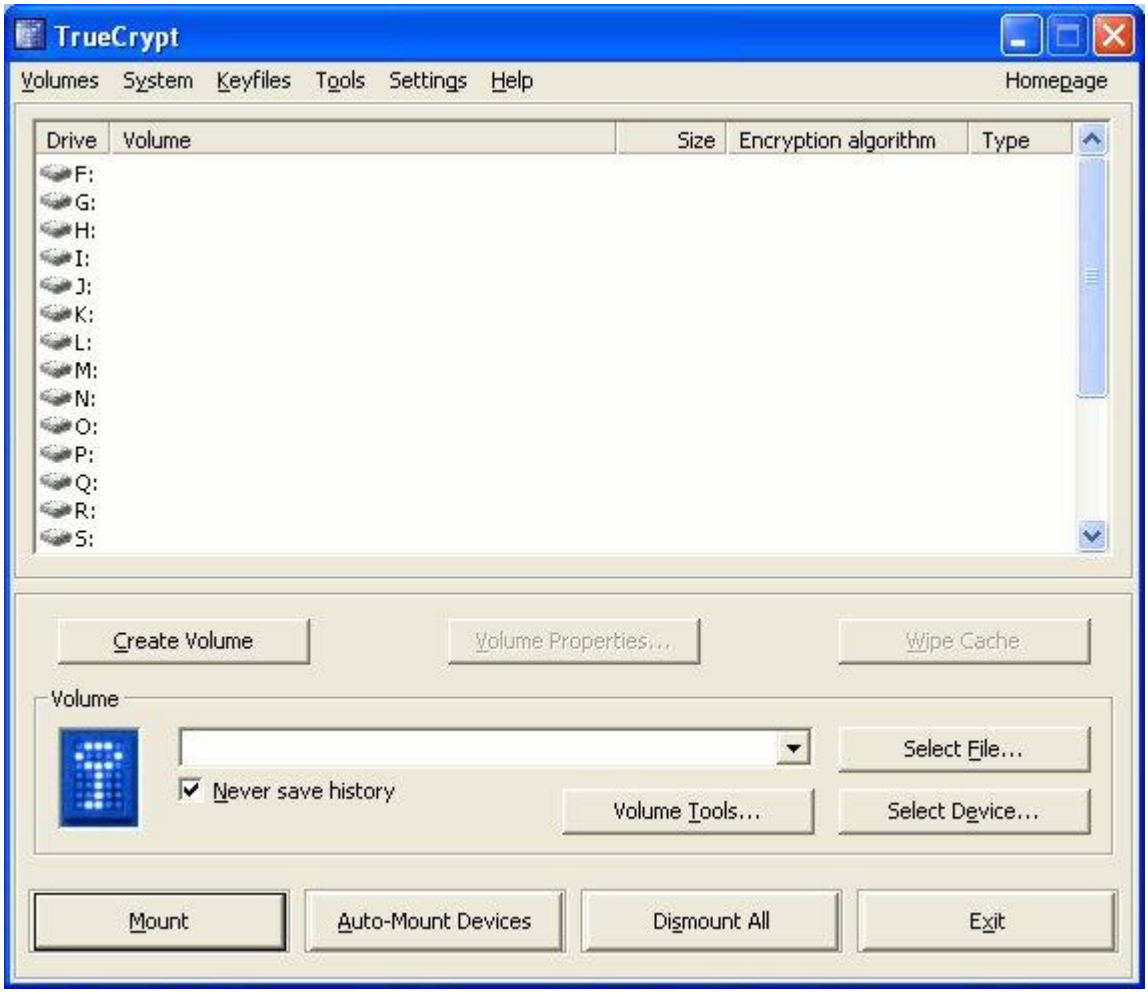


Рис.3.2.

Русификация TrueCrypt

- Из скачанного zip-архива, предназначенного для русификации программы, извлеките в папку, где располагается TrueCrypt, файл Language.ru.xml
- Запустите программу.
- В большинстве случаев ее интерфейс автоматически становится русскоязычным. В нашем примере так и есть.
- Если этого не произошло, пройдите в меню программы Settings - Language…
- В списке доступных языков должен появиться русский язык. Выберите его и нажмите кнопку "ОК"(рис. 3.3).

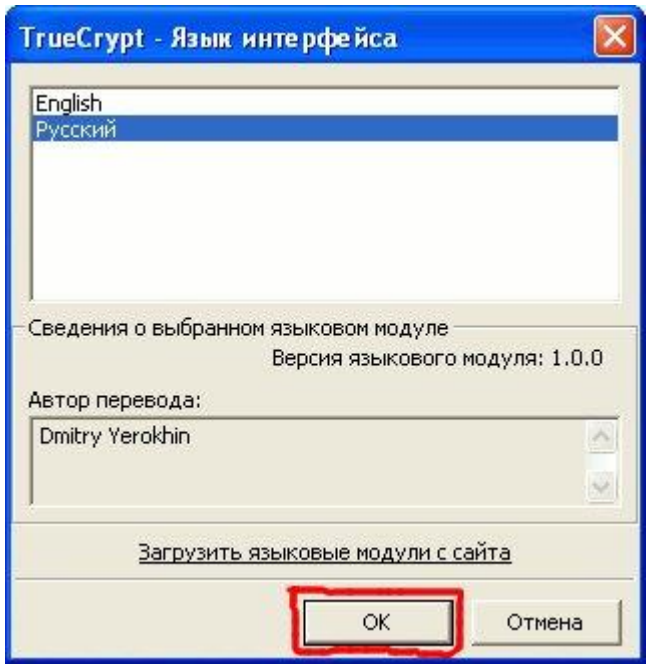


Рис. 3.3.

Теперь все диалоги программы будут на русском языке.

Настройка TrueCrypt

Для начала сконфигурируем TrueCrypt таким образом, чтобы можно было закрыть программу, размонтировать диски и очистить временный буфер моментально по сигналу тревоги. Для этого выполните следующие действия:

- Пройдите в меню программы Настройки - Горячие клавиши...
- В открывшемся окне выберите пункт "Сразу размонтировать все, очистить кэш и выйти".
- Установите курсор в поле "Клавиша" и нажмите на клавиатуре клавишу, которую вам удобно рассматривать как "горячую". В нашем примере это клавиша с буквой "Z". Вы также можете выбрать, какие клавиши использовать совместно с ней - Control, Shift, Alt, Win, установив или убрав флажок рядом с названием клавиши. Мы выбрали клавиши Control, Shift и Alt. Это означает, что при нажатии одновременно на клавиши Control, Shift, Alt и Z все TrueCrypt-диски будут размонтированы, временный буфер очищен и программа закроется.
- После выбора "горячих" клавиш нажмите кнопку "Назначить"(рис 3.4).

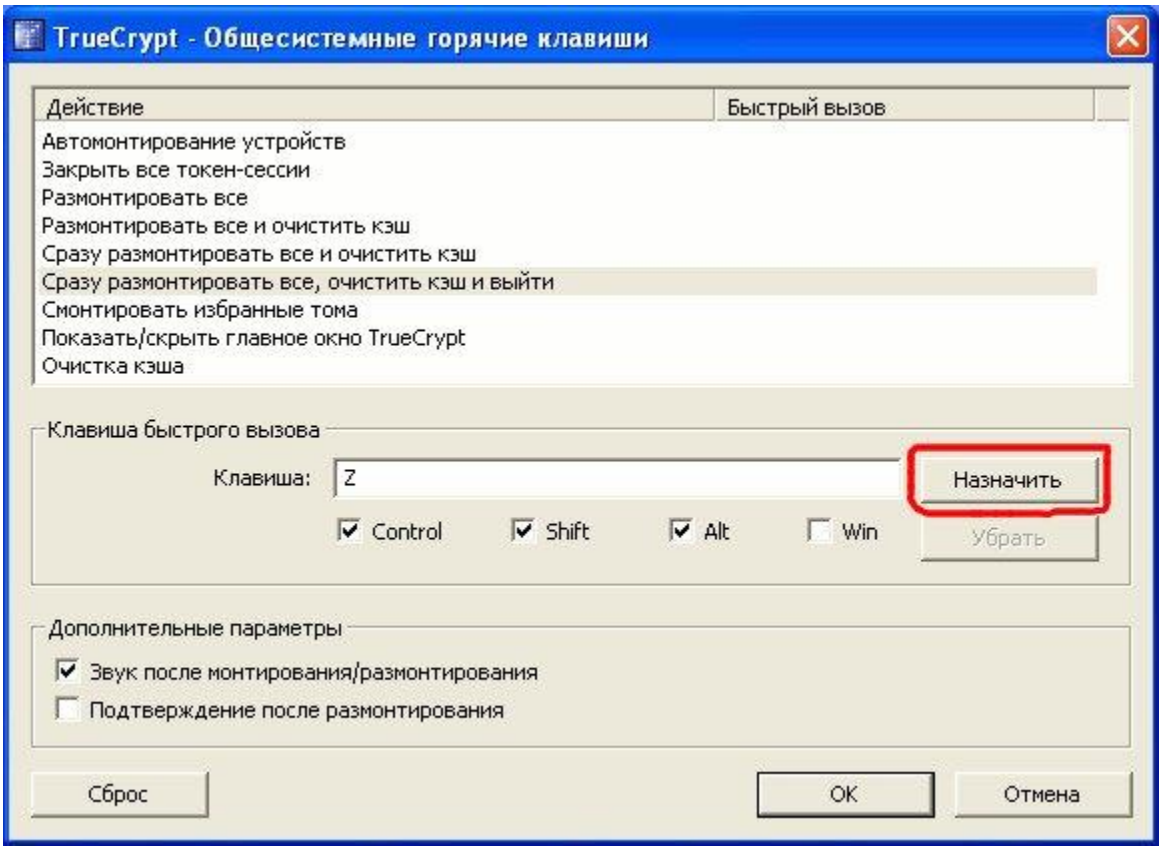


Рис. 3.4.

- Нажмите кнопку "ОК".

- Затем установите остальные настройки:
- Пройдите в меню программы Настройки - Настройки...

- Установите все настройки так, как это указано на скриншоте (рис 3.5). Нажмите кнопку "ОК".

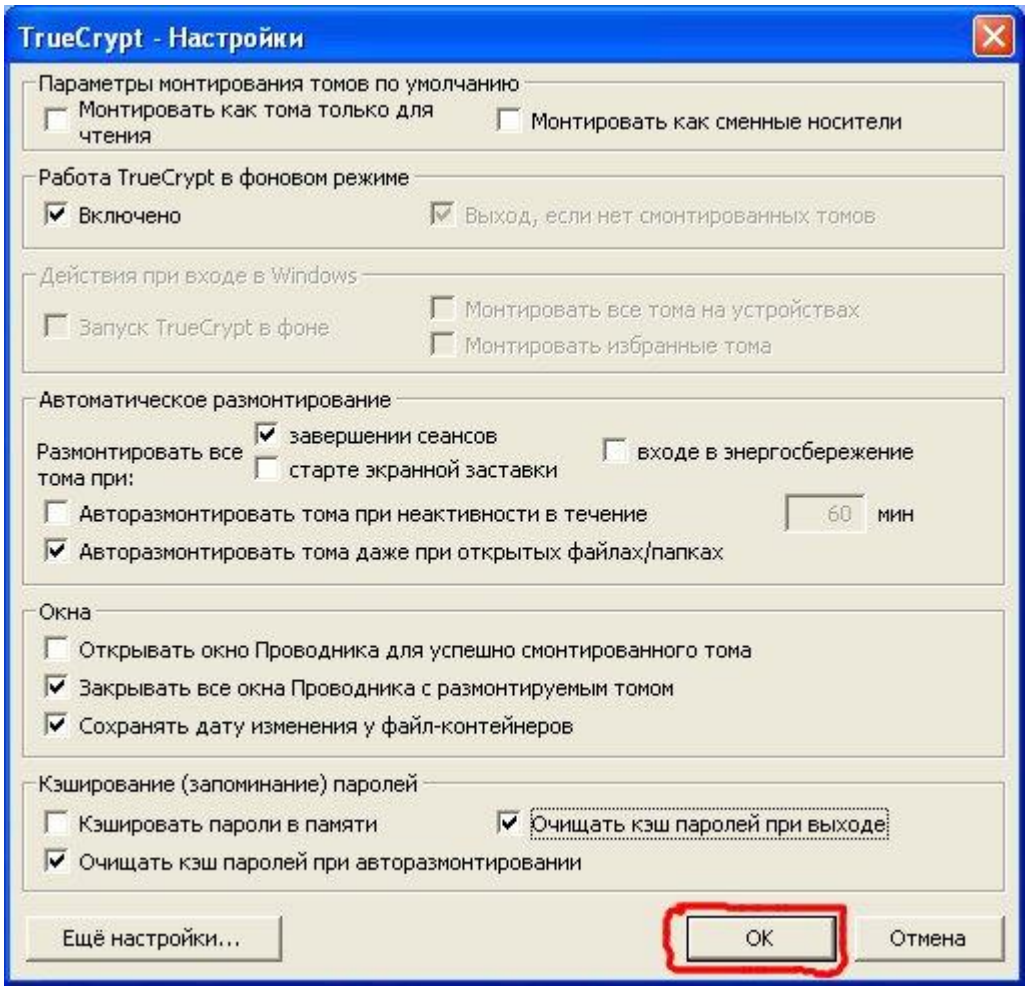


Рис. 3.5.

Создание внешнего тома TrueCrypt

- В основном окне программы нажмите кнопку "Создать том". Запустится мастер создания томов(рис 3.6.).

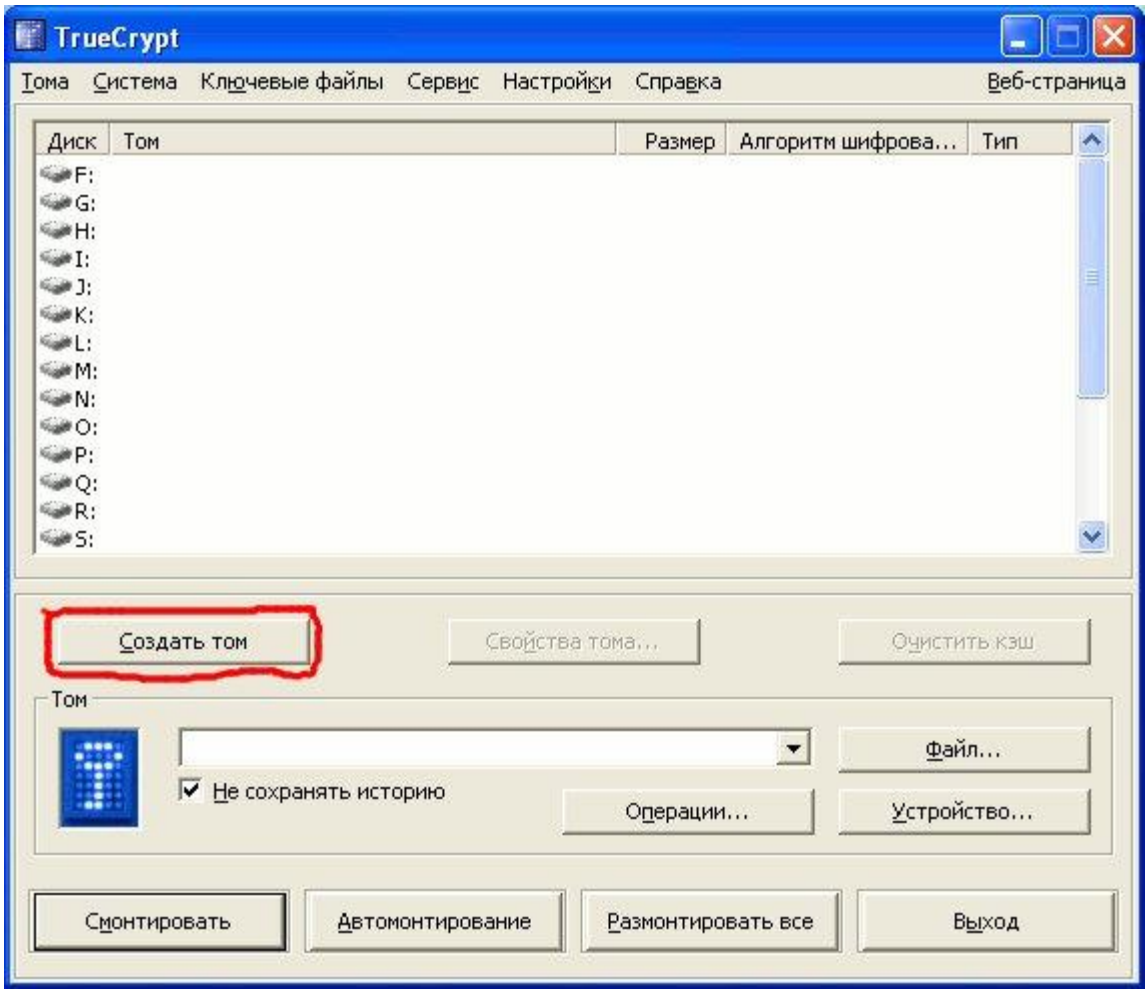


Рис. 3.6.

- В появившемся окне выберите пункт "Создать зашифрованный файловый контейнер" и нажмите кнопку "Далее"(рисунок 2.7.).

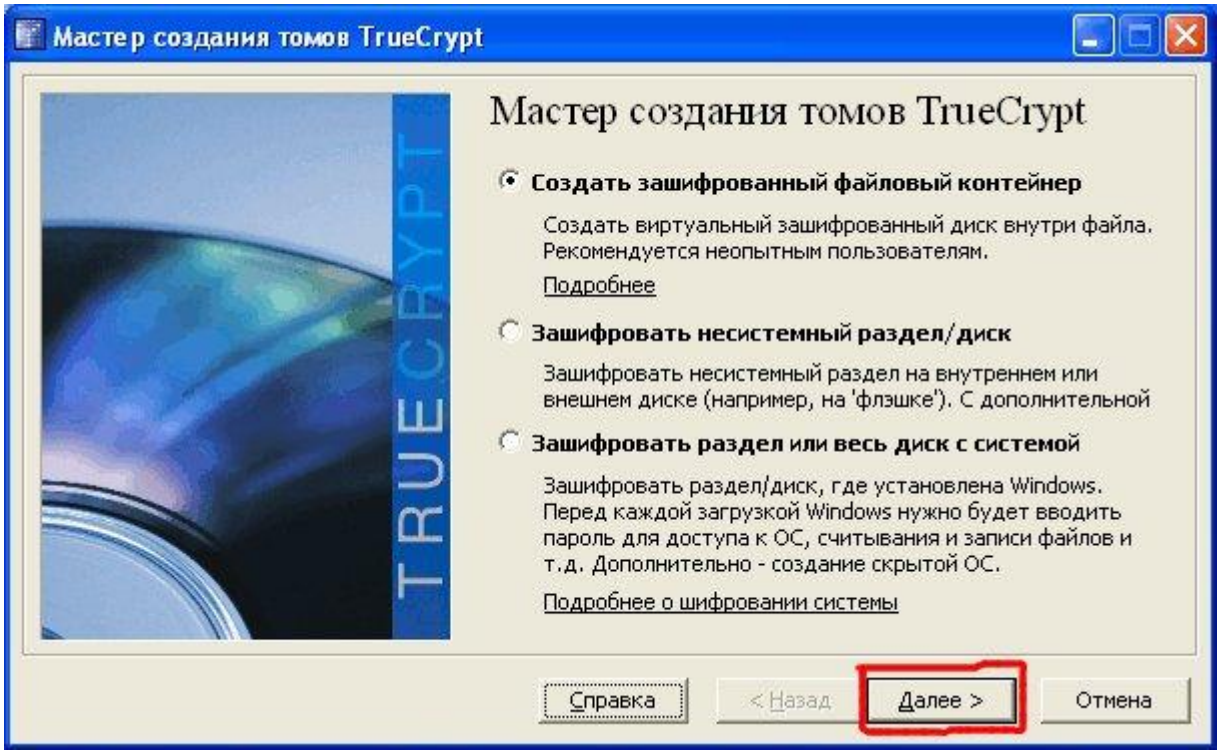


Рис. 3.7.

- Выберите пункт "Обычный том TrueCrypt" и нажмите кнопку "Далее"(рис. 3.8).

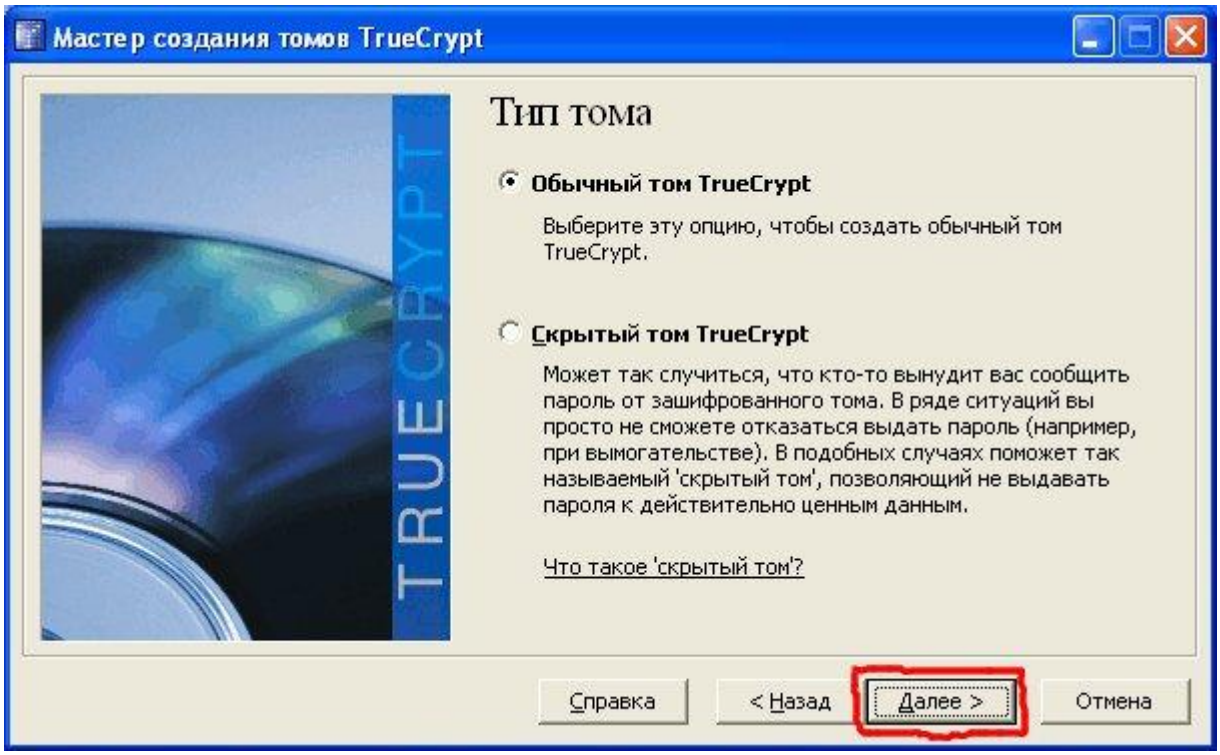


Рис. 3.8.

- Введите путь размещения файла, его имя и расширение C:TEMPFIRST.TXT. Этот файл будет создан и именно под него будет замаскирован том TrueCrypt. Убедитесь, что установлен флажок "Не сохранять историю". Нажмите кнопку "Далее". Если вы выберете уже имеющийся файл, то TrueCrypt не зашифрует его; этот файл будет удален и заменен созданным контейнером TrueCrypt. Впоследствии вы сможете зашифровать имеющиеся файлы, переместив их в контейнер TrueCrypt(рис. 3.9).

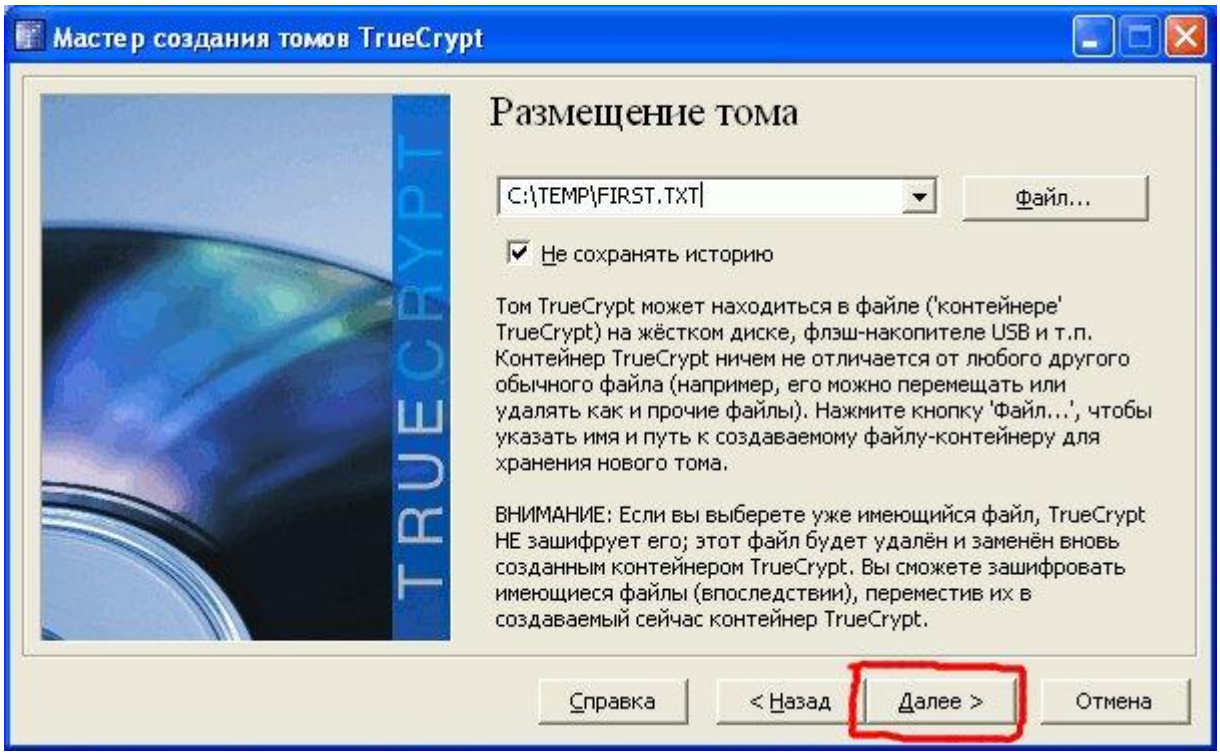


Рис. 3.9.

- В следующем окне вы можете выбрать алгоритм шифрования (по умолчанию - AES) и хэш-алгоритм (по умолчанию - RIPEMD-160). Установленные по умолчанию значения вполне надежны и обеспечивают приемлемую скорость шифрования/дешифрования информации. Поэтому рекомендуется оставить именно их. Нажмите кнопку "Далее"(рис. 3.10).

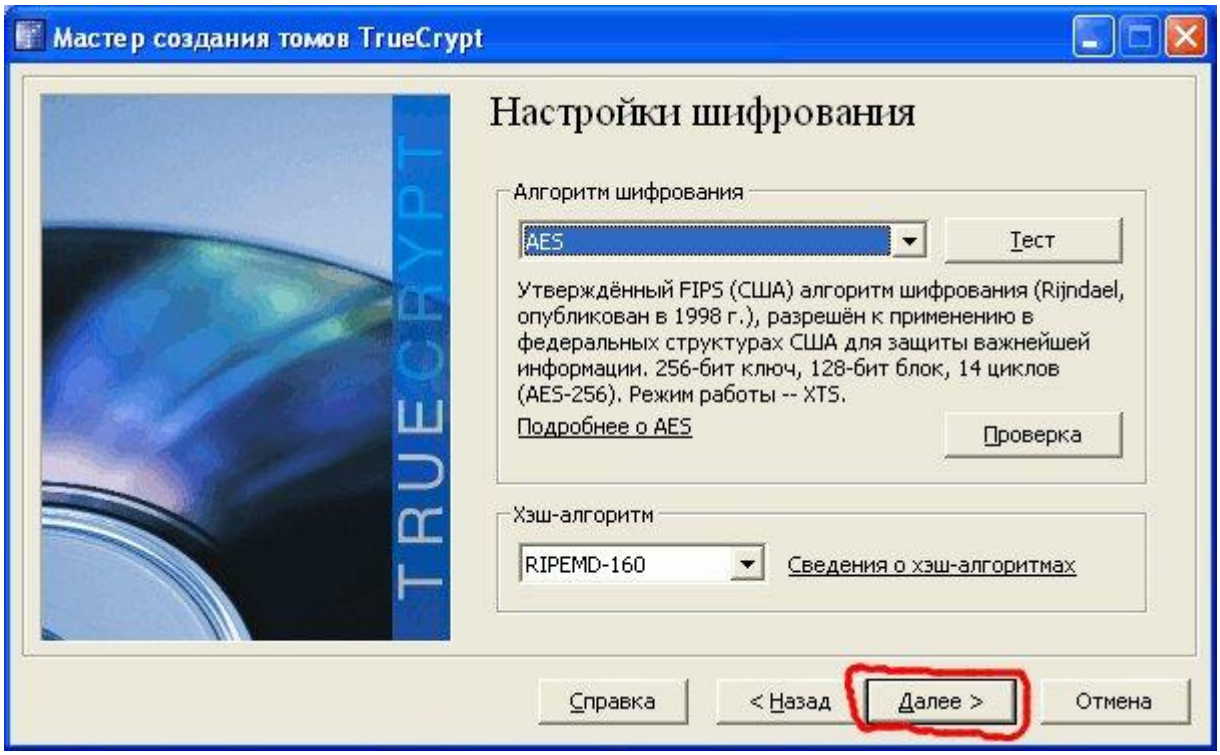


Рис. 3.10.

- Выберите размер тома и нажмите кнопку "Далее". При этом учтите, что скрытый том TrueCrypt, который мы будем создавать впоследствии, будет меньше, чем внешний том. Создадим внешний том размером 100 Мб(рис. 3.11).

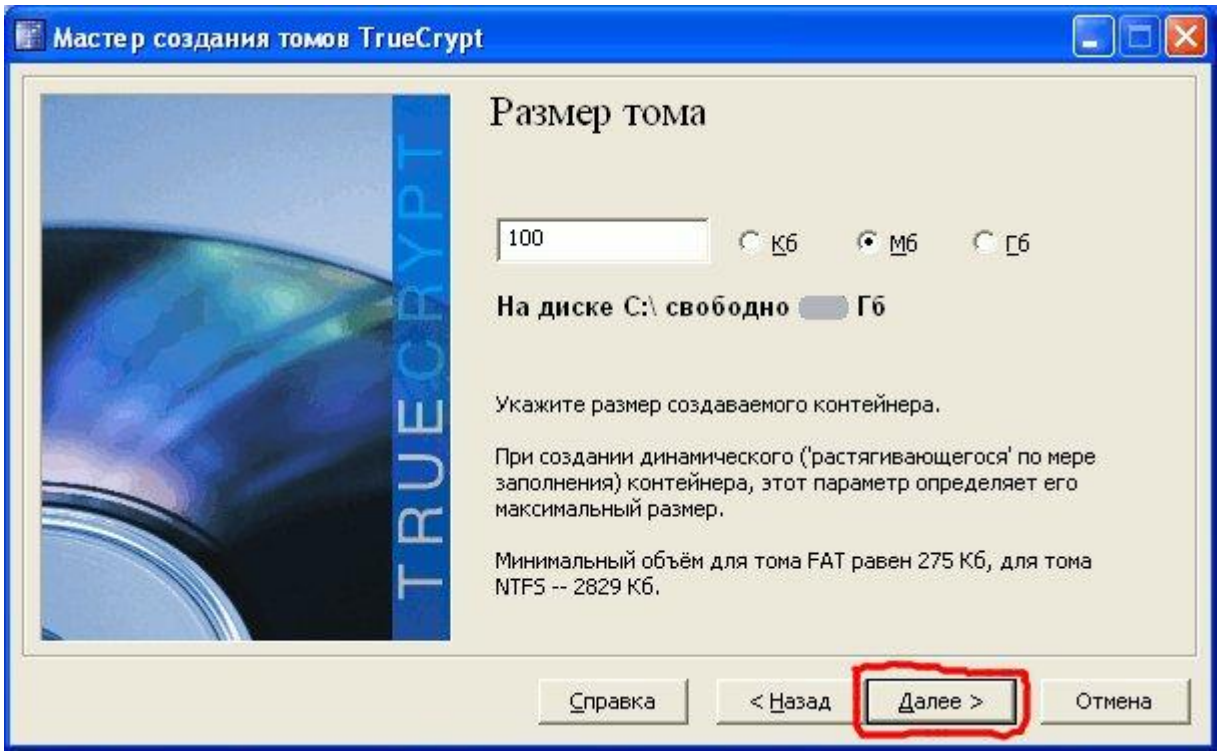


Рис. 3.11.

- Введите пароль и подтвердите его. Максимальная длина пароля - 64 символа. Постарайтесь выбрать пароль, состоящий не менее, чем из 20 знаков.

- Также рекомендуется выбрать ключевые файлы, т.е. файлы, которые вам нужно будет указывать каждый раз, когда вы будете монтировать том TrueCrypt. В качестве ключевых файлов подойдут любые, их расширение значения не имеет. Однако впоследствии вам нужно будет следить за тем, чтобы файлы, выбранные вами в качестве ключевых, не были повреждены. Иначе вы не сможете смонтировать том TrueCrypt и получить доступ к данным.

- Для выбора ключевых файлов установите флажок "Ключ. файлы" и нажмите кнопку "Ключ. файлы...". В открывшемся окне нажмите кнопку "Файлы..." и выберите ключевые файлы. При этом учтите, что запоминается путь, а не только имена файлов. Для того, чтобы том мог быть смонтирован, файлы должны располагаться в той же директории, что и при их выборе в качестве ключевых во время создания тома. Если добавить папку, нажав на кнопку "Папка", то все файлы внутри нее будут использоваться как ключевые.

- Выберите любой файл в качестве ключевого. Полный путь к файлу, который будет указан - C:song.wav. Вы можете выбрать любое количество файлов. Для большей секретности файлы могут располагаться на USB флеш-накопителе или диске. Нажмите кнопку "OK"(рис. 3.12).

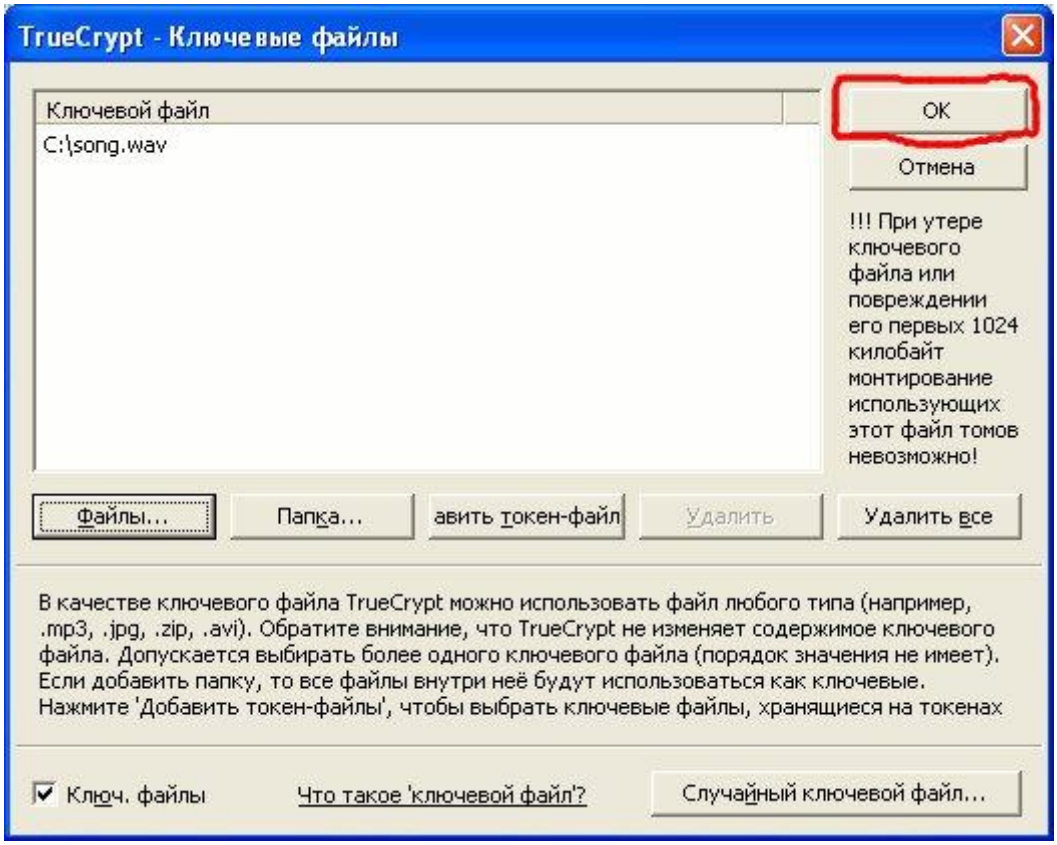


Рис. 3.12.

В окне "Мастер создания томов TrueCrypt" нажмите кнопку "Далее"(рис. 3.13).

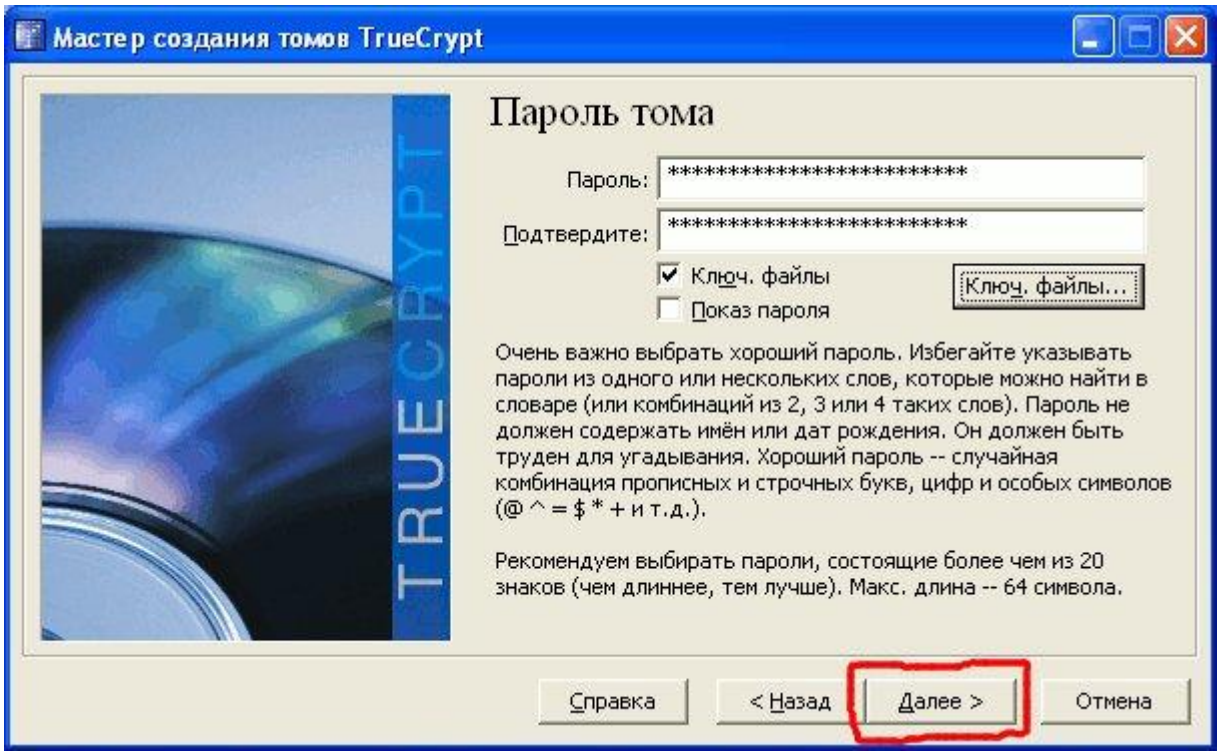


Рис. 3.13.

- В следующем окне вы можете выбрать тип файловой системы создаваемого тома. Для внешнего тома рекомендуется использовать файловую систему FAT, чтобы скрытый том можно было создать максимально возможного размера. Тип кластера установите "По умолчанию". Не устанавливайте флажок "Динамический".

- После этого поводите курсором мыши по этому окну. Вы заметите, что при движении курсора случайный код меняется быстрее. Такое броуновское движение курсором улучшает случайность шифровки этого тома. Нажмите кнопку "Разметить"(рис. 3.14).

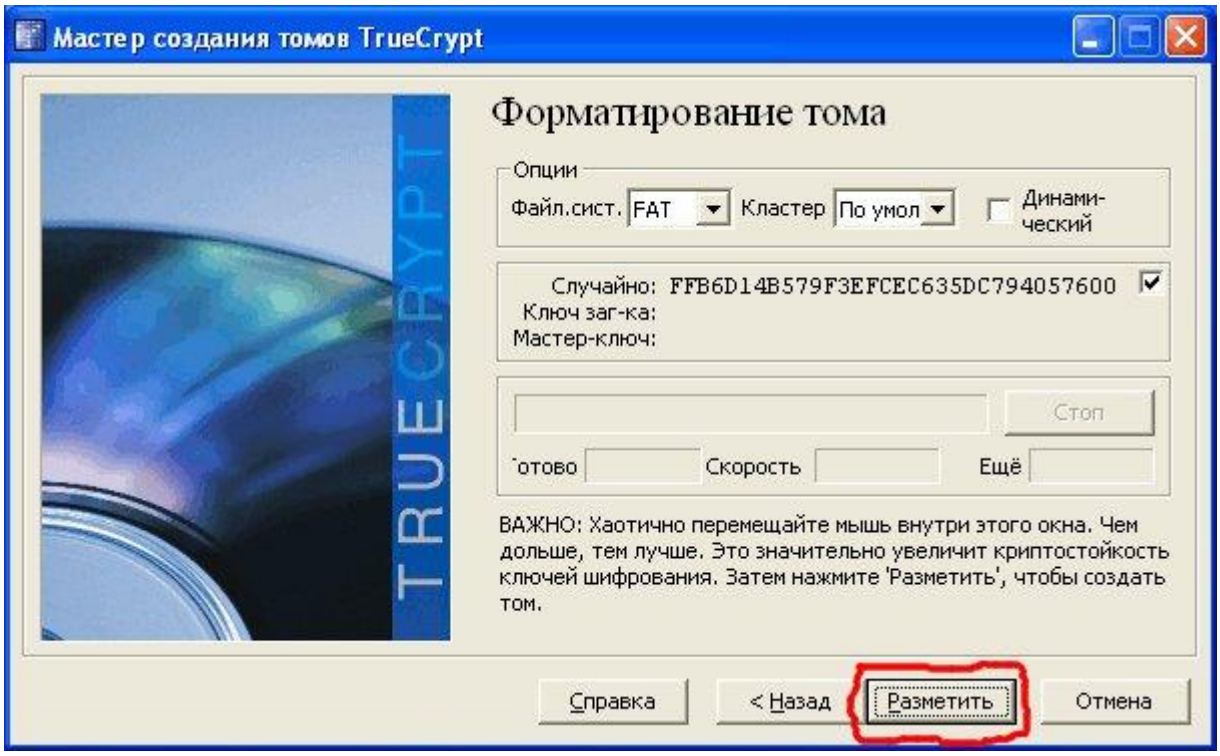


Рис. 3.14.

- После этого будет запущен процесс создания внешнего тома TrueCrypt. Дождитесь, пока он будет завершен(рис. 3.15.).

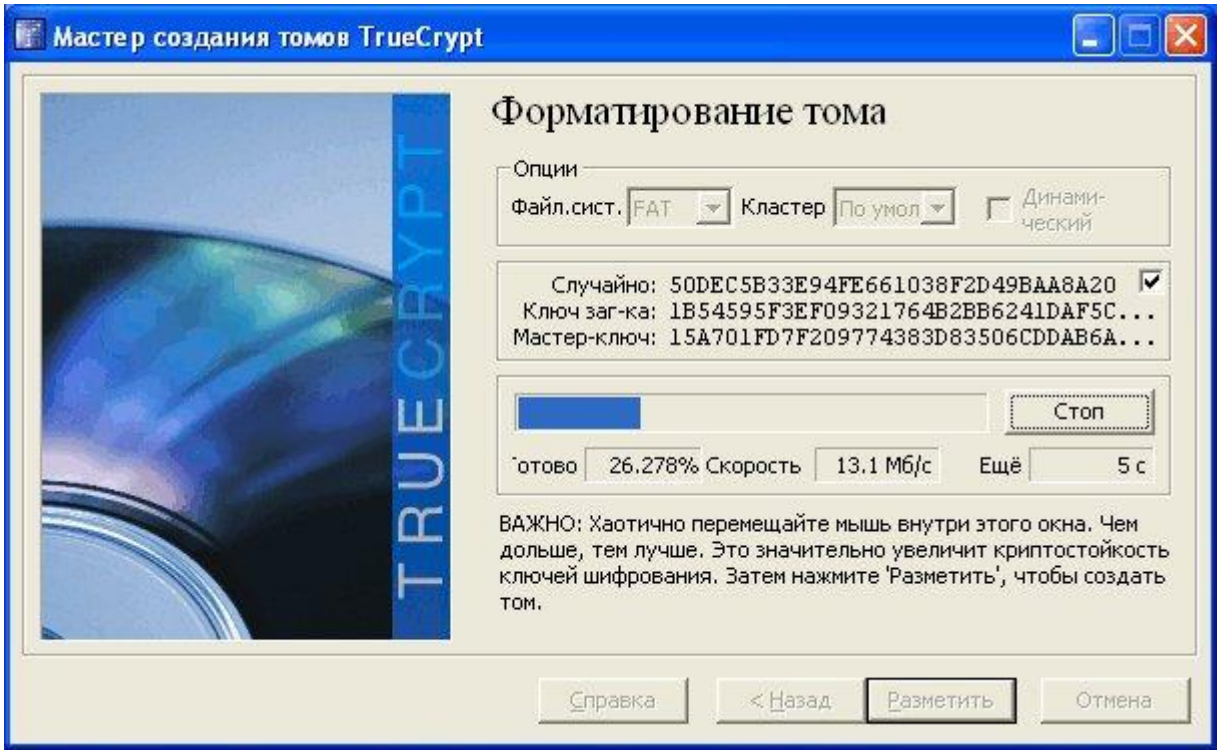


Рис. 3.15.

- После создания тома появится сообщение о его успешном создании. Нажмите кнопку "ОК".

- В следующем окне нажмите кнопку "Далее", если хотите продолжить процесс создания томов, или кнопку "Выход".

4.2. Создание скрытого тома TrueCrypt

После того, как создан внешний том TrueCrypt, можно перейти к созданию скрытого тома.

Обратите внимание, что скрытый том будет спрятан внутрь только что созданного внешнего тома. Отсюда следует, что скрытый том не может быть больше внешнего тома. В примере это 100 Мб для внешнего тома и 70 Мб для скрытого. Такая разница нужна для того, чтобы мы могли впоследствии разместить во внешнем томе несколько файлов с посторонней информацией для маскировки. Можно создавать несколько скрытых томов внутри одного внешнего тома. Главное, чтобы суммарный объем всех скрытых томов не превышал размера внешнего тома. Также обратите внимание, что для создания скрытого тома нужно ввести пароль, который вы ранее использовали для внешнего тома, а в окне "Пароль скрытого тома" - новый пароль для скрытого тома (дважды - в полях "Пароль" и "Подтвердите").

- В основном окне программы нажмите кнопку "Создать том". Запустится мастер создания томов(рис. 3.16).

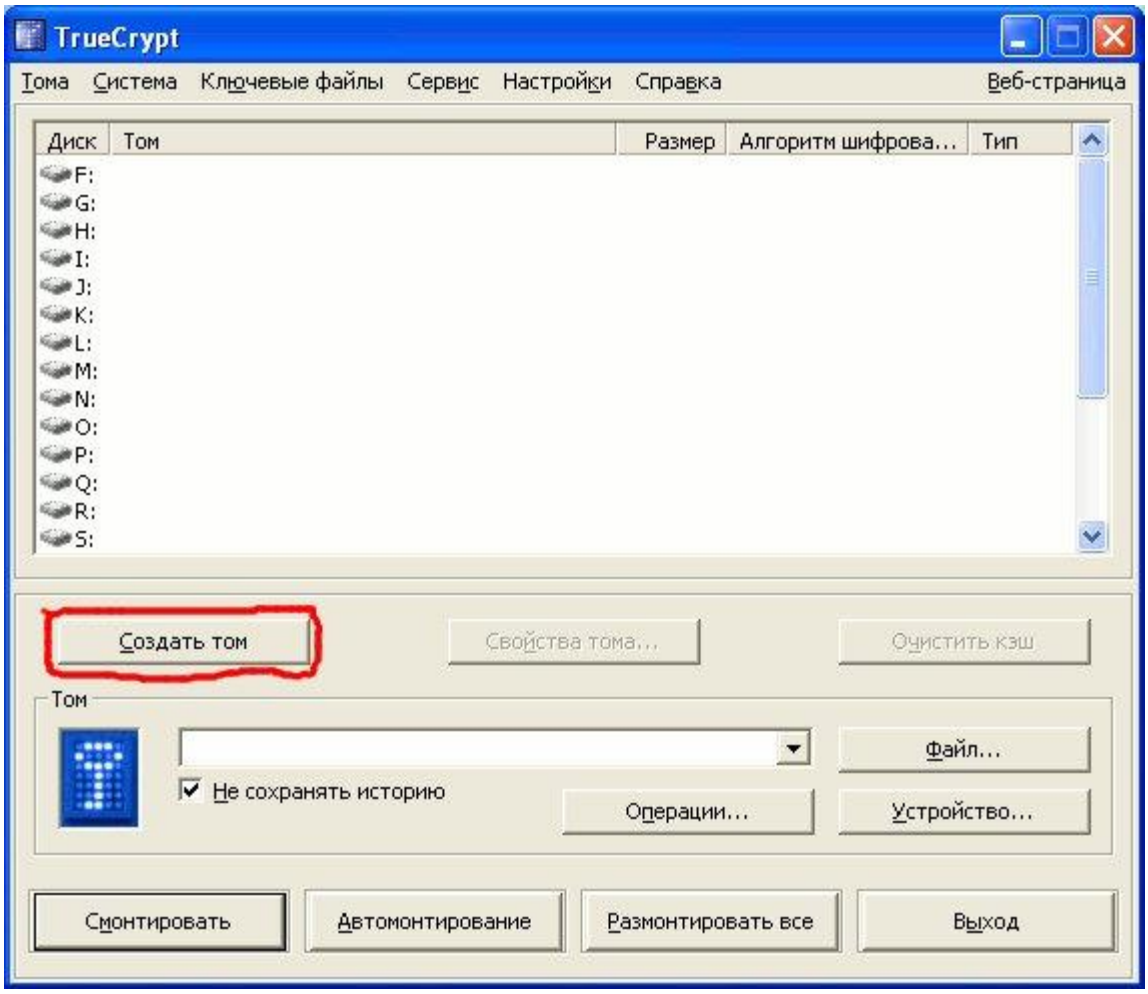


Рис. 3.16.

- В появившемся окне выберите пункт "Создать зашифрованный файловый контейнер" и нажмите кнопку "Далее"(рис. 3.17.).

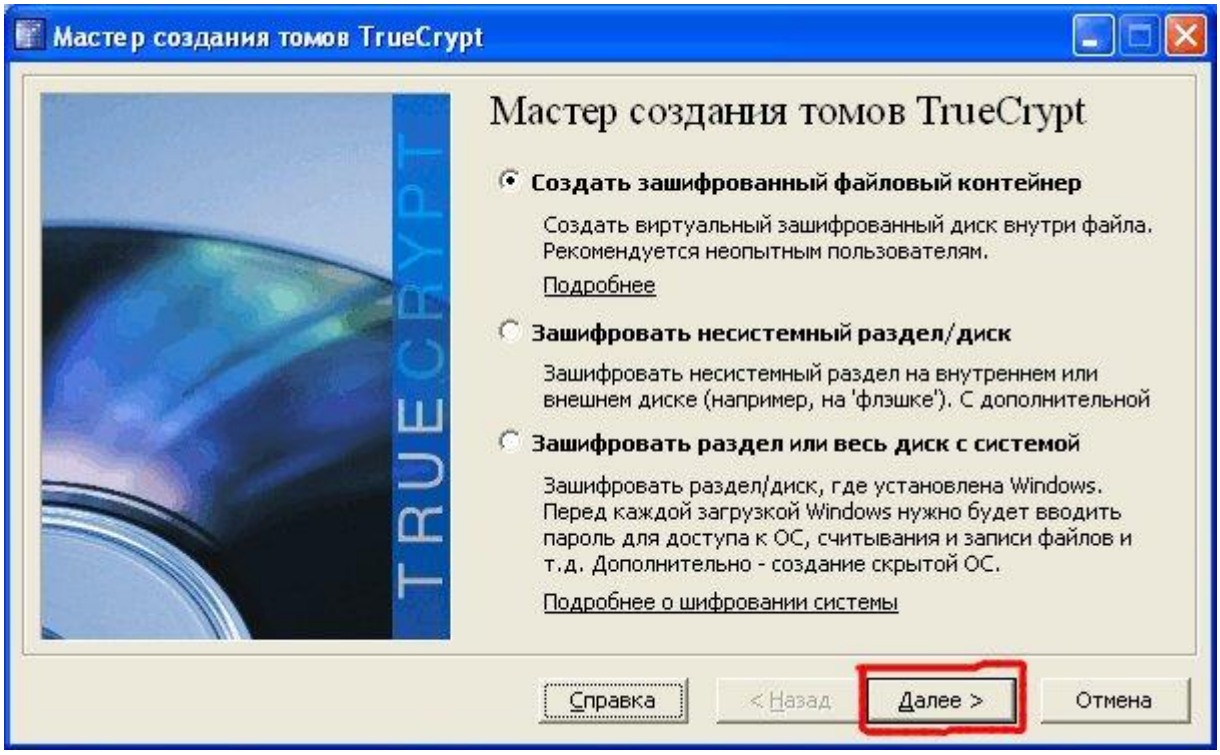


Рис. 3.17.

- Выберите пункт "Скрытый том TrueCrypt" и нажмите кнопку "Далее"(рис. 3.18).

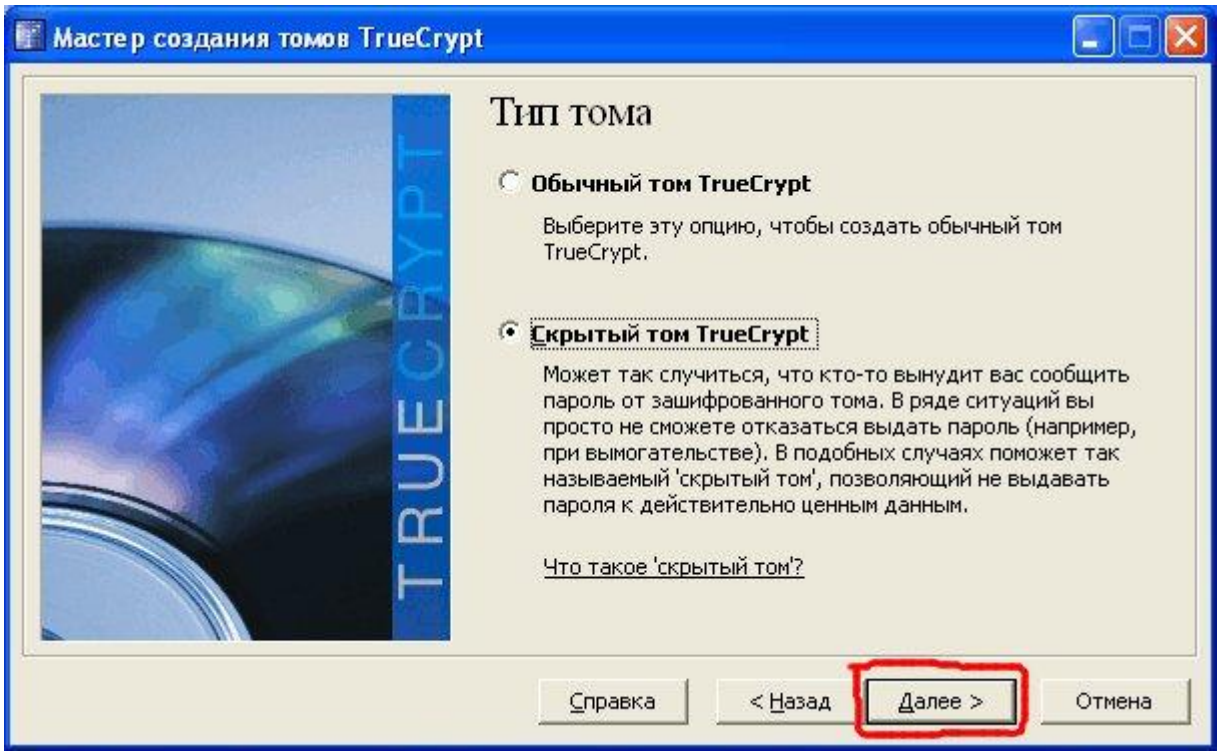


Рис. 3.18.

- Выберите пункт "Прямой режим" и нажмите кнопку "Далее"(рис. 3.19).

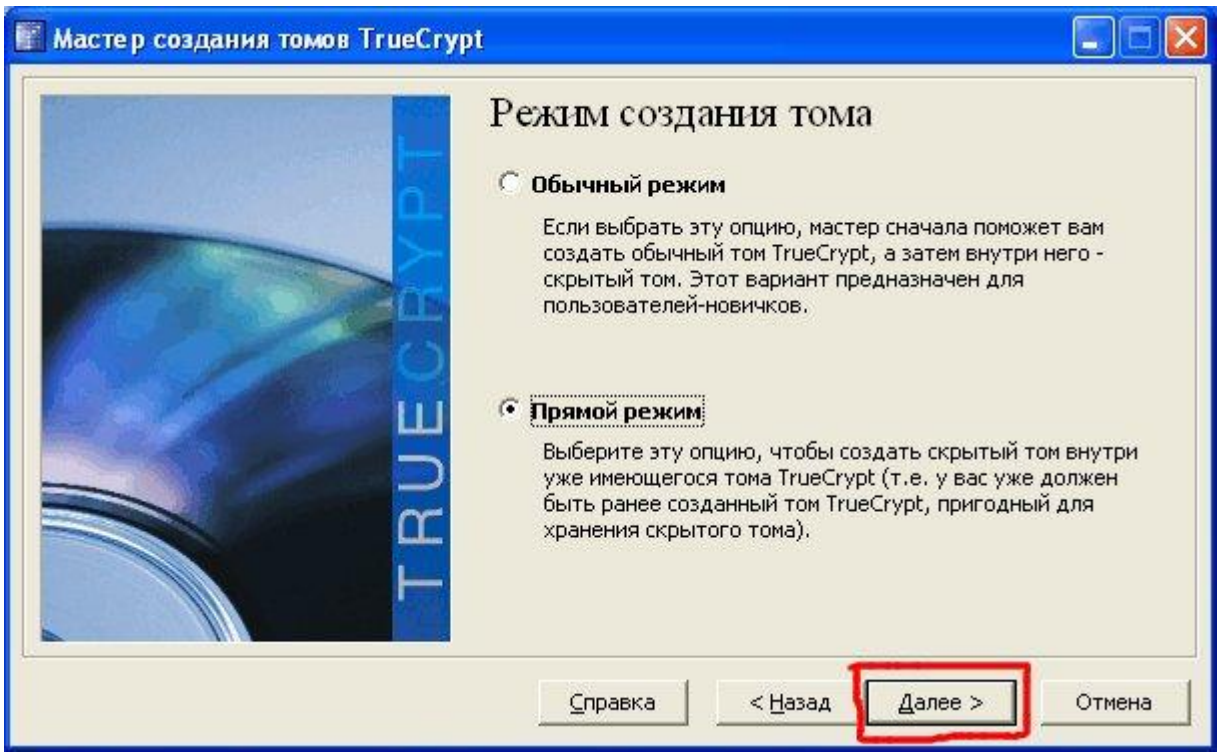


Рис. 3.19.

- Укажите путь к файлу, который маскирует внешний том TrueCrypt. Внутри этого внешнего тома будет создан скрытый том TrueCrypt. В нашем примере этот путь - C:TEMPFIRST.TXT. Убедитесь, что установлен флажок "Не сохранять историю". Нажмите кнопку "Далее"(рис. 3.20).

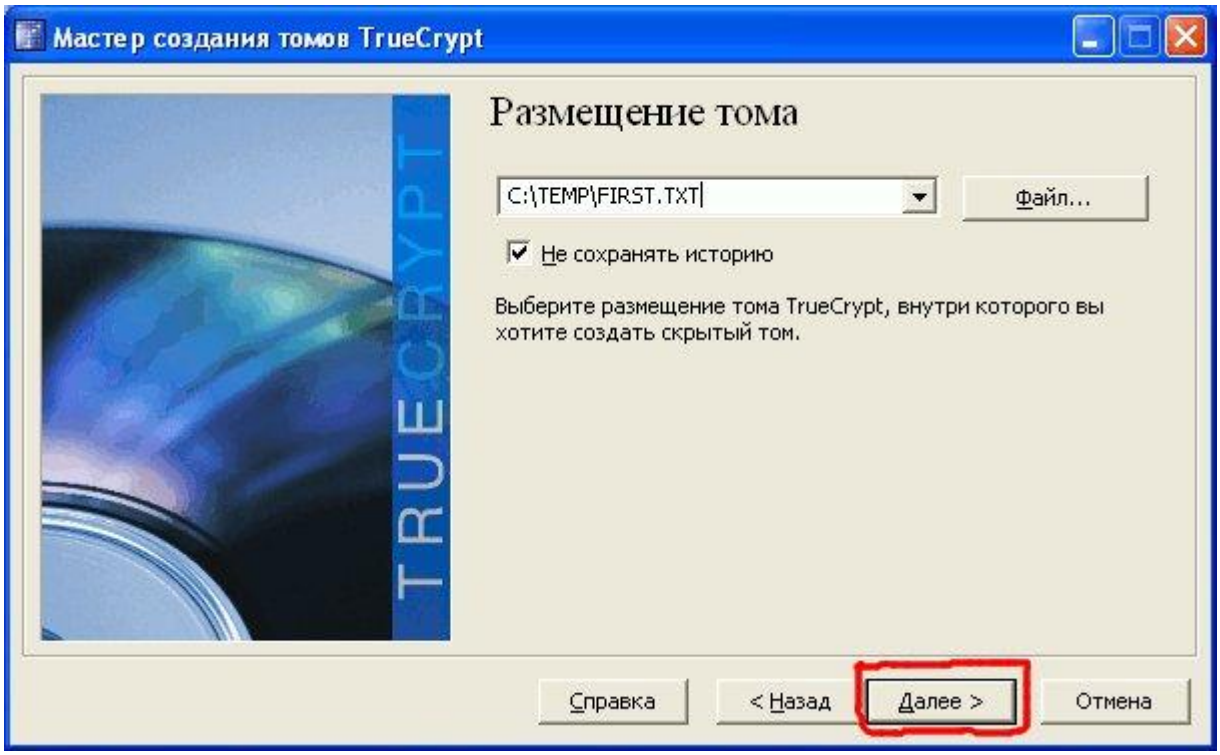


Рис. 3.20.

- В следующем окне введите пароль внешнего тома. Если вы при создании внешнего тома указывали ключевые файлы, необходимые для монтирования тома, то установите флажок "Ключ. файлы" и нажмите на кнопку "Ключ. файлы...".
- В открывшемся окне нажмите кнопку "Файлы..." и выберите файлы, которые вы указывали в качестве ключевых при создании внешнего тома. В нашем примере это C:song.wav. После того, как будут указаны все ключевые файлы, нажмите кнопку "ОК"(рис. 3.21).

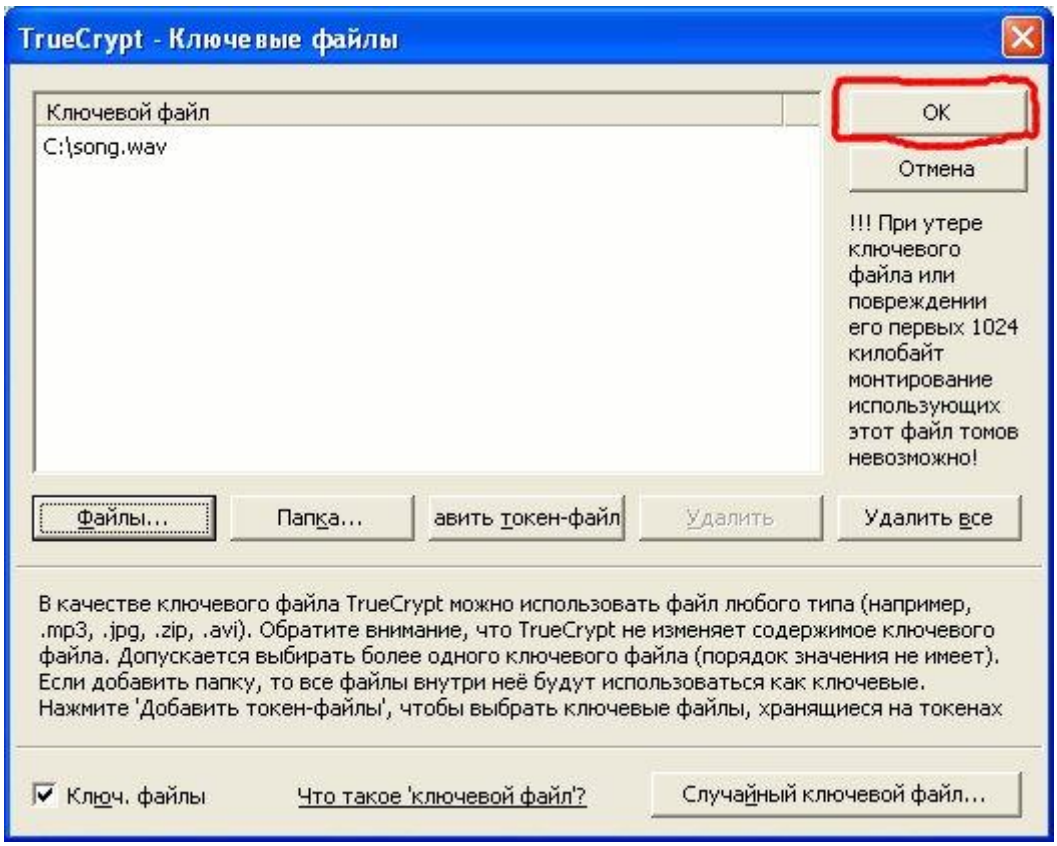


Рис. 3.21.

- В окне "Мастер создания томов TrueCrypt" нажмите кнопку "Далее"(рис. 3.22).

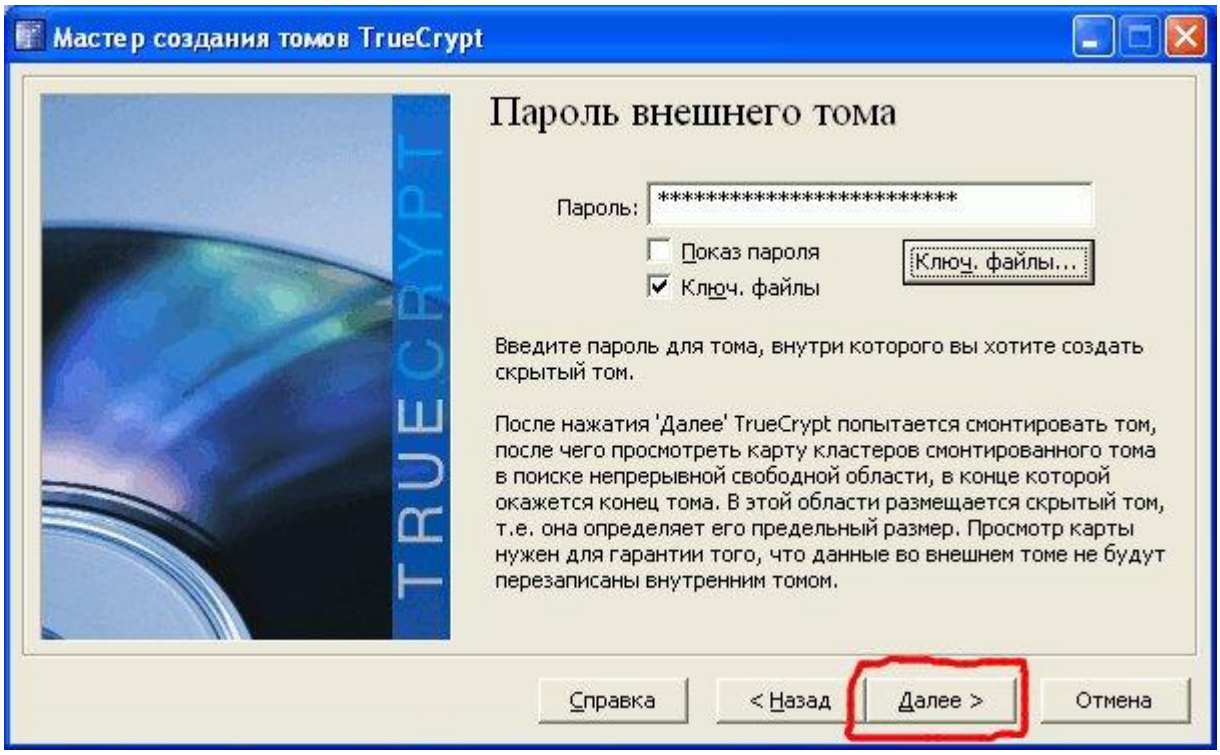


Рис. 3.22.

- В следующем окне нажмите кнопку "Далее"(рис. 3.23).

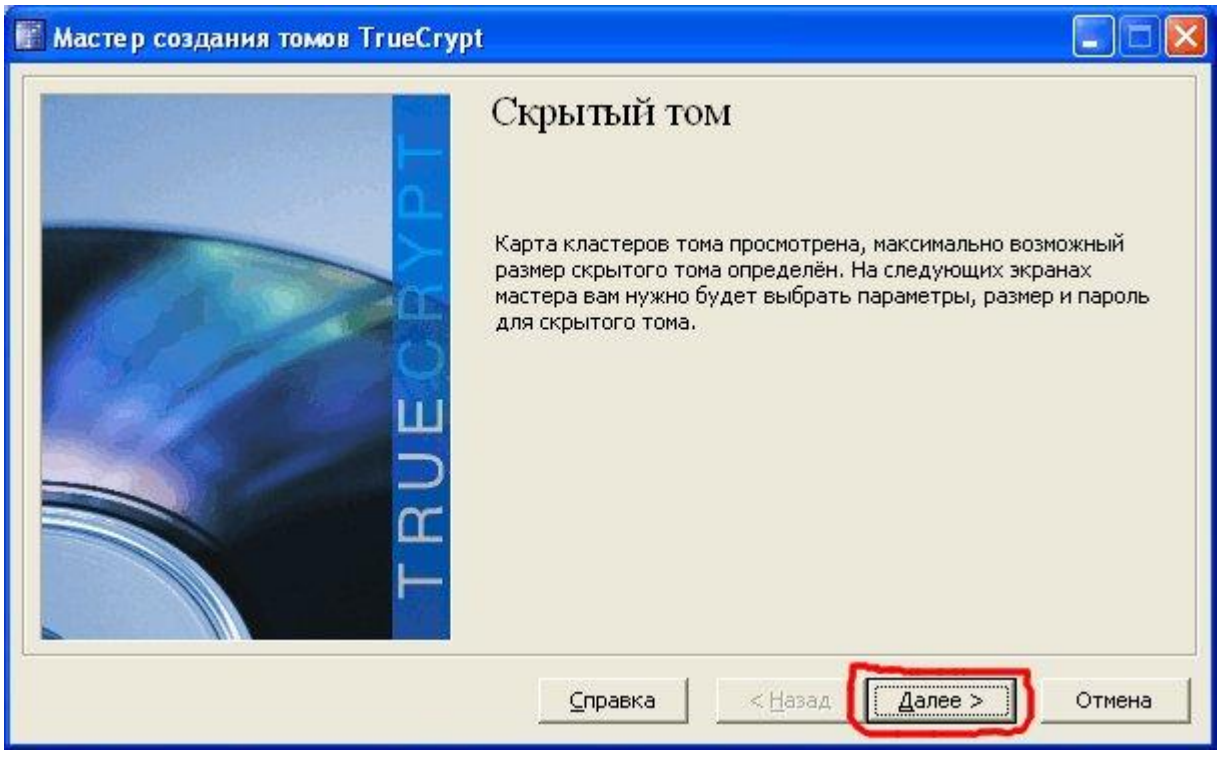


Рис. 3.23.

- В следующем окне вы можете выбрать алгоритм шифрования и хэш-алгоритм скрытого тома. Рекомендуется оставить те значения, которые установлены по умолчанию. Нажмите кнопку "Далее"(рис 3.24).

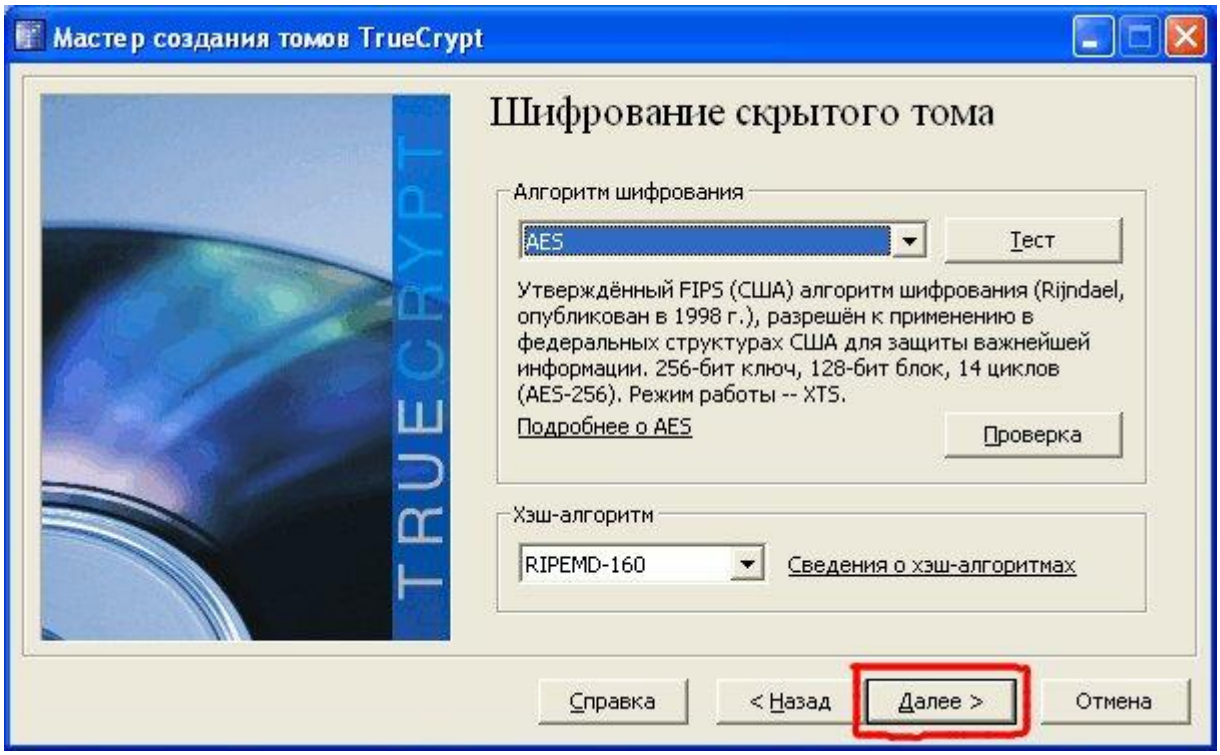


Рис. 3.24.

- В следующем окне будет указан максимально возможный размер скрытого тома. Укажите размер скрытого тома. В нашем примере мы выбрали размер 70 Мб. Нажмите кнопку "Далее"(рис. 3.25).

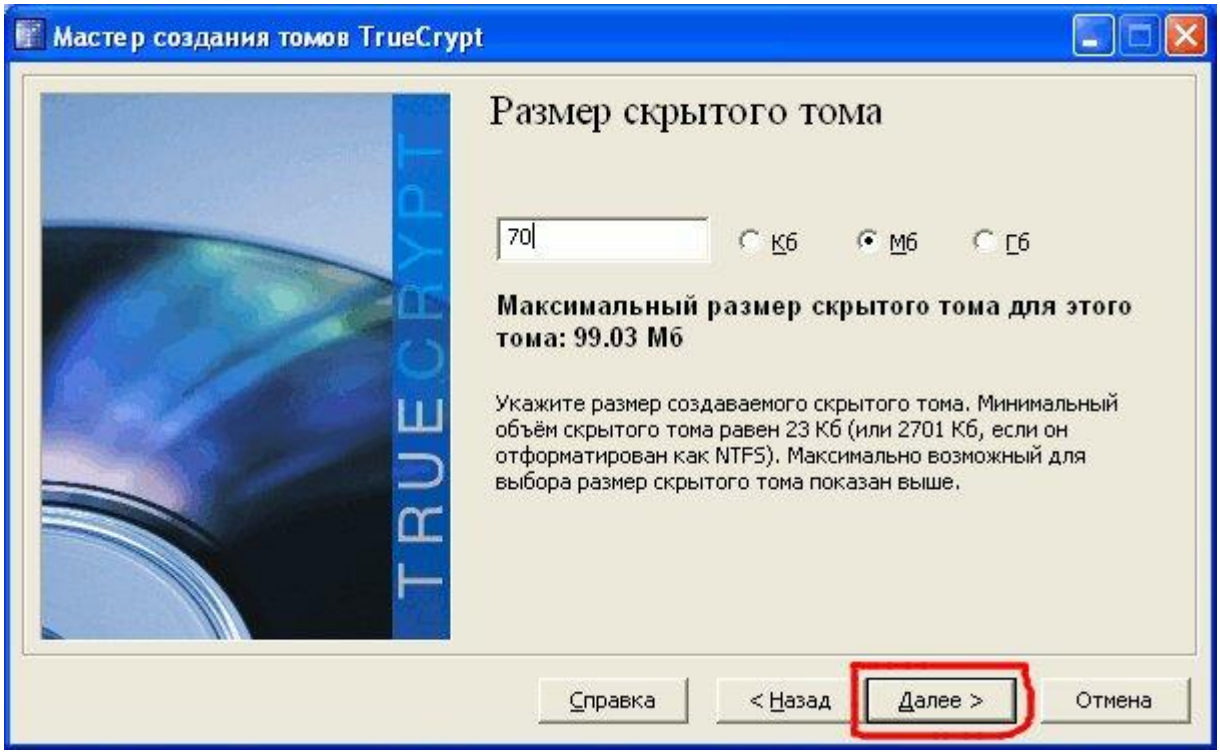


Рис. 3.25.

- Введите пароль скрытого тома и подтвердите его. Максимальная длина пароля - 64 символа. Постарайтесь выбрать пароль, состоящий не менее, чем из 20 знаков.

- Также выберите и укажите ключевые файлы так, как вы это уже умеете. Нажмите кнопку "Далее"(рис. 3.26).

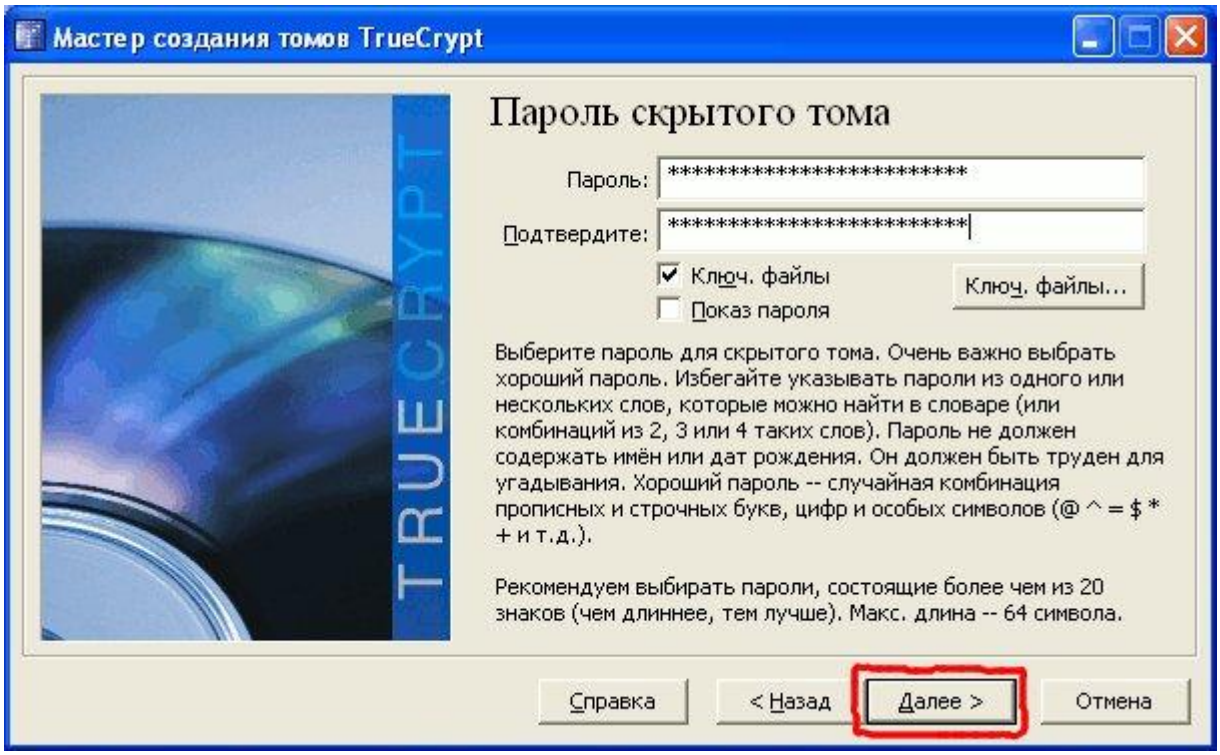


Рис. 3.26.

- В следующем окне вы можете выбрать тип файловой системы создаваемого тома. Для скрытого тома рекомендуется использовать файловую систему NTFS. Тип кластера установите "По умолчанию".

- После этого поводите курсором мыши по этому окну, как и при создании внешнего тома. Потом нажмите кнопку "Разметить"(рис.3.27).

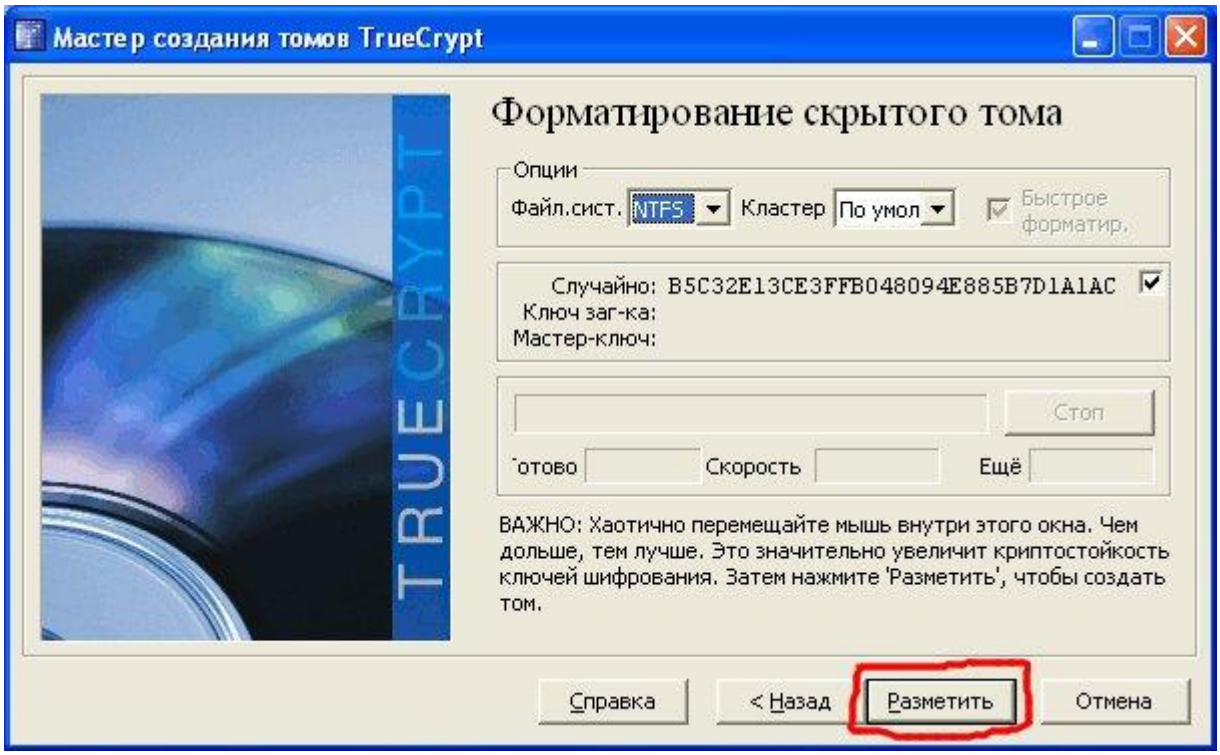


Рис. 3.27.

Будет запущен процесс создания скрытого тома TrueCrypt. Дождитесь, пока он будет завершен(рис 3.28).

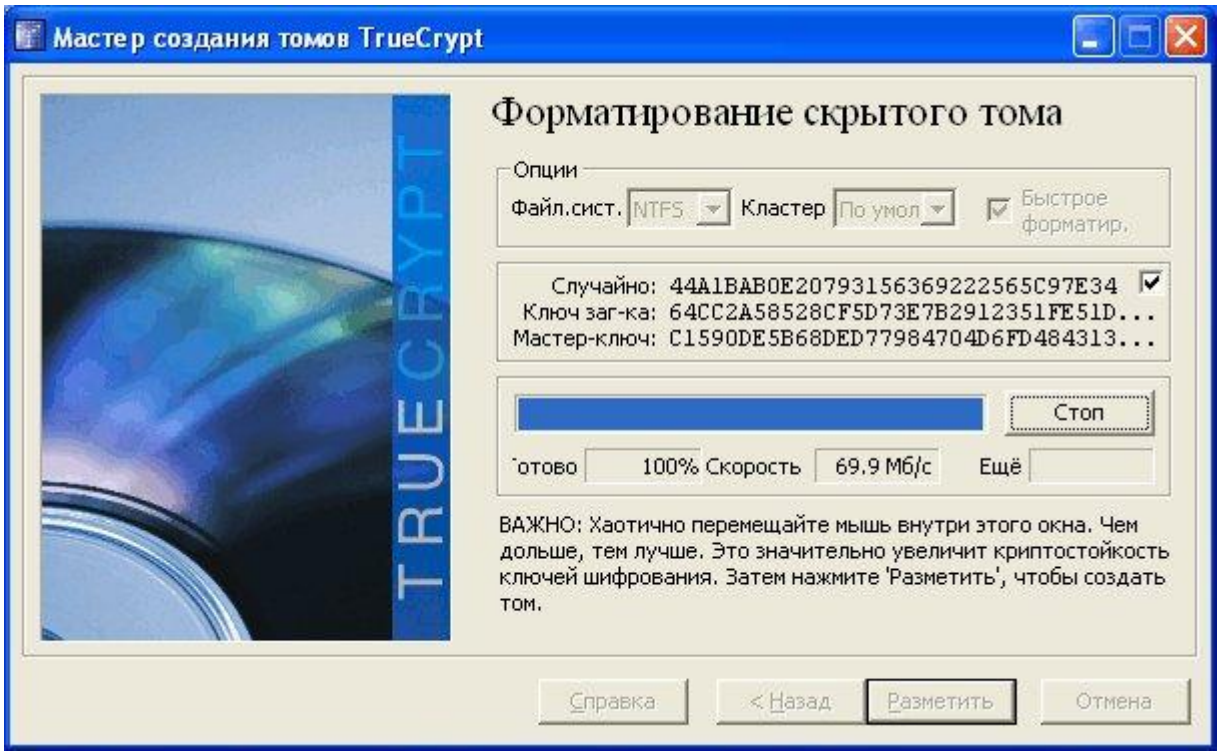


Рис. 3.28.

- После окончания процесса создания скрытого тома появится окно, предупреждающее о разумности защиты скрытых томов при монтировании внешнего тома. Если вы об этом забудете, вы можете случайно затереть информацию на скрытом томе, когда размещаете файлы на внешнем томе. В этом окне нажмите кнопку "ОК"(рис 3.29).

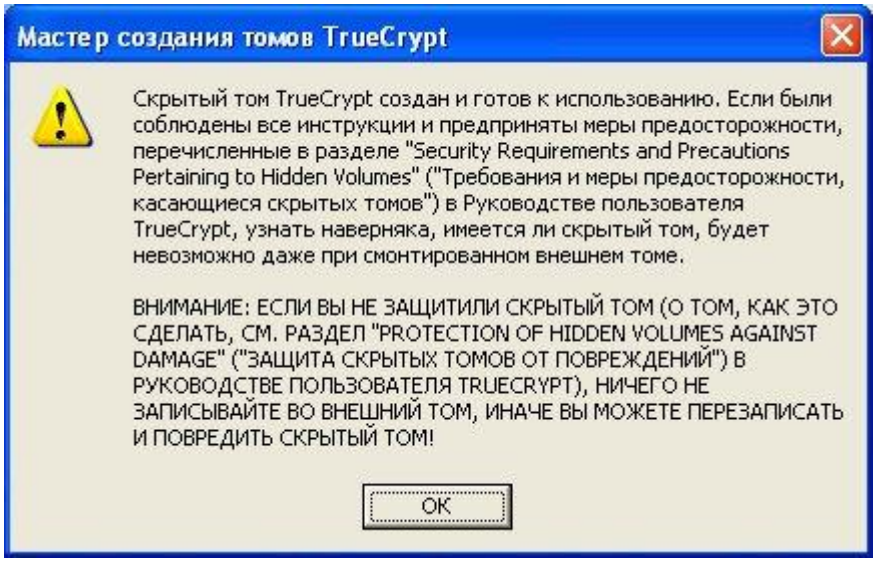


Рис. 3.29.

- В следующем окне нажмите кнопку "Выход"(рис. 3.30).

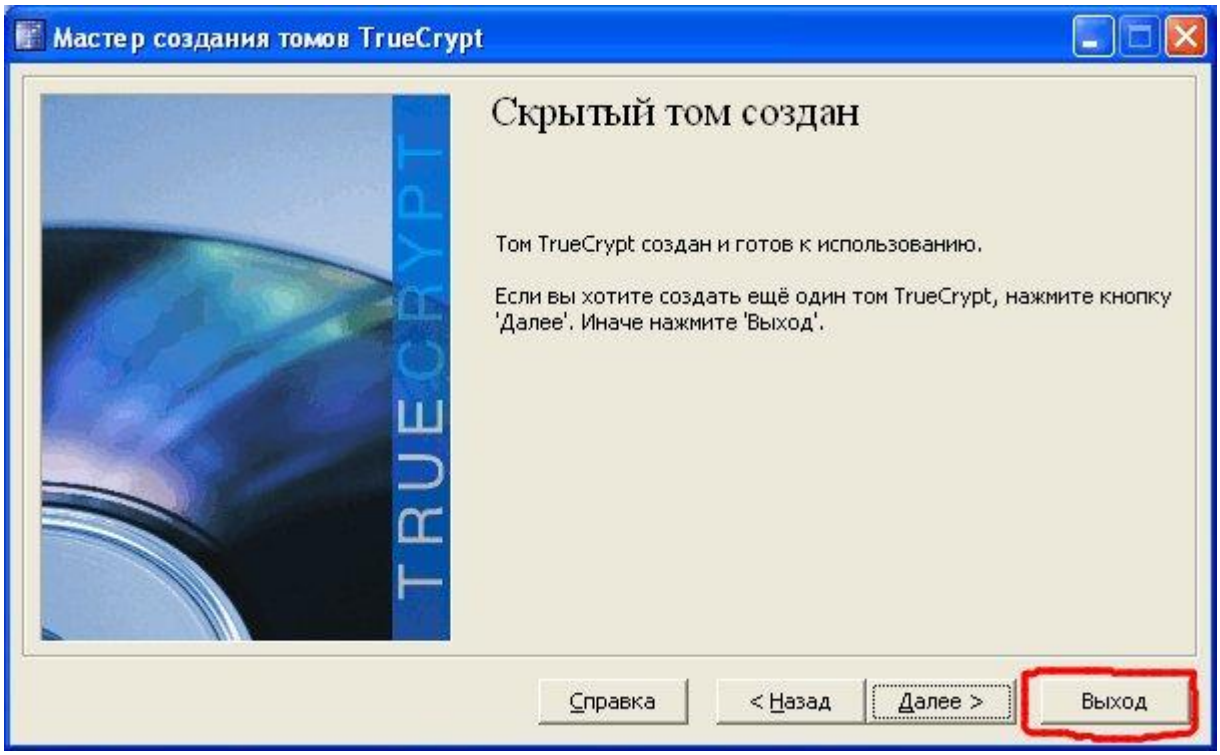


Рис. 3.30.

4.3. Работа с томами TrueCrypt

Итак, вот, что у нас получилось в результате создания внешнего и скрытого томов(рис. 3.31):

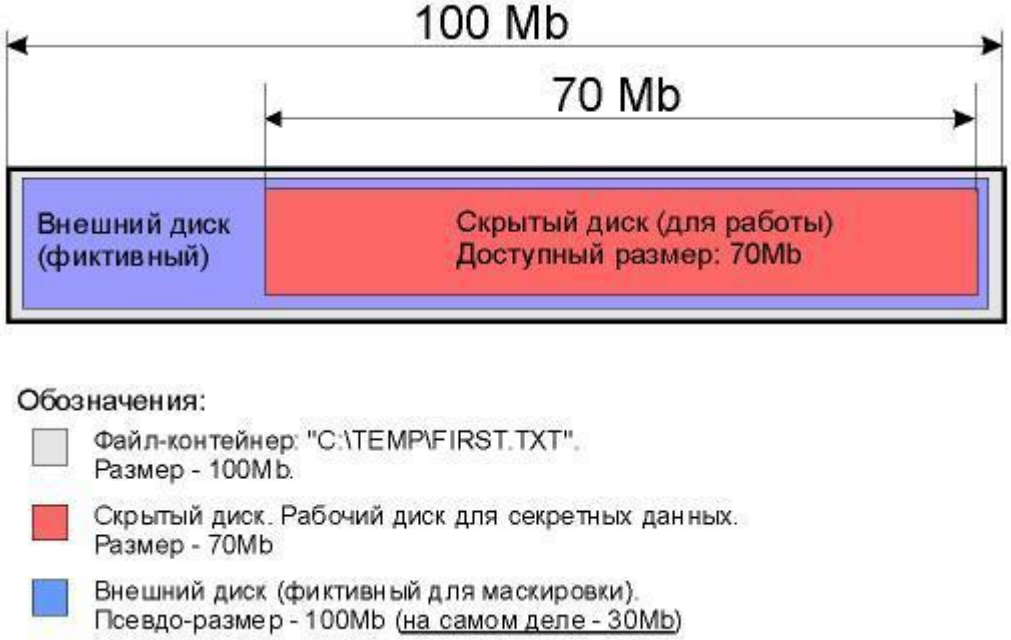


Рис. 3.31.

Теперь нужно смонтировать внешний том и заполнить его информацией для маскировки.

- В основном окне программы выберите любой диск, под видом которого будет смонтирован внешний том, например F.

- После этого в основном окне программы нажмите кнопку "Файл..."(рис. 3.32).

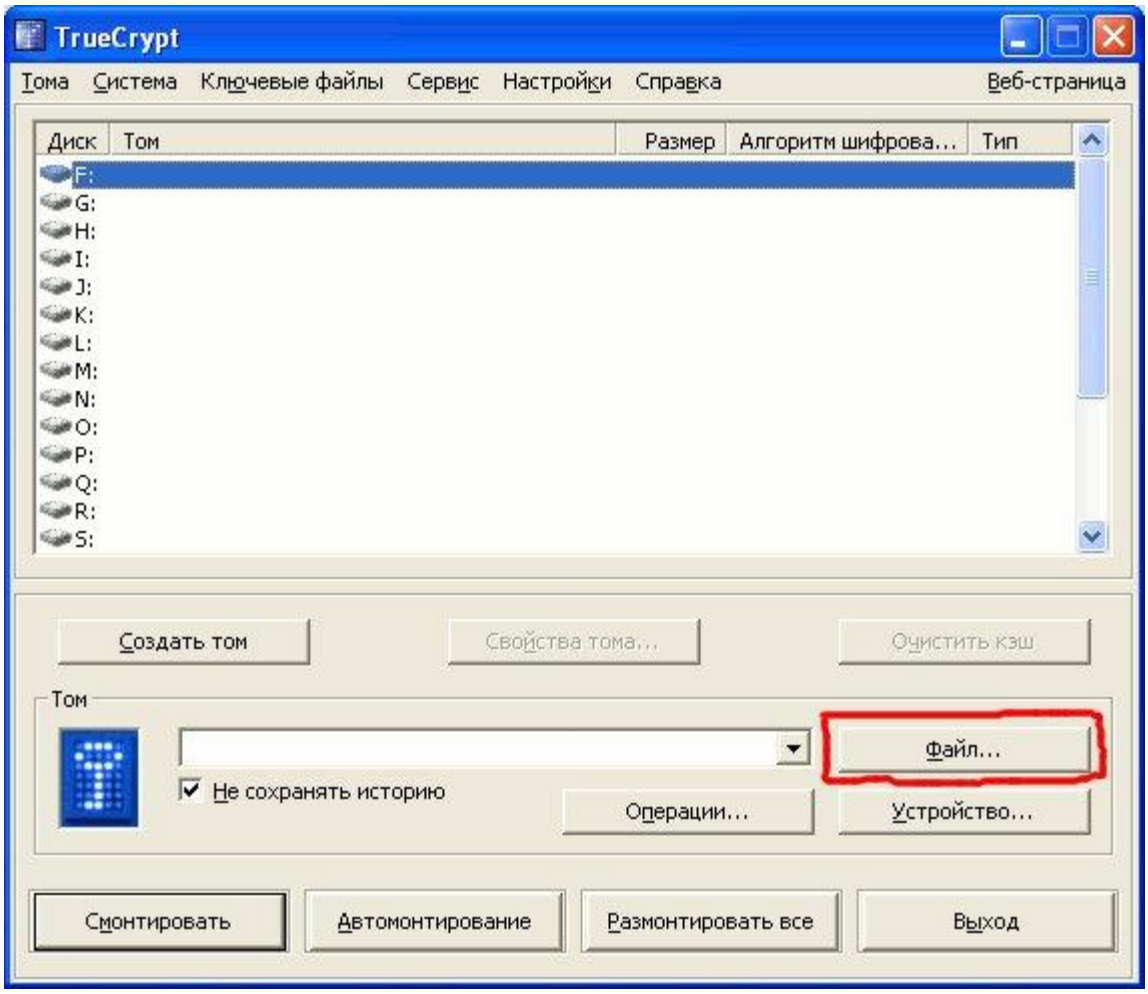


Рис. 3.32.

- Выберите файл, под который замаскирован том TrueCrypt. В нашем примере это C:TEMPFIRST.TXT. Нажмите кнопку "Смонтировать" в основном окне программы(рис. 3.33).

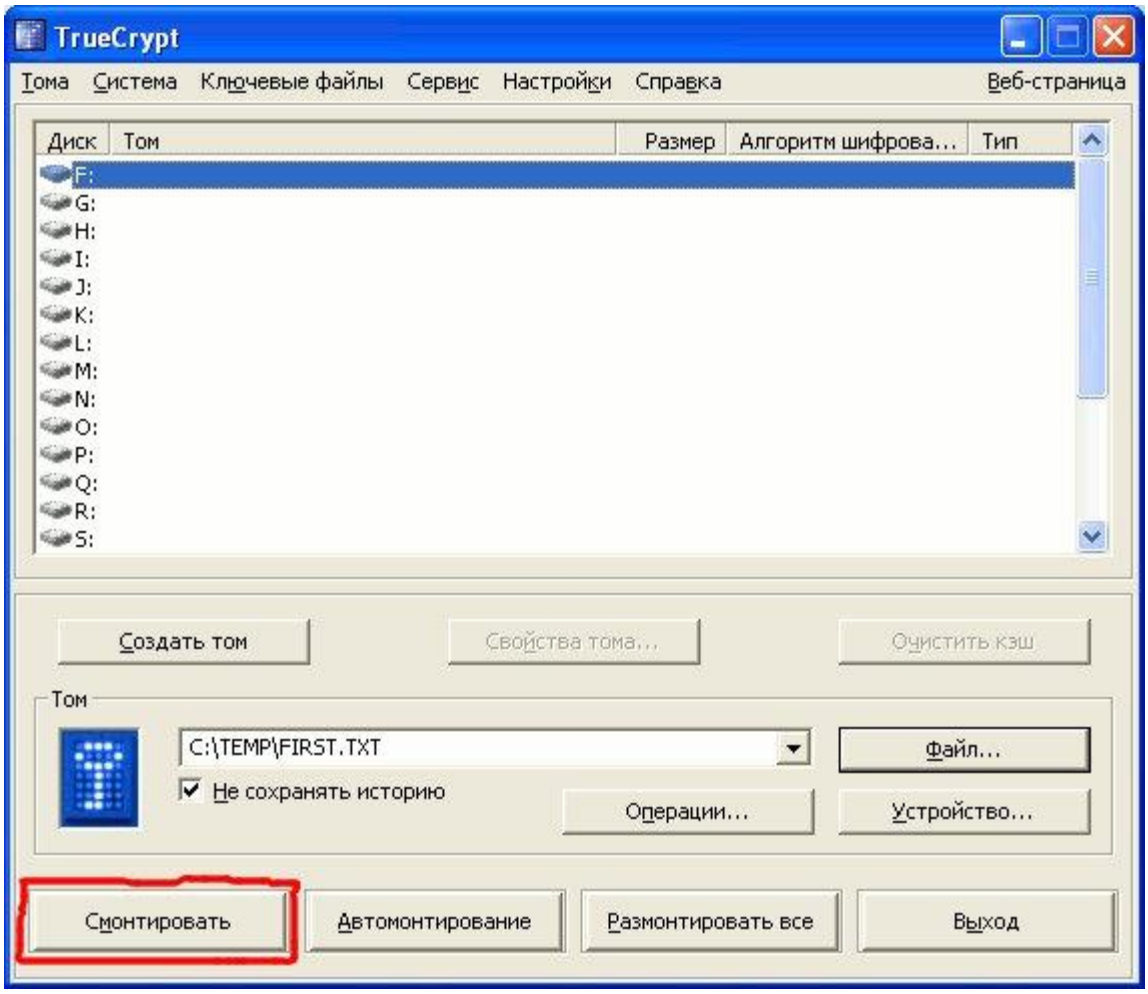


Рис. 3.33.

- В следующем окне нажмите кнопку "Параметры..."(рис. 3.34).

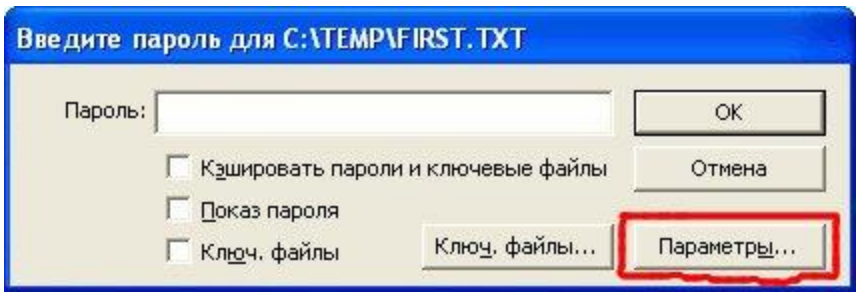


Рис. 3.34.

- В открывшемся окне установите флажок "Защитить скрытый том от повреждения при записи во внешний том", введите пароль для скрытого тома, укажите ключевые файлы для скрытого тома и нажмите кнопку "ОК"(рис. 3.35).

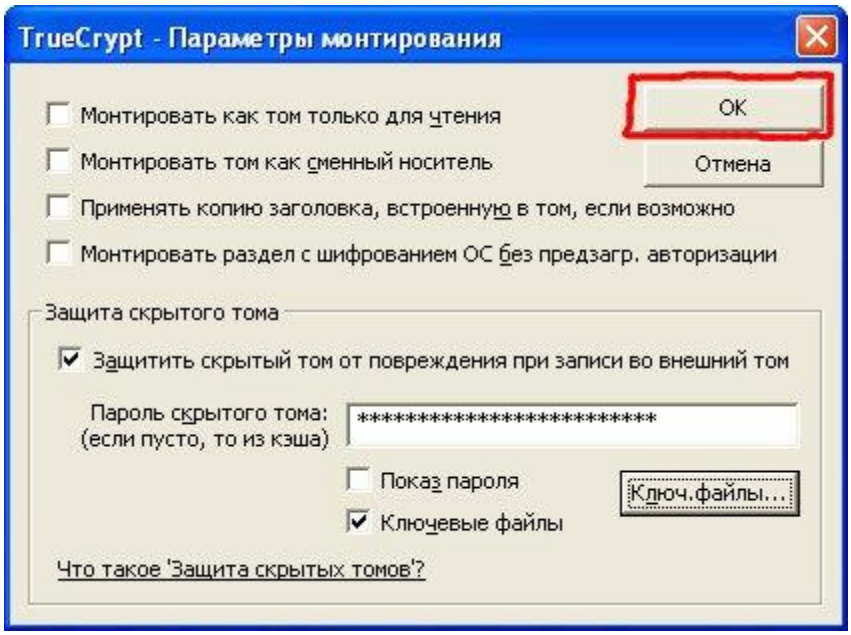


Рис. 3.35.

- Вы вернетесь к предыдущему окну. Введите пароль для внешнего тома, укажите ключевые файлы для него и нажмите кнопку "OK"(рис. 3.36).

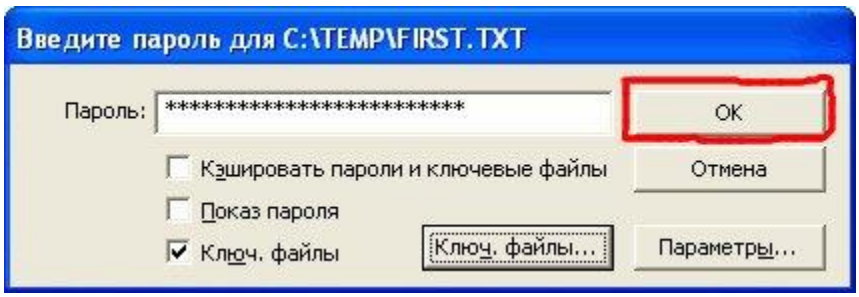


Рис. 3.36.

- Появится сообщение программы о том, что скрытый том защищен от повреждений. Нажмите кнопку "OK"(рис. 3.37).

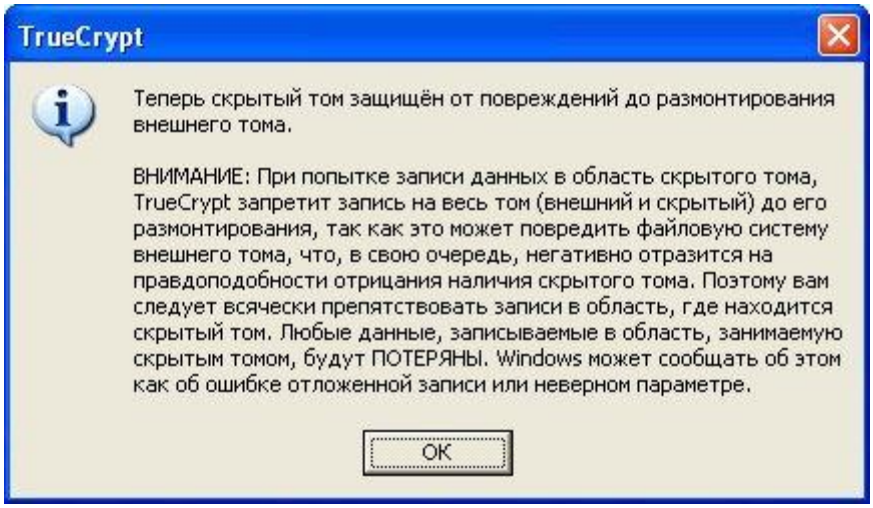


Рис. 3.37.

- После монтирования внешнего тома основное окно программы будет выглядеть следующим образом (рис. 3.38):

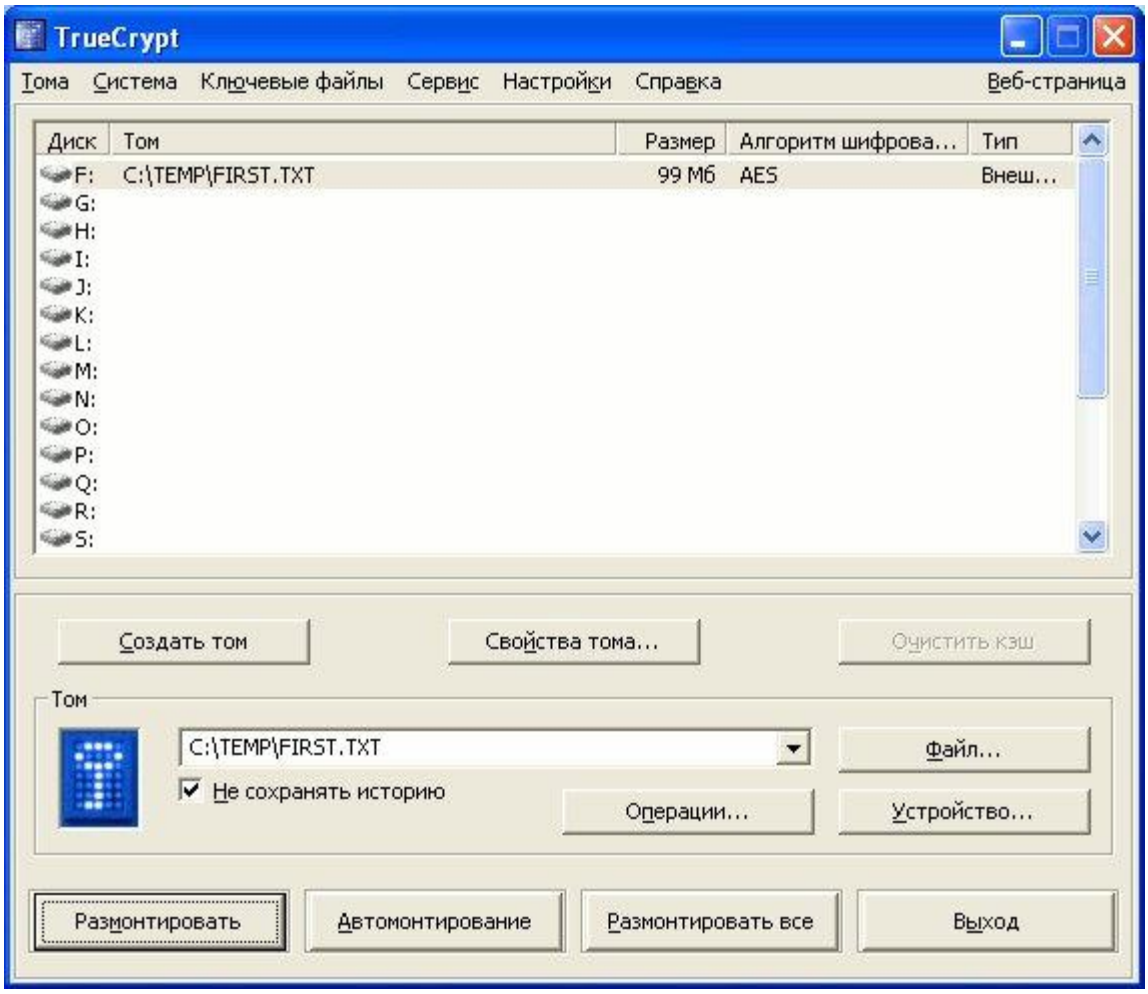


Рис. 3.38.

- Внешний том был смонтирован как локальный диск F. Чтобы открыть его, пройдите Мой компьютер - Локальный диск F. Если вы используете скрытый том, то положите на внешний том что-нибудь псевдодискредитирующее. Например, "Протоколы сионских мудрецов", как в нашем примере. Иначе сразу будет бросаться в глаза, что внешний том липовый(рис. 3.39).

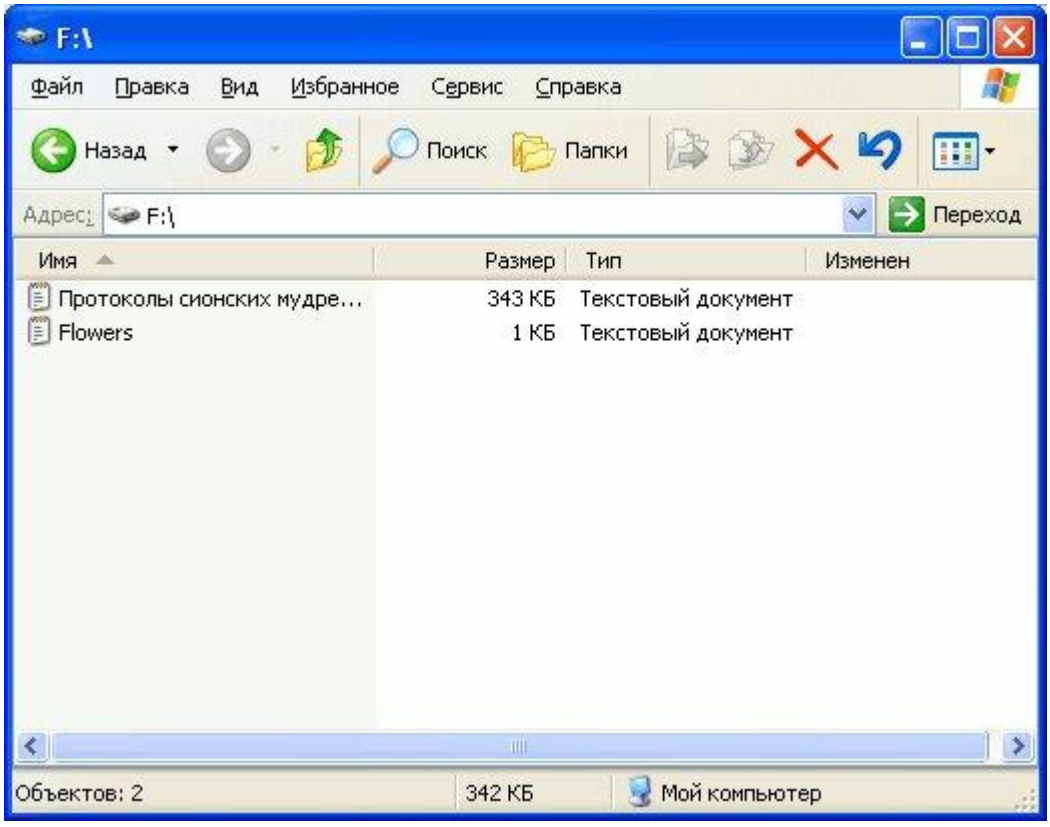


Рис. 3.39.

- Вы не сможете открыть файл FIRST.TXT, под который мы в данном примере маскируем тома TrueCrypt, пока смонтирован внешний или скрытый тома. Если же открыть этот файл после размонтирования тома, то вы увидите зашифрованную «абракадабру» (рис. 3.40).

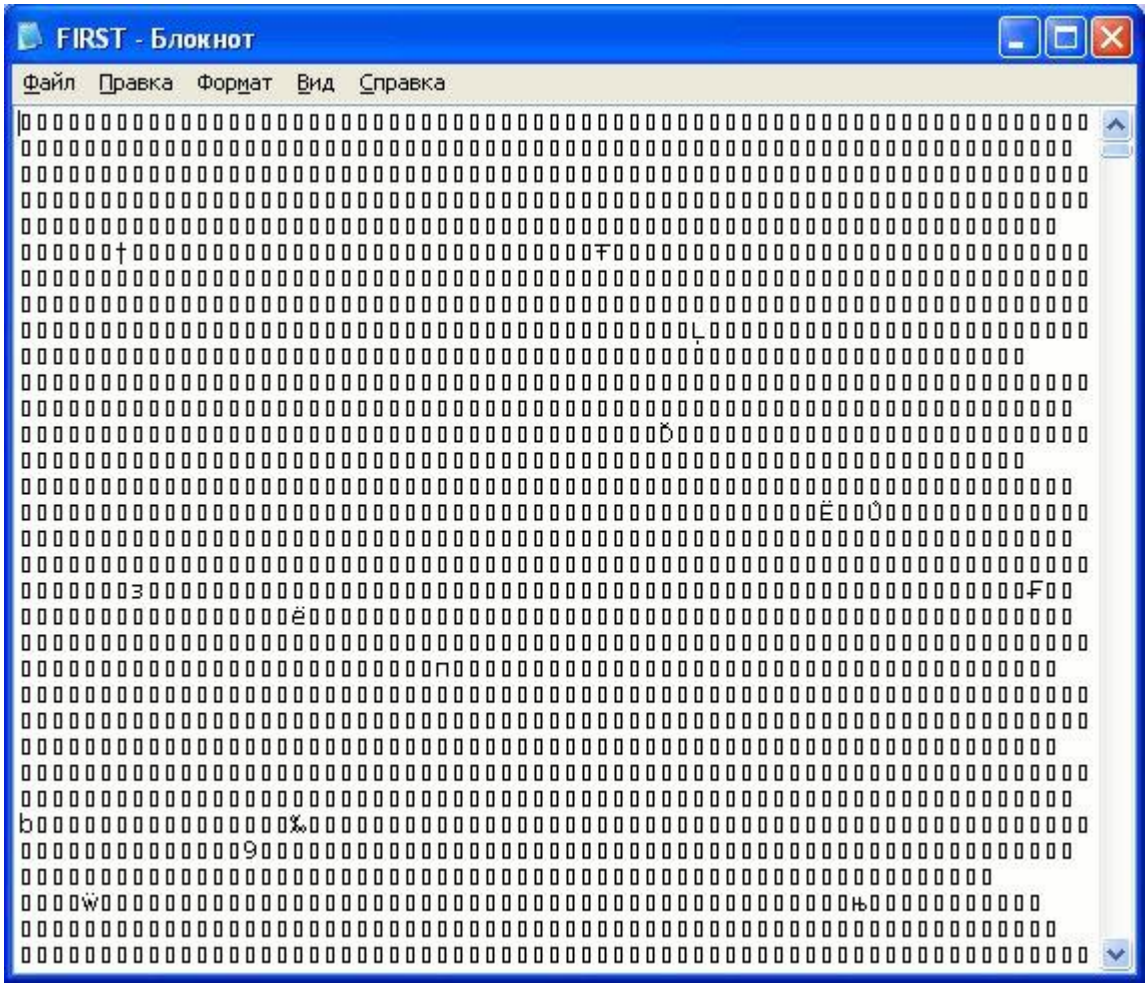


Рис. 3.40.

- Учтите, что вы должны защищать скрытый том от повреждения при каждом монтировании внешнего тома.

- Теперь смонтируйте скрытый том TrueCrypt. Для этого выполните следующие действия:

- Размонтируйте внешний том. Можете размонтировать том, нажав сочетание "горячих" клавиш (в нашем примере это Control+Shift+Alt+Z). Другой вариант - в основном окне программы нажмите кнопку "Размонтировать"(рис. 3.41).

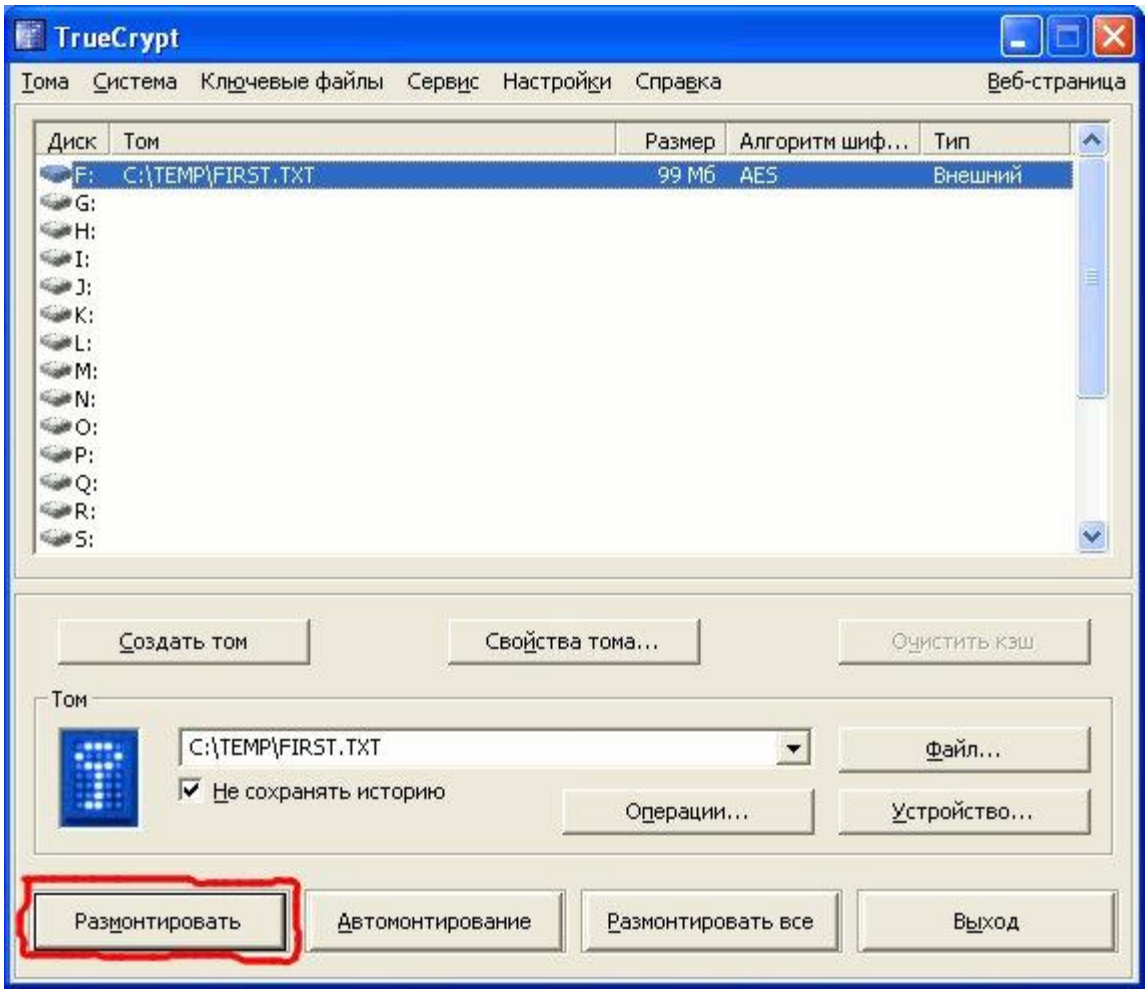


Рис. 3.41.

- Смонтируйте скрытый том TrueCrypt. Последовательность шагов такая же, как и при монтировании внешнего тома, только не нужно нажимать кнопку "Параметры..." и, естественно, нужно вводить пароль и указывать ключевые файлы уже скрытого тома, а не внешнего (рис. 3.42).

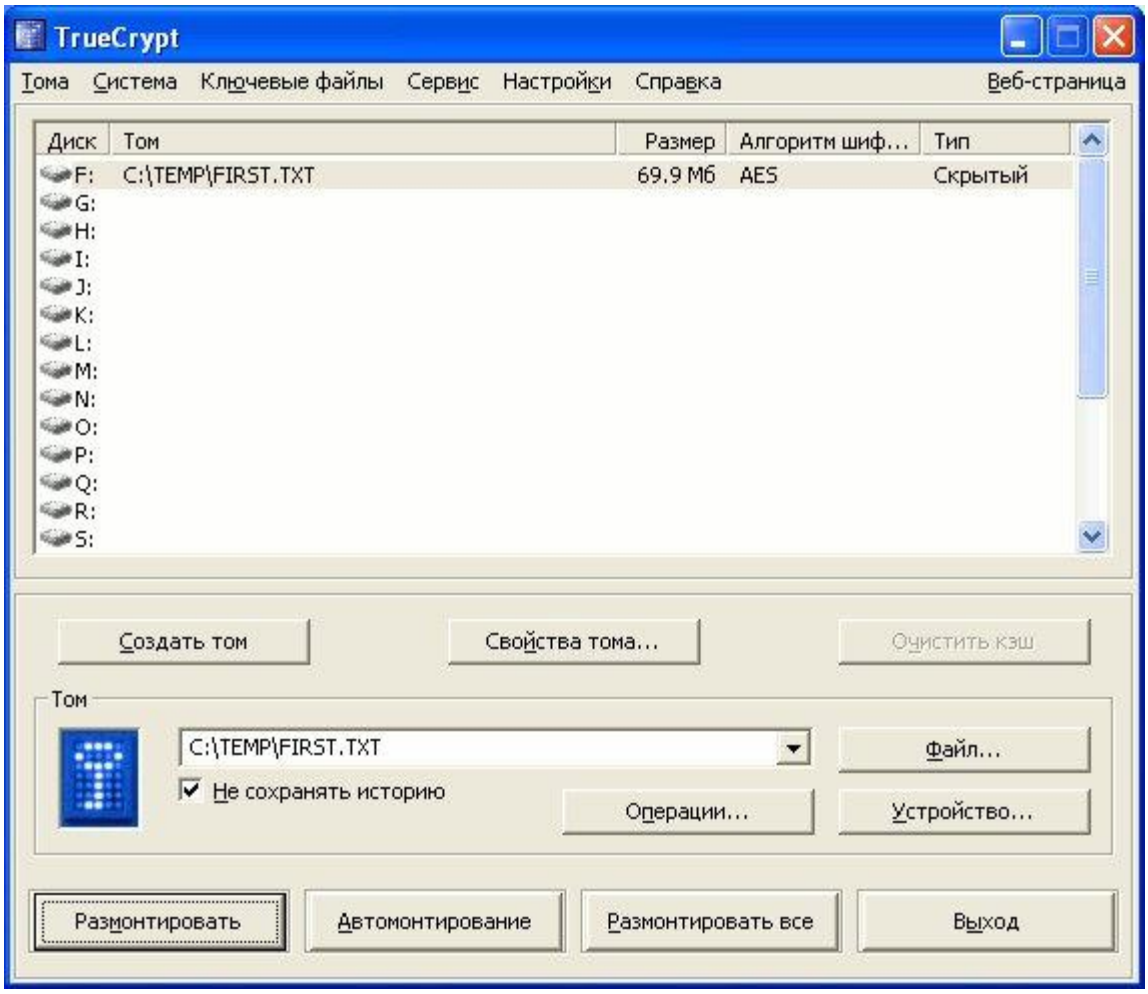


Рис. 3.42.

- Откройте этот том, смонтированный в нашем примере как локальный диск F. Теперь на скрытый том вы можете добавить по-настоящему конфиденциальную информацию. Вы видите, что скрытый том содержит другие файлы, нежели внешний (рис. 3.43).

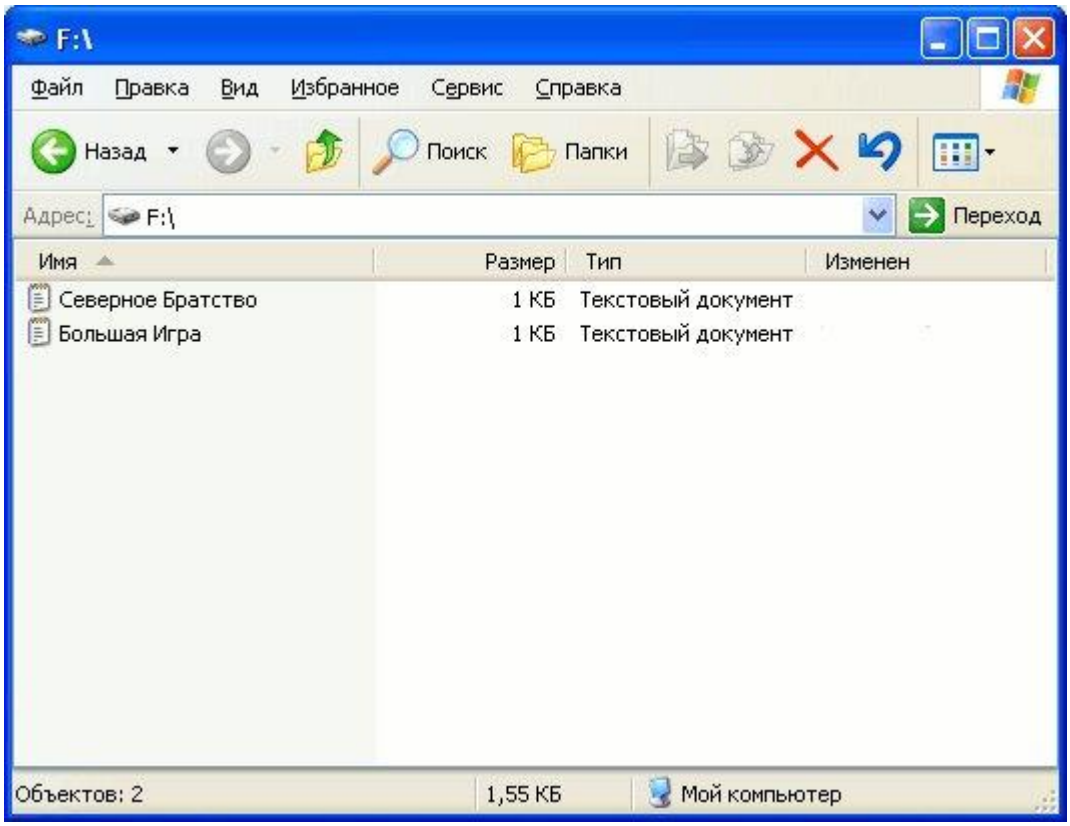


Рис. 3.43.

3. Содержание отчета

1) титульный лист;
2) формулировку цели работы;
3) описание результатов выполнения;
4) выводы, согласованные с целью работы.

ЛАБОРАТОРНАЯ РАБОТА №4.

Изучение функциональных возможностей программы- анализатора сетевого трафика Wireshark.

Цель работы: Научиться пользоваться основными функциональными возможностями программы *Wireshark.*

Wireshark представляет собой анализатор сетевого трафика с графическим интерфейсом. Программа позволяет просматривать и анализировать пакеты, полученные из сетевого интерфейса или ранее собранного файла

1. Порядок выполнения работы:

1. Запустите виртуальную машину *Ubuntu.* Определить настройки протокола *TCP/IP* Вашего компьютера с помощью команды *ifconfig.* Сделайте экранный снимок сетевых настроек (рис. 4.1).

```
Файл Правка Вид Поиск Терминал Справка
mtucivb@mtucivb-VirtualBox:~$ ifconfig
eth6      Link encap:Ethernet  HWaddr 08:00:27:b4:24:a6
          inet addr:192.168.100.209  Bcast:192.168.100.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feb4:24a6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:42 errors:0 dropped:0 overruns:0 frame:0
          TX packets:95 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6842 (6.8 KB)  TX bytes:13909 (13.9 KB)

lo        Link encap:Локальная петля (Loopback)
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2256 (2.2 KB)  TX bytes:2256 (2.2 KB)
```

Рис. 4.1.

2. Запустите из под суперпользователя *Wireshark.*
3. Выполните команду *ping localhost.* Убедитесь, что *Wireshark* отображает сетевые пакеты и что у пакетов имеются поля "источник", "приемник", "тип протокола". Сделайте экранный снимок (рисунок 1.2) главного окна *Wireshark*

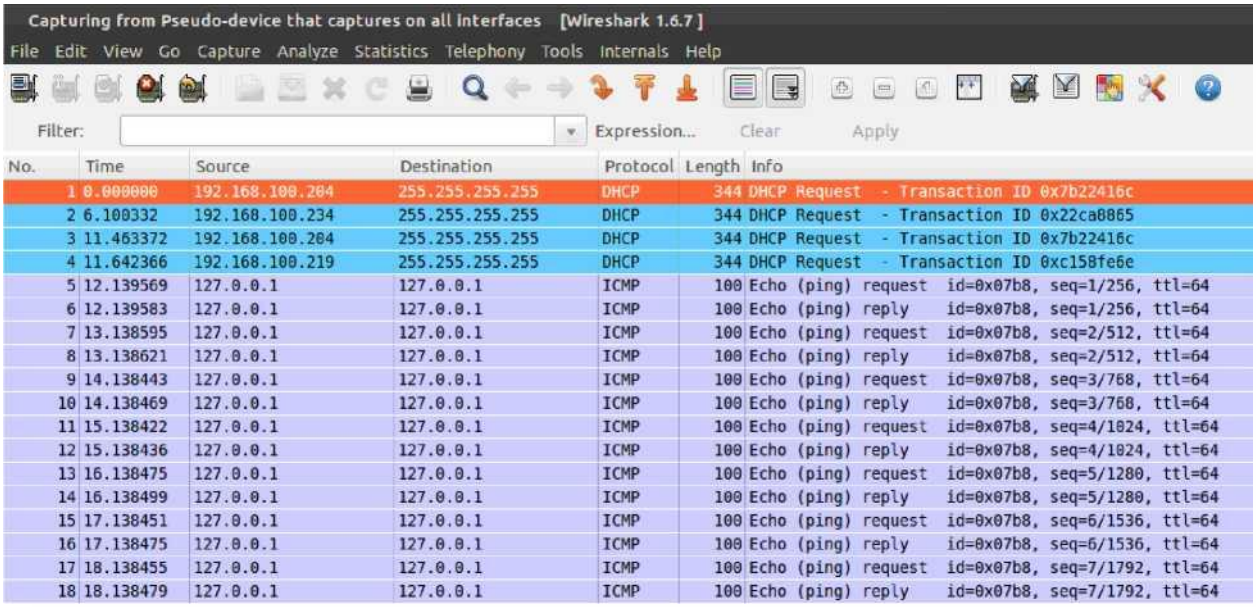


Рис. 4.2

4. Сделайте экранный снимок ICMP-пакета (рис. 4.3), отображаемого в *Wireshark.*

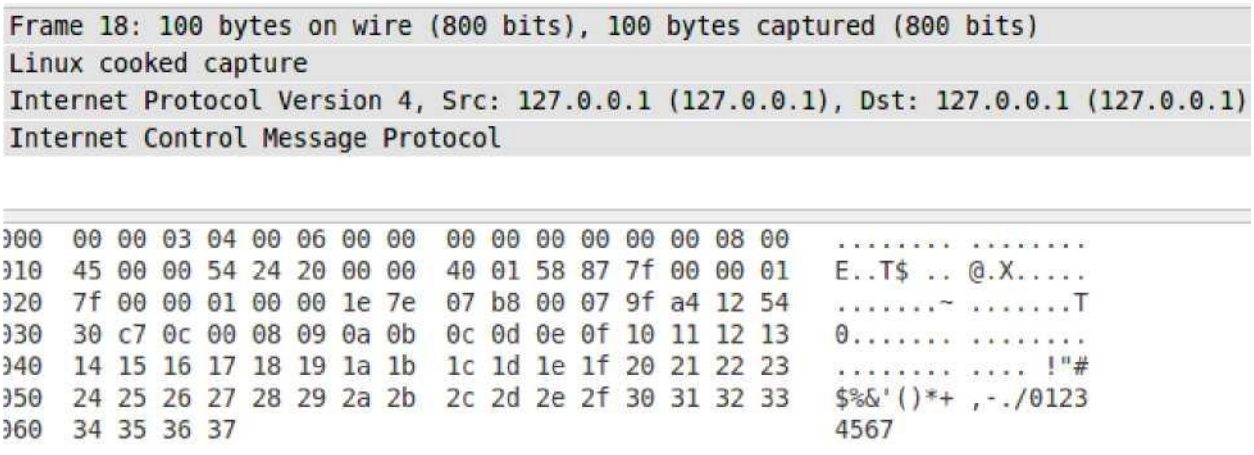


Рис 4.3

5. В браузере перейдите по адресу [http : //localhost/](http://localhost/).

6. Сделайте экранный снимок HTTP-пакета (рис. 4.4), отображаемого в *Wireshark.*

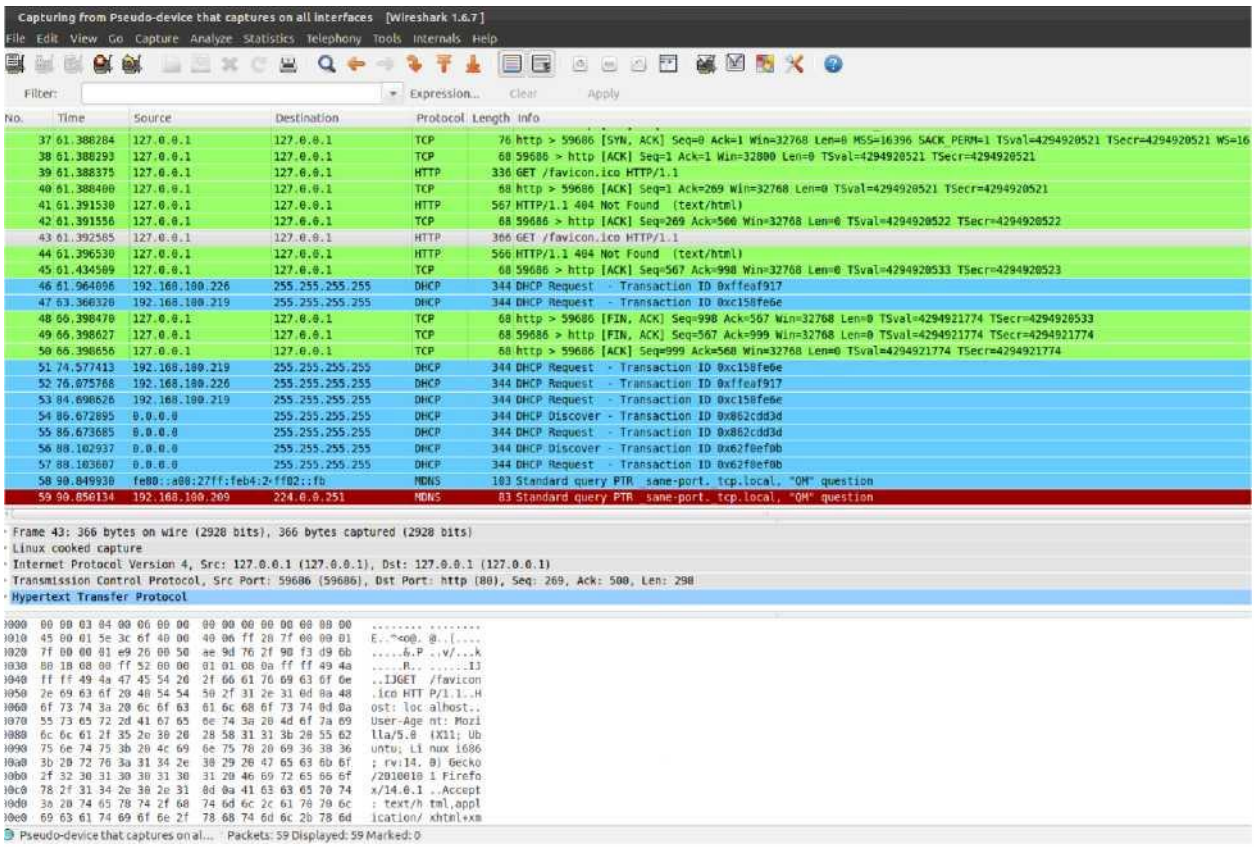


Рис. 4.4.

7. Откройте новую консоль. Вбейте *telnet localhost 80 {Enter}*. Далее введите произвольный текст и нажмите *{Enter}*

8. Найдите в *Wireshark* введенный текст и сделайте экранный снимок с найденным текстом (рис. 4.5).

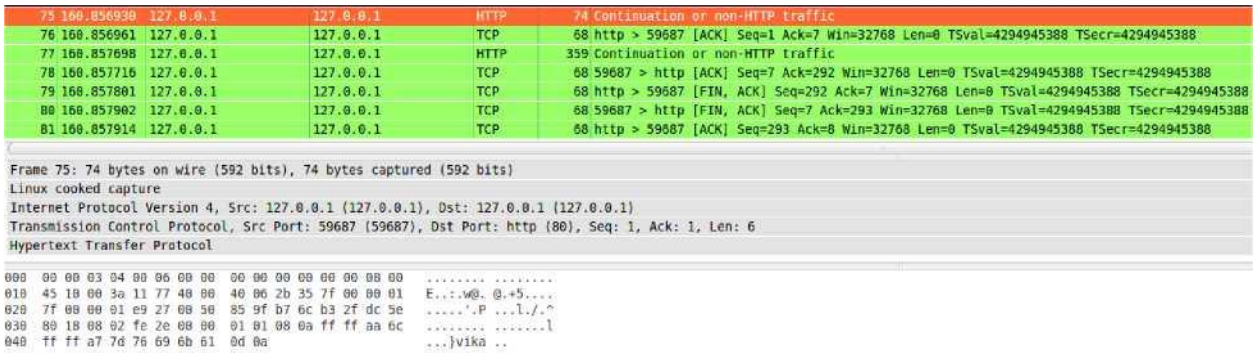


Рис. 4.5.

9. Доспользуйтесь утилитой ab, выполнив команду ab -c 100 -n 10000 http://localhost/). В Wireshark отобразите график изменения количества принятых пакетов во времени. Сделайте экранный снимок полученного результата (рис. 4.6).

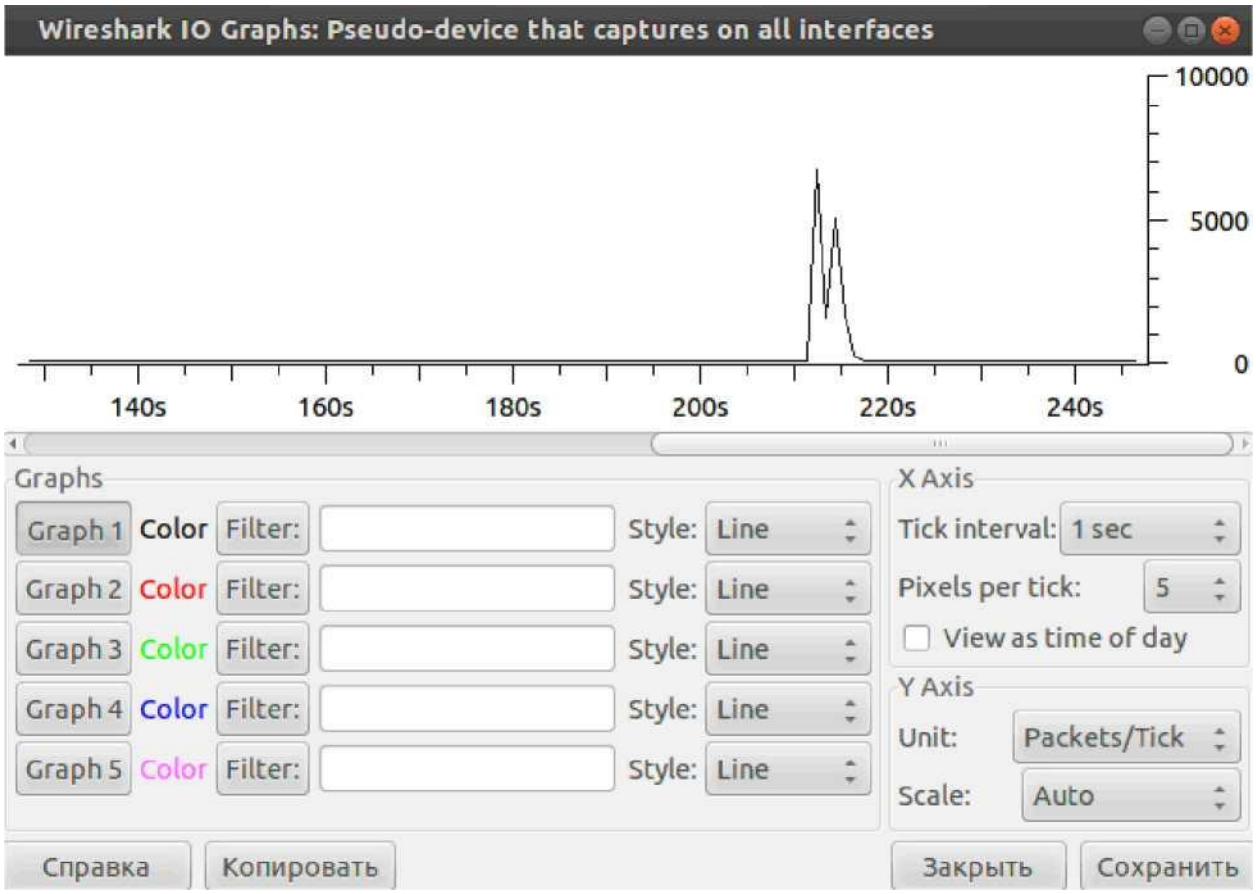


Рис. 4.6.

10. Осуществите сканирование портов с помощью команды nmap -v -A localhost. Сделайте экранный снимок (рисунок 1.7) отображаемых в Wireshark пакетов результатов. Проанализируйте сетевой трафик, генерируемый утилитой nmap. Определите какой из способов сканирования сети использует утилита nmap:

- установление полного TCP-соединения с тестируемым портом;
- сканирование в режиме половинного открытия (half-open scanning);
- сканирование с помощью FIN-сегментов.

11. Результат анализа вставьте в отчет.

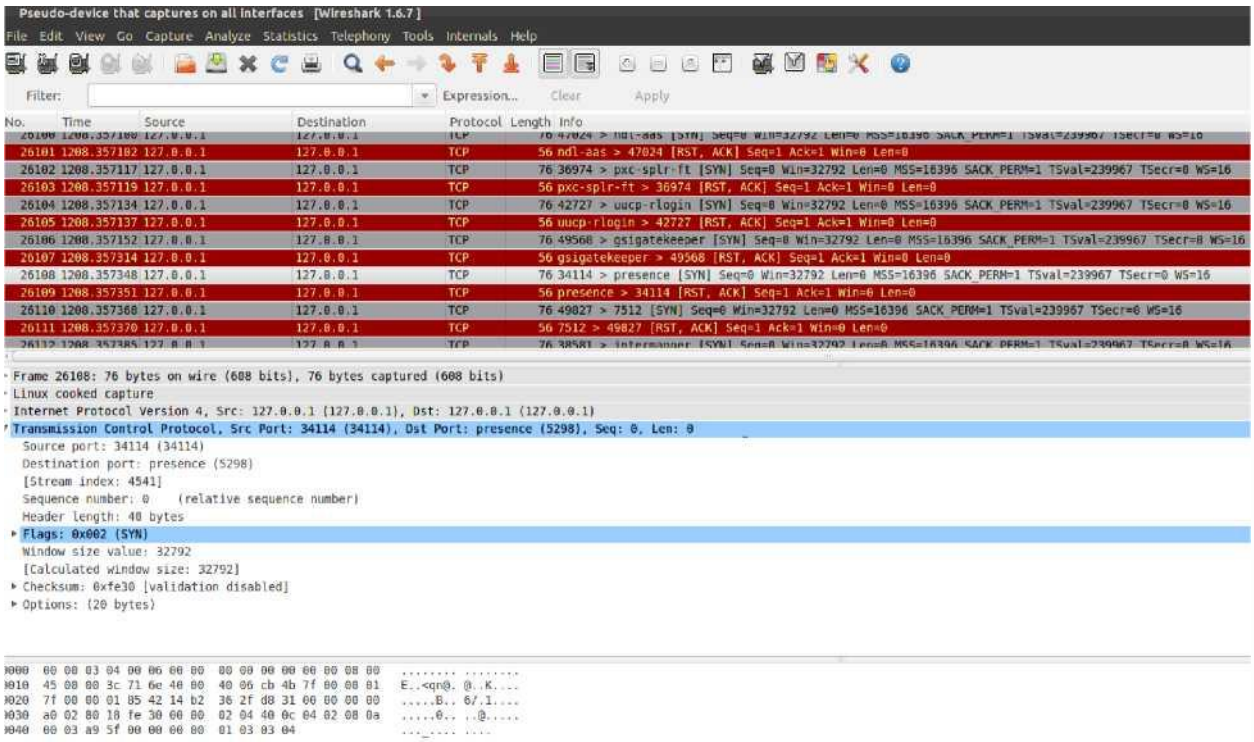


Рис. 4.7. Пример отображаемых в *Wireshark* пакетов после выполнения сканирования портов

12. Вставьте в отчет экранные снимки, демонстрирующие основные функции программы Wireshark (сохранение и восстановление перехваченных сетевых пакетов, фильтрация сетевых пакетов (рисунок 1.8), выделение цветом по правилам сетевых пакетов (рисунок 1.9), сводная статистика (рисунок 1.10), предназначение модулей, расположенных в меню Statistics и др).

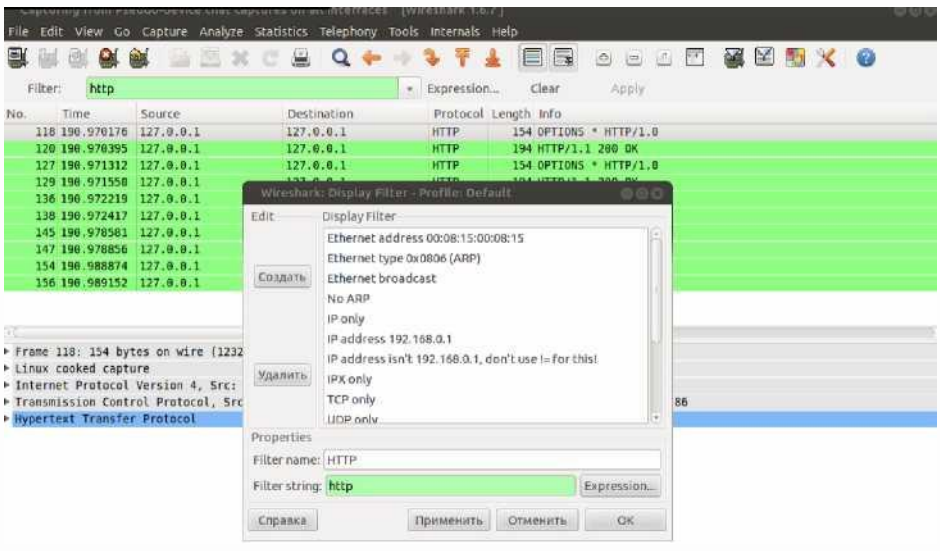


Рис. 4.8. Пример фильтрации сетевых пакетов

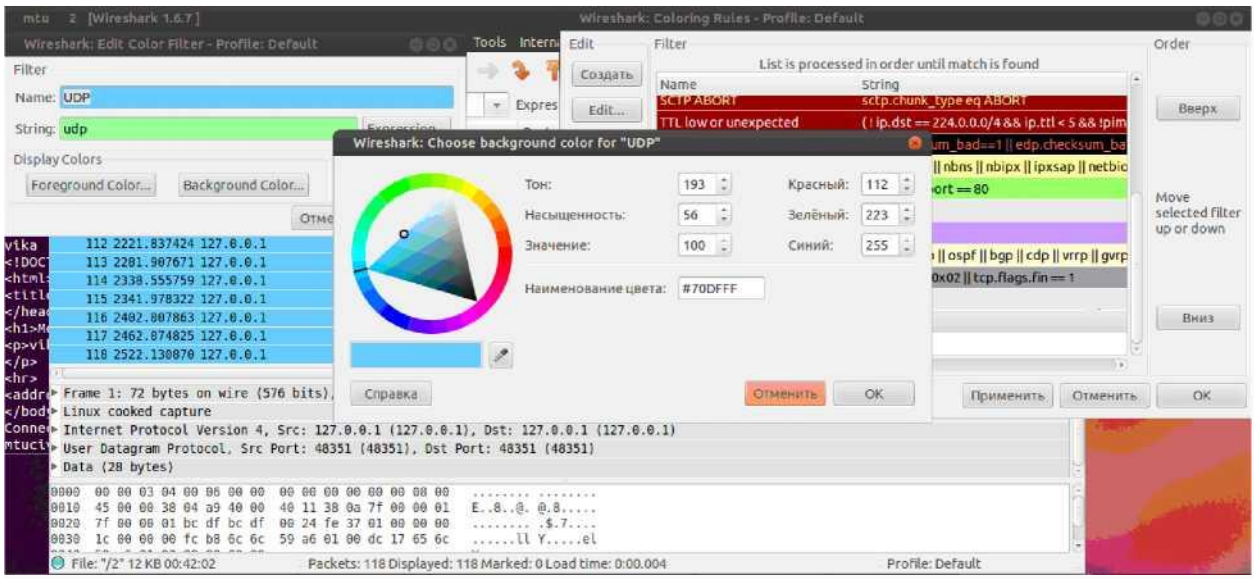


Рис.4.9. Пример настройки выделения цветом сетевых пакетов

2. Содержание отчета

1) титульный лист;

2) формулировку цели работы;

3) описание результатов выполнения;

4) выводы, согласованные с целью работы.

3. Контрольные вопросы:

1. Предназначение анализаторов сетевого трафика.
2. Что такое сниффер?
3. Способы осуществления перехвата сетевого трафика.
4. Что может быть обнаружено в результате анализа сетевого трафика?
5. Наименование и функциональные возможности программ- анализаторов сетевого трафика.
6. Основные функциональные возможности программы *Wireshark.*
7. Отличия программы *Wireshark* от *tcpdump* .
8. Для чего применяется неразборчивый (англ. promiscuous mode) режим сетевой карты?

ЛАБОРАТОРНАЯ РАБОТА №5
Настройка работы IPsec в Linux.

Цель работы: Научиться выполнять базовую настройку ***IPsec*** в ***Linux.***

1. Основные сведения

IPsec (сокращение от *IP Security)* — набор протоколов для обеспечения защиты данных, передаваемых по межсетевому протоколу IP. Позволяет осуществлять подтверждение подлинности (аутентификацию), проверку целостности и/или шифрование IP-пакетов. *IPsec* также включает в себя протоколы для защищённого обмена ключами в сети Интернет. Используется для организации vpn-соединений.

IKE — протокол, связывающий все компоненты *IPsec* в работающее целое. В частности, IKE обеспечивает первоначальную аутентификацию сторон, а также их обмен общими секретными ключами.

В работе протоколов *IPsec* можно выделить пять этапов

1. Первый этап начинается с создания на каждом узле, поддерживающим стандарт *IPsec,* политики безопасности. На этом этапе определяется какой трафик подлежит шифрованию, какие функции и алгоритмы могут быть использованы.

2. Второй этап является по сути первой фазой *IKE.* Ее цель — организовать безопасный канал между сторонами для второй фазы *IKE.* На втором этапе выполняются:

- аутентификация и защита идентификационной информации узлов,
- проверка соответствий политик *IKE Security Association (SA)* узлов для безопасного обмена ключами,
- обмен Диффи-Хеллмана, в результате которого у каждого узла будет общий секретный ключ,
- создание безопасного канала для второй фазы *IKE.*

3. Третий этап — является второй фазой *IKE.* Его задачей является создание *IPsec*

туннеля. На третьем этапе выполняются следующие функции:

- согласуются параметры *IPsec SA* по защищаемому *IKE SA* каналу, созданному в первой фазе *IKE,*
- устанавливается *IPsec* SA,
- периодически осуществляется пересмотр *IPsec* SA, чтобы убедиться в ее безопасности,
- (Опционально) выполняется дополнительный обмен Диффи-Хеллмана.

4. Рабочий этап. После создания *IPsec SA* начинается обмен информацией между узлами через IPsec-туннель, используются протоколы и параметры, установленные в SA.

5. Прекращают действовать текущие *IPsec SA.* Это происходит при их удалении или при истечении времени жизни (определенное в *SA* в байтах информации, передаваемой через канал, или в секундах), значение которого содержится в *SAD* на каждом узле. Если требуется продолжить передачу, запускается фаза два *IKE* (если требуется, то и первая фаза) и далее создаются новые *IPsec SA.* Процесс создания новых *SA* может происходить и до завершения действия текущих, если требуется непрерывная передача данных.

2. Порядок выполнения:

1. Установить утилиты для работы с *IPsec,* выполнив команду: *apt-get install ipsec-tools*2.

2. Настроить файл /etc/ipsec-tools.conf на первом узле. Пример настройки:

```
# Configuration for 192.168.1.100

# Flush the SAD and SPD
flush;
spdflush;

# Attention: Use this keys only for testing purposes!
# Generate your own keys!

# AH SAs using 128 bit long keys
add 192.168.1.100 192.168.2.100 ah 0x200 -A hmac-md5
        0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 192.168.2.100 192.168.1.100 ah 0x300 -A hmac-md5
        0x96358c90783bbfa3d7b196ceabe0536b;

# ESP SAs using 192 bit long keys (168 + 24 parity)
add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc
        0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc
        0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 192.168.1.100 192.168.2.100 any -P out ipsec
           esp/transport//require
           ah/transport//require;

spdadd 192.168.2.100 192.168.1.100 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

3. Настроить файл /etc/ipsec-tools.conf на втором узле также как и на шаге 2 за исключением раздела *# Security policies.* На втором узле этот раздел примет вид (пример):

```
# Security policies
spdadd 192.168.1.100 192.168.2.100 any -P in ipsec
           esp/transport//require
           ah/transport//require;

spdadd 192.168.2.100 192.168.1.100 any -P out ipsec
           esp/transport//require
           ah/transport//require;
```

4. Установить на обоих узлах права для доступа к файлу /etc/ipsec-tools.-conf, выполнив команду sudo chmod 750 ipsec-tools.conf.

5. Выполнить на обоих узлах команду *sudo /etc/init.d/setkey start*

6. Убедиться с помощью *Wireshark* в том, что сетевой трафик между двумя настраиваемыми узлами шифруется. Сделать скриншоты.

7. Определить с помощью *Wireshark* какие используются заголовки. Сделать скриншоты заголовков.

3. Содержание отчета

1) титульный лист;

2) формулировку цели работы;

3) описание результатов выполнения;

4) выводы, согласованные с целью работы.

4. Контрольные вопросы:

1. Что такое *IPsec?*
2. Какие протоколы лежат в основе *IPsec?*
3. Для чего применяется *IPsec?*
4. На каком уровне модели *OSI* работает *IPsec?*
5. Какие *IPsec* использует заголовки. Формат заголовков.
6. Что такое *IKE?*
7. Какие алгоритмы шифрования используются в *IPsec* ?
8. Как организовать vpn-соединения на базе *IPsec* ?
9. Что такое *Racoon* ?

ЛАБОРАТОРНАЯ РАБОТА № 6.

Обнаружение ARP-spoofing атаки

Цель: ознакомиться с основными возможностями программы **Ettercap, Arpwatch**. Реализация и обнаружение ARP-Spoofing атаки .

1. Основные сведения

ARP-spoofing (ARP — poisoning) — разновидность сетевой атаки типа MITM (англ. *Man in the middle*), применяемая в сетях с использованием протокола ARP. В основном применяется в сетях Ethernet. Атака основана на недостатках протокола ARP.

При использовании в распределённой вычислительной сети алгоритмов удалённого поиска существует возможность осуществления в такой сети типовой удалённой атаки «ложный объект распределённой вычислительной системы». Анализ безопасности протокола ARP показывает, что, перехватив на атакующем узле внутри данного сегмента сети широковещательный ARP-запрос, можно послать ложный ARP-ответ, в котором объявить себя искомым узлом (например, маршрутизатором), и в дальнейшем активно контролировать сетевой трафик дезинформированного узла, воздействуя на него по схеме «ложный объект РВС».

Протокол ARP

Протокол ARP(Address Resolution Protocol или протокол разрешения адреса) — сетевой протокол, использующийся для связи IP-адреса устройства с его MAC адресом, тем самым делая возможным связь между устройствами сетевого уровня локальной и глобальной сети.

Принцип работы

Пусть у нас есть две локальных сети Ethernet, соединенных маршрутизатором, и пусть узел первой локальной сети (узел A) хочет передать узлу второй локальной сети (узлу B) пакет. По протоколу IP определяется IP адрес интерфейса маршрутизатора (IP1), к которому подключена локальная сеть 1, где находится узел A. Теперь, чтобы пакет инкапсулировать в кадр и передать на интерфейс маршрутизатора с адресом IP1, нужно знать MAC адрес этого интерфейса. Далее начинается работа протокола ARP. Протокол ARP имеет на каждом сетевом интерфейсе компьютера или маршрутизатора свою ARP-таблицу, в которой записаны соответствия между IP-адресами и MAC адресами сетевых устройств. Пусть вначале все ARP-таблицы пусты. Далее:

1. Протокол IP запрашивает у протокола ARP MAC адрес интерфейса с адресом IP1(Рис. 6.1.).

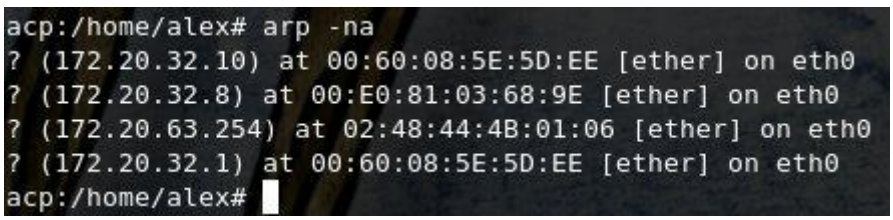


Рис. 6.1.

2. Протокол ARP проверяет свою ARP-таблицу и не находит MAC адреса, соответствующего адресу IP1
3. Тогда протокол ARP формирует ARP-запрос (смысл запроса -узнать MAC-адрес интерфейса по его IP -адресу), вкладывает его в кадр Ethernet и рассылает данный кадр широковещательно в локальной сети 1.
4. Каждый интерфейс локальной сети 1 получает ARP-запрос и направляет его своему протоколу ARP.
5. Если протокол ARP интерфейса нашел совпадение IP адреса интерфейса с IP1, то ARP генерирует ARP-ответ (В данной ситуации это ARP маршрутизатора), который направляется запрашивающему узлу. В ARP-ответе содержится MAC адрес интерфейса, на который нужно отправить кадр(в данном случае это интерфейс маршрутизатора, подключенный к локальной сети 1).
6. Далее узел А передает кадр с инкапсулированным пакетом на соответствующий интерфейс маршрутизатора.

Уязвимости ARP

Протокол ARP является уязвимым - он не проверяет подлинность ARP-запросов и ARP-ответов. А так как сетевые интерфейсы на компьютерах поддерживают самопроизвольный ARP (ARP-ответ присылается на интерфейс устройства без необходимости), то как раз в таком случае возможна атака ARP-spoofing.

Атака ARP-spoofing

Описание атаки

1. Два компьютера(узла)(рис. 6.2.) M и N в локальной сети Ethernet обмениваются сообщениями. Злоумышленник X, находящийся в этой же сети, хочет перехватывать сообщения между этими узлами. До применения атаки ARP-spoofing на сетевом интерфейсе узла M ARP-таблица содержит IP и MAC адрес узла N.Также на сетевом интерфейсе узла N ARP-таблица содержит IP и MAC узла M.

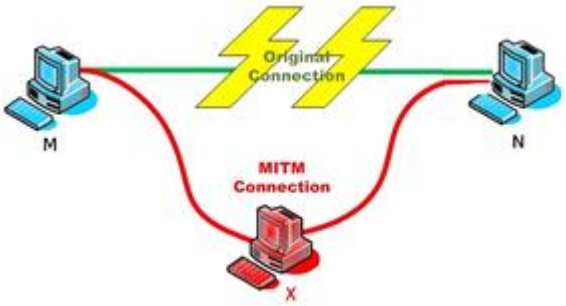


Рис. 6.2.

Зеленое соединение - связь между M и N до применения ARP-spoofing. **Красное соединение** - связь между M и N после применения ARP-spoofing (все пакеты проходят через X)

1. Во время атаки ARP-spoofing узел X (злоумышленник) отсылает два ARP ответа (без запроса) – узлу M и узлу N. ARP-ответ узлу M содержит IP-адрес N и MAC-адрес X. ARP-ответ узлу N содержит IP адрес M и MAC-адрес X.
2. Так как компьютеры M и N поддерживают самопроизвольный ARP, то, после получения ARP-ответа, они изменяют свои ARP таблицы, и теперь ARP-таблица M

содержит MAC адрес X, привязанный к IP-адресу N, а ARP-таблица N содержит MAC адрес X, привязанный к IP-адресу M.

3. Тем самым атака ARP-spoofing выполнена, и теперь все пакеты(кадры) между M и N проходят через X. К примеру, если M хочет передать пакет компьютеру N, то M смотрит в свою ARP-таблицу, находит запись с IP-адресом узла N, выбирает оттуда MAC-адрес (а там уже MAC-адрес узла X) и передает пакет. Пакет поступает на интерфейс X, анализируется им, после чего перенаправляется узлу N.

Инструменты для выполнения ARP-спуфинга

- Ettercap,
- Cain & Abel,
- dsniff,
- arp-sk,
- ARP Builder,
- AyCarrumba.
- Intercepter-ng
- Simsang

Инструменты и методы для обнаружения и предотвращения ARP-спуфинга

- **arpwatch,**
- **BitCometAntiARP**

Эти программы отслеживают ARP активность на заданных интерфейсах. Могут обнаружить атаку ARP-spoofing, но не могут предотвратить ее. Для предотвращения атаки требуется вмешательство администратора сети.

Организация VLAN

Если в локальной сети есть разделение на несколько VLAN, то атака ARP-spoofing может быть применена только к компьютерам, находящимся в одном VLAN. Идеальной ситуацией, с точки зрения безопасности, является наличие только одного компьютера и интерфейса маршрутизатора в одном VLAN. Атака ARP-spoofing для такого сегмента невозможна.

Использование статического ARP

Можно избежать атаки ARP-spoofing путем настраивания ARP-таблицы вручную. Тогда злоумышленник не сможет обновлять ARP-таблицы путем посылки ARP-ответов на интерфейсы компьютеров.

Использование шифрования

Для предотвращения атаки ARP-spoofing в локальной сети можно использовать протоколы шифрования данных, для защиты передаваемой информации от злоумышленника. Например такие протоколы как PPPoE или IPSec

Выполнение ARP-spoofing'а с помощью ettercap

Пример выполнения атаки с помощью ettercap. Рассмотрим как выполнить вышеописанную атаку с помощью ettercap.

- Что происходит на машине A.
- Что происходит на машине B.
- Что происходит на машине C.

Детально рассмотрим как выполняется ARP-spoofing. В качестве инструмента будем использовать программу ettercap, однако другие инструменты для выполнения ARP-spoofing'а работают аналогичным образом.

- Машина A — hostA — 192.168.15.201 — 00:04:75:75:46:B1
- Машина B — hostB — 192.168.15.254 — 00:0A:01:D4:D1:39
- Машина C — hostC — 192.168.15.200 — 00:0A:01:D4:D1:E3

Атаку выполняет *hostC* против узлов *hostA* и *hostB*.

Установить ettercap принятым в системе способом:

```
hostC%# apt-get install ettercap
```

Выполнить атаку против узлов hostA и hostB:

```
%# ettercap -T -M arp -L log /192.168.15.201/ /192.168.15.254/
```

Опции означают:

- `-T` — использовать текстовый (консольный) интерфейс;
- `-M arp` — использовать модуль ARP-spoofing'а для выполнения атаки;
- `-L log` — записывать журнал перехвата в файлы с именем log.*;

В качестве аргументов указываются IP-адреса машин, против которых нужно выполнять атаку ARP-spoofing.

Пусть, например, в это время *узел A* обращается к *узлу B* по протоколу POP3, классическому примеру незащищённого, но очень распространённого протокола — проверяет почту.

```
hostA %# nc 192.168.15.254 110
USER user
+OK
PASS password
+OK
LIST
+OK
```

.

Данные передающиеся между клиентом *hostA* и сервером *hostB* проходят через *узел C*. Они выводятся на экран и записываются в файлы.

После того как атака завершена для выхода из **ettercap** необходимо нажать **q**. Программа отсылает ARP-пакеты для восстановления старых записей в кэше ARP узлов, чтобы они общались друг с другом напрямую.

В текущем каталоге должны появиться два файла, начинающиеся словом, указанным после ключа `-L` при вызове **ettercap**:

```
%# ls log.*
log.eci
log.ecp
```

Просмотреть их содержимое можно с помощью программы **etterlog**, входящей в пакет **ettercap**:

```
%# etterlog log.eci

etterlog NG-0.7.3 copyright 2001-2004 ALoR & NaGA
Log file version    : NG-0.7.3
Timestamp           : Thu Jun 21 12:23:11 2007
Type                : LOG_INFO
1698 tcp OS fingerprint
7587 mac vendor fingerprint
2183 known services
================================================
 IP address   : 192.168.15.201
 MAC address  : 00:04:75:75:46:B1
...
 MANUFACTURER : Sohoware
 DISTANCE     : 0
 TYPE         : LAN host
 FINGERPRINT      :
 OPERATING SYSTEM : UNKNOWN
   PORT     : TCP 110 | pop-3   []
      ACCOUNT : user
 / password
  (192.168.15.201)
================================================
```

Как видно, пароль был успешно перехвачен.

Посмотрим как на узле *hostA* (атакуемом) меняется ARP-таблица

До атаки.

```
hostA%# arp -an
? (192.168.15.254) at 00:0A:01:D4:D1:39 [ether] on eth0
? (192.168.15.200) at 00:0A:01:D4:D1:E3 [ether] on eth0
```

Во время атаки.

```
hostA%# arp -an
? (192.168.15.254) at 00:0A:01:D4:D1:E3 [ether] on eth0
? (192.168.15.200) at 00:0A:01:D4:D1:E3 [ether] on eth0
```

После атаки.

```
hostA%# arp -an
? (192.168.15.254) at 00:0A:01:D4:D1:39 [ether] on eth0
? (192.168.15.200) at 00:0A:01:D4:D1:E3 [ether] on eth0
```

Если смотреть, что происходит на интерфейсе eth0 компьютера *hostA* (через который выполняется атака), можно увидеть, что как только начинается атака, на интерфейс поступают ARP-пакеты, которые указывают, что MAC-адрес машины 192.168.15.254 изменился. Пакеты приходят постоянно. Когда атака завершена, MAC-адрес в пакета внезапно меняется на другой. А потом они вообще перестают приходить.

```
%# tcpdump -i eth0 arp
08:34:20.231680 arp reply 192.168.15.254 is-at 00:0a:01:d4:d1:e3 (oui Unknown)
08:34:21.259637 arp reply 192.168.15.254 is-at 00:0a:01:d4:d1:e3 (oui Unknown)
08:34:22.287591 arp reply 192.168.15.254 is-at 00:0a:01:d4:d1:e3 (oui Unknown)
08:34:23.315522 arp reply 192.168.15.254 is-at 00:0a:01:d4:d1:e3 (oui Unknown)
08:34:32.463255 arp reply 192.168.15.254 is-at 00:0a:01:d4:d1:39 (oui Unknown)
08:34:33.491040 arp reply 192.168.15.254 is-at 00:0a:01:d4:d1:39 (oui Unknown)
08:34:34.514988 arp reply 192.168.15.254 is-at 00:0a:01:d4:d1:39 (oui Unknown)
```

Точно также можно перехватить и прослушать телефонный звонок - VoIP-sniffing.

Такая техника гарантирует, что таблица ARP на жертвах будет восстановлена и никто не заметит атаки.

Обнаружение arp-spoofing с помощью arpwatch.

Программа **arpwatch** отслеживает всю ARP-активность на указанных интерфейсах. Когда она замечает аномалии, например, изменение MAC-адреса при сохранении IP-адреса, или наоборот, она сообщает об этом в syslog.

Инсталляция и конфигурирование arpwatch

Процедуру инсталляции и конфигурирования **arpwatch** рассмотрим на примере системы Debian GNU/Linux.

Установка arpwatch выполняется традиционным для дистрибутива способом:

```
%# apt-get install arpwatch

0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 124kB of archives.
```

```
After unpacking 389kB of additional disk space will be used.
Get:1 http://debian.ZLO.ZLO.ZLO etch/main arpwatch 2.1a13-2 [124kB]
Fetched 124kB in 0s (177kB/s)
Selecting previously deselected package arpwatch.
(Reading database ... 22406 files and directories currently installed.)
Unpacking arpwatch (from .../arpwatch_2.1a13-2_i386.deb) ...
Setting up arpwatch (2.1a13-2) ...
Starting Ethernet/FDDI station monitor daemon: (chown arpwatch /var/lib/arpwatch/arp.dat) arpwatch.
```

После того как демон проинсталлирован, он автоматически запускается. (в других системах его, возможно, нужно будет запускать вручную.)

```
%# ps aux | grep arpwatch
arpwatch  4810  0.5  0.4   3448  2360 ?        S    08:36   0:00 /usr/sbin/arpwatch -u arpwatch -N -
p
root      4827  0.0  0.1   2852   712 pts/6    R+   08:36   0:00 grep arpwatch
```

Демон не имеет никаких конфигурационных файлов. Конфигурация arpwatch полностью определяется набором передаваемых ему ключей. В Debian GNU/Linux ключи указываются в конфигурационном файле `/etc/default/arpwatch` (в FreeBSD — в файле `/etc/rc.conf`).

При необходимости изменить конфигурацию arpwatch (в частности, заставить его прослушивать другие интерфейсы), нужно править указанный файл:

```
%# vi /etc/default/arpwatch
%# cat /etc/default/arpwatch

# Global options for arpwatch(8).
# Debian: don't report bogons, don't use PROMISC.
ARGS="-N -p"
# Debian: run as `arpwatch' user.  Empty this to run as root.
RUNAS="arpwatch"
```

Если конфигурация была изменена, демон должен быть перезапущен:

```
%# /etc/init.d/arpwatch restart
```

Когда демон запускается, он обнаруживает новые станции. Не выполняется никаких активных действий — просто прослушивается ARP-трафик. Обнаруженные узлы запоминаются; о том, что обнаружен новый узел arpwatch сообщает в syslog.

Обо всех зафиксированных им аномалиях в работе протокола ARP демон также сообщает в syslog:

```
# tail -f /var/log/daemon.log
Jun 21 08:37:08 s_all@linux2 arpwatch: new station 192.168.15.200 0:a:1:d4:d1:e3 eth0
Jun 21 08:37:08 s_all@linux2 arpwatch: new station 192.168.15.201 0:4:75:75:46:b1 eth0
Jun 21 08:37:09 s_all@linux2 arpwatch: new station 192.168.15.254 0:a:1:d4:d1:39 eth0
Jun 21 08:37:09 s_all@linux2 arpwatch: changed ethernet address 192.168.15.254 0:a:1:d4:d1:e3
(0:a:1:d4:d1:39) eth0
Jun 21 08:37:11 s_all@linux2 arpwatch: ethernet mismatch 192.168.15.254 0:a:1:d4:d1:e3
(0:a:1:d4:d1:39) eth0
```

```
Jun 21 08:37:12 s_all@linux2 arpwatch: ethernet mismatch 192.168.15.254 0:a:1:d4:d1:e3
(0:a:1:d4:d1:39) eth0
Jun 21 08:37:13 s all@linux2 arpwatch: ethernet mismatch 192.168.15.254 0:a:1:d4:d1:e3
(0:a:1:d4:d1:39) eth0
```

Здесь можно обратить внимание на строку

```
Jun 21 08:37:09 s_all@linux2 arpwatch: changed ethernet address 192.168.15.254 0:a:1:d4:d1:e3
(0:a:1:d4:d1:39) eth0
```

которая сообщает о том, что узел 192.168.15.254 изменил MAC-адрес.

Возможно, это означает, что против узла, на котором запущен arpwatch выполняется ARP-spoofing с целью перехвата трафика, которым он обменивается с узлом 192.168.15.254.

2. Порядок выполнения.

Реализуйте атаку ARP-spoofing на соседний ПК. Установив arpwatch проанализируйте файл syslog на обнаружение ARP-spoofing.

3. Контрольные вопросы.

1. Назовите причины уязвимости протокола ARP?

2. К какому классу атак относится ARP-spoofing?

3. Какие есть основные методы обнаружения ARP-spoofing?

4. Как осуществляется атака ARP-spoofing?

4. Содержание отчета

1) титульный лист;

2) формулировку цели работы;

3) описание результатов выполнения;

4) выводы, согласованные с целью работы.

ЛАБОРАТОРНАЯ РАБОТА № 7.

Сравнительный анализ эффективности антивирусных программных комплексов. Основные признаки присутствия на компьютере вредоносных программ.

Цель работы: Ознакомиться с основными принципами работы антивирусных программ.

1. Общие сведения

Антивирусная программа - специализированная программа для обнаружения компьютерных вирусов, а также нежелательных (считающихся вредоносными) программ вообще и восстановления заражённых (модифицированных) такими программами файлов, а также для профилактики — предотвращения заражения (модификации) файлов или операционной системы вредоносным кодом.

Компьютерным вирусом называется программа (некоторая совокупность выполняемого кода/инструкций), которая способна создавать свои копии (не обязательно полностью совпадающие с оригиналом) и внедрять их в различные объекты/ресурсы компьютерных систем, сетей и т.д. без ведома пользователя. При этом копии сохраняют способность дальнейшего распространения.

Вирусы можно разделить на классы по следующим признакам:

- По среде обитания вируса.
- По способу заражения среды обитания.
- По деструктивным возможностям.
- По особенностям алгоритма вируса.

а) по среде обитания вирусы можно разделить на сетевые, файловые и загрузочные. Сетевые вирусы распространяются по компьютерной сети, файловые внедряются в выполняемые файлы, загрузочные - в загрузочный сектор диска (Boot-сектор) или в сектор, содержащий системный загрузчик винчестера (Master Boot Record). Существуют сочетания - например, файловозагрузочные вирусы, заражающие как файлы, так и загрузочные сектора дисков. Такие вирусы, как правило, имеют довольно сложный алгоритм работы и часто применяют оригинальные методы проникновения в систему.

б) способы заражения делятся на резидентный и нерезидентный.

Резидентный вирус при инфицировании компьютера оставляет в оперативной памяти

свою резидентную часть, которая затем перехватывает обращение операционной системы к объектам заражения и внедряется в них. Резидентные вирусы находятся в памяти и являются активными вплоть до выключения или перезагрузки компьютера. Нерезидентные вирусы не заражают память компьютера и являются активными ограниченное время. Некоторые вирусы оставляют в оперативной памяти небольшие резидентные программы, которые не распространяют вирус.

Такие вирусы считаются нерезидентными.

в) По деструктивным возможностям вирусы можно разделить на:

1) безвредные, т.е. никак не влияющие на работу компьютера (кроме уменьшения свободной памяти на диске в результате своего распространения);

2) неопасные, влияние которых ограничивается уменьшением свободной памяти на диске и графическими, звуковыми и пр. эффектами;

3) опасные вирусы, которые могут привести к серьезным сбоям в работе;

4) очень опасные, которые могут привести к потере программ, уничтожить данные, стереть необходимую для работы компьютера информацию, записанную в системных областях памяти.

г) По особенностям алгоритма можно выделить следующие группы вирусов:

- компаньон - вирусы (companion) - это вирусы, не изменяющие файлы. Алгоритм работы этих вирусов состоит в том, что они создают для EXE-файлов файлы-спутники, имеющие то же самое имя, но с расширением .COM, например, для файла XCOPY.EXE создается файл XCOPY.COM. Вирус записывается в COM-файл и никак не изменяет EXE-файл. При запуске такого файла DOS первым обнаружит и выполнит COM-файл, т.е. вирус, который затем запустит и EXE-файл.

- черви - вирусы, которые распространяются в компьютерной сети и, так же как и компаньон - вирусы, не изменяют файлы или сектора на дисках. Они проникают в память компьютера из компьютерной сети, вычисляют сетевые адреса других компьютеров и рассылают по этим адресам свои копии. Такие вирусы иногда создают рабочие файлы на дисках системы, но могут вообще не обращаться к ресурсам компьютера (за исключением оперативной памяти). К счастью, в вычислительных сетях IBM-компьютеров такие вирусы пока не завелись.

- стелс - вирусы (вирусы-невидимки, stealth), представляющие собой весьма совершенные программы, которые перехватывают обращения DOS к пораженным файлам или секторам дисков и — подставляют вместо себя незараженные участки информации. Кроме этого, такие вирусы при обращении к файлам используют достаточно оригинальные алгоритмы, позволяющие обманывать резидентные антивирусные мониторы.
- полиморфик - вирусы (самошифрующиеся или вирусы-призраки, polymorphic) - достаточно труднообнаруживаемые вирусы, не имеющие сигнатур, т.е. не содержащие ни одного постоянного участка кода. В большинстве случаев два образца одного и того же полиморфик - вируса не будут иметь ни одного совпадения. Это достигается шифрованием основного тела вируса и модификациями программы-расшифровщика.
- Макро-вирусы - вирусы этого семейства используют возможности макро - языков, встроенных в системы обработки данных (текстовые редакторы, электронные таблицы и т.д.). В настоящее время наиболее распространены макро - вирусы заражающие текстовые документы редактора Microsoft Word.

Возможные симптомы вирусного поражения. Основные симптомы вирусного поражения следующие:

- замедление работы некоторых программ;
- увеличение размеров файлов (особенно выполняемых);
- появление не существовавших ранее - странных файлов;
- уменьшение объема доступной оперативной памяти (по сравнению с обычным режимом работы);
- внезапно возникающие разнообразные видео и звуковые эффекты.

Механизм работы современных антивирусов

Современный антивирус является сложным программным средством, которое должно обеспечить надежную защиту компьютерного устройства (компьютера, карманного компьютера или нетбука) от различных вирусов (зловредных программ). Общая схема антивируса представлена на рисунке 7.1.

Рис. 7.1

Как видно из схемы, антивирус состоит из следующих частей:

1. Модуль резидентной защиты
2. Модуль карантина
3. Модуль "протектора" антивируса
4. Коннектор к антивирусу-серверу
5. Модуль обновления
6. Модуль сканера компьютера

Модуль резидентной защиты является основным компонентом антивируса, находящийся в оперативной памяти компьютера и сканирующий в режиме реального времени все файлы, с которыми осуществляется взаимодействие пользователя, операционной системы или других программ. Слово "резидентный" означает "невидимый", "фоновый". Резидентная защита проявляет себя только при нахождении вируса. Именно на резидентной защите основывается главный принцип антивирусного ПО — предотвратить заражение компьютера. В ее состав входят такие компоненты, как активная защита (сравнение антивирусных сигнатур со сканируемым файлом и выявление известного вируса) и проактивная защита (совокупность технологий и методов, используемых в антивирусном программном обеспечении, основной целью которых является предотвращение заражения системы пользователя, а не поиск уже известного вредоносного программного обеспечения в системе).

Модуль карантина является модулем, который отвечает за помещение

подозрительных файлов в специальное место, именуемое карантином. Файлы, перемещенные в карантин, не имеют возможности выполнять какие-либо действия (они заблокированы) и находятся под наблюдением антивируса. Антивирус принимает решение поместить файл на карантин при обнаружении в файле признака вирусной деятельности (при этом сам файл с точки зрения антивируса вирусом в этом случае не является, просто файл является потенциальной угрозой), либо если файл действительно заражен вирусом, но его необходимо излечить, а не удалять целиком (например, важный документ пользователя, в который попал вирус). В последнем случае файл будет помещен в карантин для последующего излечения от вируса (если же антивирус не сможет вылечить файл, его придется удалить, либо оставить, в надежде на то, что с новым обновлением антивирус сможет вылечить этот файл). Обычно карантин создается в особой папке антивирусной программы, которая изолирована от каких-либо действий, кроме действий со стороны антивируса.

Модуль протектора антивируса является модулем, который защищает антивирус от стороннего вмешательства со стороны различных программных средств. Этот модуль является защитником антивируса. Часто вирусы хотят стереть антивирус или предотвратить его работу путем блокировки антивируса. Модуль протектора антивируса не даст это сделать. Впрочем, не все современные антивирусы снабжены качественными протекторами. Некоторые из них ничего не могут сделать против современных вирусов, а вирусы в свою очередь могут спокойно и беспрепятственно полностью стереть антивирус. Также появились вирусы, которые имитируют удаление антивируса со стороны пользователя, то есть протектор антивируса считает, что сам пользователь по каким-либо причинам хочет удалить антивирус, и поэтому не препятствует этому, хотя на самом деле это деятельность вируса. В настоящее время антивирусные компании стали более серьезно подходить к выпуску протекторов, и становится очевидно, что если антивирус не будет иметь хороший протектор, его эффективность в борьбе с вирусами будет очень мала.

Коннектор к антивирусу-серверу является важной частью антивируса. Коннектор служит для соединения антивируса к серверу, с которого антивирус может скачать актуальные базы с описанием новых вирусов. При этом соединение должно проходить по специальному защищенному Интернет каналу. Это очень важный момент, так как злоумышленник может подложить неверные антивирусные базы с лживым описанием вирусов, если антивирус будет соединяться с сервером по незащищенному Интернет-каналу. Также в современных антивирусах коннектор служит еще и для соединения к специальному серверу, который управляет антивирусом. Подобное соединение изображено на рисунке 7.2.

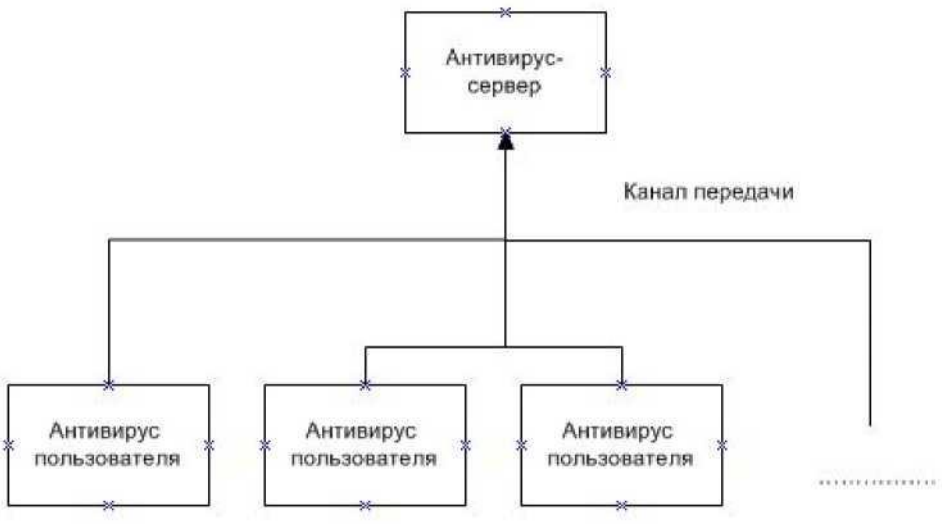


Рис. 7.2

Как видно из рисунка, коннектор позволяет соединять множество антивирусов пользователей с единым антивирусом-сервером, с которого антивирусы пользователя могут скачивать обновления, а также если на стороне антивируса пользователя возникли какие-либо неразрешимые проблемы, то антивирус-сервер будет удаленно их решать (например, у антивируса пользователя стал неисправен какой-либо из модулей и антивирус-сервер предоставит этот модуль отдельно для скачивания). В этом случае также очень важную роль играет защищенность канала передачи (канала связи) информации. Со стороны злоумышленников стала применяться интересная практика, в результате которой захватывается контроль над самим каналом передачи информации, и фактически злоумышленник становится управляющим для антивирусов пользователя (для всех или частично, в зависимости от того, какой именно участок канала передачи будет перехвачен злоумышленником). В свою очередь, создатели антивирусов стали зашифровывать данные

на канале информации, чтобы злоумышленник не мог получить к ним доступ и как-либо завладеть ими.

Модуль обновления отвечает за то, чтобы обновление антивируса, его отдельных частей, а также его антивирусных баз прошло правильно. В современной практике создания антивирусов стала применяться следующая идея: модуль обновления также должен определять подлинные или нет антивирусные базы скачивает сам модуль. Подлинность при этом может проверяться различными методами - от проверок контрольной суммы файла с базами до поиска внутри файла с базами специальной метки, которая говорит о том, что этот файл является подлинным. Подобные действия стали вводиться после того, как участились случаи подмены антивирусных баз со стороны злоумышленников.

Модуль сканера компьютера является, пожалуй, самым старым модулем в современных антивирусах, так как раньше антивирусы состояли только из этого модуля. Этот модуль отвечает за то, чтобы сканировать компьютер на наличие вирусов, если этого будет требовать пользователь компьютера. Сам модуль при сканировании компьютера использует антивирусные базы, которые были добыты с помощью модуля обновления антивируса. Если сканер найдет, но не справится с вирусом сразу же, то он поместит файл с вирусом в карантин. Потом, впоследствии, модуль сканера компьютера может связаться через коннектор с антивирусом-сервером и получить инструкции по обезвреживанию зараженного файла. Следует отметить, что модуль сканера компьютера предназначен для профилактики компьютера от вирусов, так как основную защиту представляет модуль резидентной защиты. В модуле сканера компьютера используются только антивирусные базы, в которых четко описаны вирусы. Различные элементы проактивной защиты (например, эвристика) не используются в модуле сканера компьютера. Обычно создатели вирусов не строят специальную защиту для своих вирусов от модулей сканера компьютера, так как знают, что пользователь не часто проверяет компьютер сканером, и этого промежуточного времени от проверки до проверки хватит, чтобы украсть персональные данные пользователя.

Надежность современных антивирусных программ

Прежде всего, необходимо уяснить то, что абсолютно надежных антивирусных программ не бывает в принципе из-за изменчивой природы вирусов. Если говорят, что

какой-либо антивирус является лучшим и защищает абсолютно от всех существующих вирусов, то это с большей долей вероятностью является рекламным ходом антивирусной компании, либо антивирус защищает от всех вирусов только в короткий промежуток времени, так как вирусы по всему миру выходят постоянно и неизвестно, какой именно вирус завтра будет бушевать на просторах сети Интернет. Причина такой непостоянной защищенности, которую предоставляют антивирусы проста - сначала должен появиться вирус, а только потом уже защита от него. И хотя в современных условиях антивирусы достаточно быстро реагируют на появление вирусов и уже в течение часа могут предоставить сигнатурную базу с описанием вируса и его лечением, все равно остается определенный промежуток времени, когда неизвестно, как лечить этот новый вирус. Частично проблема решается путем эвристического подхода, который позволяет блокировать вирусы, не попавшие в сигнатурные базы, но и он не всегда позволяет противостоять новым вирусам. Зачастую бывает так, что вирусописатели специально для кражи определенного типа данных с определенного места (например, кража всех логинов и паролей к онлайн- сервису) пишут определенный специально направленный вирус, который не сможет обнаружить антивирус. В этом случае такой вирус может ходить от онлайн-сервиса к онлайн-сервису до тех пор, пока он не попадет в руки антивирусной лаборатории, которая исследует вирус и занесет его в сигнатурную базу. Также вирусописатели изменяют свой подход к написанию вирусов в сторону улучшения их внедрения на компьютер пользователя. Это делается с помощью тех самых каналов связи, через которые антивирус получает сигнатурные базы, либо инструкции к некоторым действиям. Вирус просто блокирует эти каналы, и антивирус остается рабочим, но без обновлений и правильных сигнатурных баз. Это является благодатной почвой для вирусов, и в результате иногда случаются вирусные эпидемии на компьютерах пользователей. Во избежание этого антивирусные компании стараются максимально защищать каналы связи различными шифровальными методами, а также другими методами, которые позволили бы защитить антивирус от постороннего вмешательства вирусов в свою работу. Но, как бы ни старались максимально улучшить свое детище антивирусные компании, еще ни один антивирус не может уничтожать 100% угроз на протяжении долгого промежутка времени. Об этом говорит практика, а также различные аналитические центры. В среднем, самый лучший

антивирус может сохранить свое преимущество перед остальными антивирусами недолгий промежуток времени, а также он будет справляться только с 70-80% от всех вирусных угроз, которые существуют (при этом во внимание берутся только распространенные угрозы. Вирусы, которые пишутся для определенной атаки на компьютер и фактически являющиеся одноразовыми, в этой статистике обычно не учитываются). В теории решением такой проблемы было бы использование одновременно нескольких антивирусов на одном компьютере, но это зачастую невозможно сделать из-за того, что антивирусы в этом случае будут конфликтовать между собой, усугубляя тем самым положение самого компьютера в свете защиты от вирусов. Однако существуют некоторые независимые лаборатории, в которых стоит сразу же большое количество антивирусов. Такие лаборатории могут выполнять анализ одного файла сразу многими антивирусами. Результат от такого анализа будет более точен, если тот же файл отправить на анализ только одному антивирусу. Но, несмотря на существование таких лабораторий, проблема 100% защиты от вирусов все же остается, так как лаборатория не может обеспечить постоянный надзор над компьютером, ей можно отсылать только файлы, которые вызывают подозрение у пользователя.

Существует также мнение, которое гласит, что антивирус никогда не сможет на 100% справляться с вирусами, потому что он является компонентом операционной системы (ее прикладной программой, хотя антивирусы и относят к системному программному обеспечению), а не ее непосредственной частью. Раньше эта проблема выглядела особенно остро, когда антивирусы и операционные системы постоянно конфликтовали между собой, причем, если пользователь хотел удалить антивирус, ему приходилось в большинстве случаев переустанавливать операционную систему, так как в ней начинались постоянные сбои и значительное снижение скорости работы. В современном периоде этот конфликт частично исправлен, но далеко не всегда получается так, что антивирус интегрируется в операционную систему без последствий. К тому же остается та проблема, что операционная система все равно остается "королем" (выражаясь абстрактно), а антивирус ее "слугой". Другими словами, если операционная система вынесет какие-либо действия по отношению к антивирусу, то антивирус не сможет отказать операционной системе. Этим непременно пытаются воспользоваться вирусописатели при написании своих детищ. Они пишут вирус так, чтобы он, прежде всего, воздействовал на операционную систему, которая может

произвести воздействие на антивирус, чтобы отключить или удалить его. Схематически это может выглядеть следующим образом (рисунок 7.3).

Рис. 7.3

Из данной схемы можно увидеть следующее: Вирус проникает через проактивную защиту антивируса, далее он проникает через операционную систему и завладевает ею, заставляя операционную систему воздействовать на антивирус. Контролируя операционную систему, вирус может отдать ей любой приказ относительно антивируса, при этом антивирус не сможет противиться действиям операционной системы. Расширенной версией такой схемы является версия с попаданием и контролированием вируса системы BIOS (BIOS (англ. basic input/output system — "базовая система ввода-вывода") — реализованная в виде микропрограмм часть системного программного обеспечения, которая предназначается для обеспечения операционной системы API доступом к аппаратуре компьютера и подключенным к нему устройствам). В этом случае вирус будет контролировать абсолютно все действия, которые выполняются на компьютере, а также на любом компьютере, где есть BIOS. Впрочем, такие вирусы уже стали редкостью и практически не применяются в современной практике, так как проактивные защиты антивирусов не дают настолько глубоко проникнуть вирусу в компьютер пользователя.

Частыми случаями отказа в правильной работе антивирусов бывает недостаточно хорошая работа "протектора" антивируса. Обычно это происходит потому, что при написании протектора невозможно предусмотреть абсолютно все случаи возможного стороннего вмешательства в работу антивируса. Как показывают различные аналитические

источники, современные протекторы не всегда могут обеспечить защиту антивируса даже в таком простом случае, когда вирус выдает свое действие по прекращению работы антивируса за действие пользователя. Этот факт должен напоминать пользователям о том, что необходимо быть внимательными к неожиданному прекращению работы антивируса и активировать его работу вручную.

Основные моменты использования современных антивирусных программ

Безусловно, антивирусные программы предоставляют защиту от вредоносных программ достаточно хорошо, но это не значит, что пользователь ничего не должен делать для поддержания этой работы и совершенно не должен следить за работой антивируса. Пользователь должен делать следующие действия, чтобы поддерживать работу антивируса, а также сделать ее максимально эффективной:

1. Обращать внимание на результаты работы сканера
2. Регулярно делать обновления антивируса
3. Периодически проверять работу резидентной защиты
4. Регулярно проверять сканером свой компьютер
5. В случае неполадок обязательно сообщать о них в службу поддержки антивируса
6. Регулярно просматривать отчеты антивируса
7. Настроить антивирус должным образом

Рассмотрим данные пункты более подробно:

Пункт №1 (обращать внимание на результаты работы сканера). Несмотря на то, что большинство современных сканеров антивируса сами находят вирусы и решают, что с ними делать, пользователь должен обращать внимание на результаты работы сканера. Это необходимо делать потому, что иногда сканер может допустить ошибку и поместить в карантин абсолютно безвредный файл (например, файл одного из свежеустановленных драйверов). Если пользователь уверен в том, что антивирус поместил в карантин файл по ошибке, то его необходимо изъять из карантина, так как в карантине файл не может совершать каких-либо действий, и от этого может пострадать работа компьютера в целом. Чтобы уверенность была объективной, необходимо проверить действительную

принадлежность файла к лицензионному программному средству, либо к операционной системе (обычно на сайте производителя программного обеспечения можно узнать такую информацию) и если программа является свежеустановленной, то можно изъять такой файл из карантина. Правда, в некоторых случаях необходимо помнить, что это делается на страх и риск пользователя, и до конца быть уверенным, что файл является безопасным, все равно нельзя. Также бывают случаи, когда пользователь мог набирать код программного средства в текстовом редакторе, а при проверке сканером такой документ может быть помещен в карантин, если код содержит какие-либо опасные с точки зрения безопасности вставки, не считая того, что это файл документа, который не может быть исполняемым файлом по определению.

Пункт №2 (регулярно делать обновления антивируса). От соблюдения этого пункта фактически зависит безопасность персональных данных пользователя, а также эффективность работы антивируса в целом. Стоит отметить, что обновления антивируса предусмотрены самим антивирусом, то есть антивирус регулярно проверяет себе обновления, но бывает и так, что планировщик обновлений сбивается, и приходится обновлять антивирус вручную. Пользователь должен помнить этот факт и внимательно следить за обновлениями антивируса, потому что без актуальных баз антивирус не сможет качественно выполнять свои функции по защите компьютера от вирусов.

Пункт №3 (периодически проверять работу резидентной защиты). Обычно резидентная защита включена по умолчанию постоянно и не должна быть выключена в какой-либо промежуток времени. Но известны случаи, когда при установке программного обеспечения резидентная защита антивируса временно отключается с целью обеспечить плавную установку нового программного обеспечения. При этом после установки программного обеспечения необходимо проверить, была ли включена снова резидентная защита антивируса, так как бывают случаи, когда она не активируется снова. Также целесообразно периодически проверять правильность самой работы резидентной защиты в целях профилактики работы антивируса. В этом могут помочь специальные лжевирусы, которые сами по себе не приносят вреда компьютеру, но создаются специально для того, чтобы проверить правильность работы резидентной защиты. Обычно такие лжевирусы выпускают различные аналитические компании, а также независимые пользователи, которые

заинтересованы в регулярной проверке резидентной защиты антивируса. Такие "вирусы" можно скачать из сети Интернет (обычно они предоставляются открыто). Все эти меры помогут поддерживать правильность работы резидентной защиты. Если же в результате подобных проверок будет обнаружено, что резидентная защита не справляется полностью со своими функциями, необходимо связаться с создателями антивируса и описать проблему, либо вообще сменить антивирусную программу.

Пункт №4 (регулярно проверять сканером свой компьютер). Данный пункт является профилактикой заражения компьютера различными вирусами. Пользователь должен помнить, что не всегда резидентная защита антивируса может остановить абсолютно все угрозы, связанные с вирусами, и поэтому необходимо периодически устраивать проверки сканером антивируса. Такие проверки исключат возможность нахождения различных вирусов на компьютере. Проверки необходимо делать хотя бы раз в месяц и проверять компьютер необходимо полностью.

Пункт №5 (в случае неполадок обязательно сообщать о них в службу поддержки антивируса). Многие пользователи, обнаруживая какие-либо неполадки в работе антивирусной программы, пытаются сами исправить эти неполадки, даже если не совсем понимают, как именно это надо делать. А вместе с тем любая неполадка, даже самая незначительная на первый взгляд, может нести в себе элементы опасности для безопасной работы компьютера. Ведь зачастую остается неизвестной причина самой неполадки. Может быть, неполадка вызвана конфликтом операционной системы с антивирусом, может, программным обеспечением компьютера, а может и зловредной программой, которая хочет полностью отключить защиту антивируса. Поэтому при любых неполадках, даже самых незначительных, пользователь должен обращаться в службу поддержки антивируса с четким и понятным описанием проблемы, которая возникла в работе антивируса. Создатели антивируса гораздо более быстро выяснят причину неполадок и зачастую смогут дать полезные советы по обращению с их детищем. Необходимо помнить, что даже, казалось бы, безобидное временное прекращение работы резидентной защиты или кратковременное отключение обновлений антивируса может говорить о том, что на компьютере присутствует вирус, который надо как можно скорее обезвреживать.

Пункт №6 (регулярно просматривать отчеты антивируса). Это действие, которое

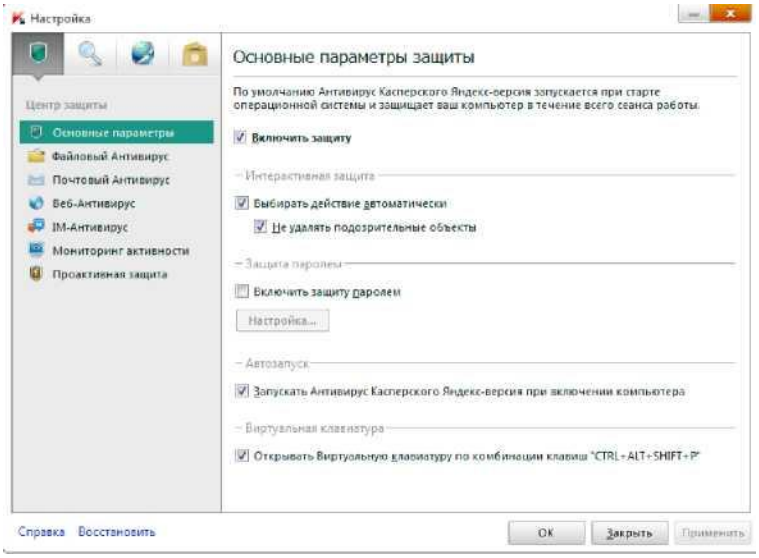
является больше профилактическим, а также позволит лучше понять, что именно происходит на компьютере пользователя. Отчеты антивируса предназначены, в первую очередь, для самих пользователей, чтобы пользователи могли наглядно увидеть работу антивируса, узнать, какие файлы заблокировала резидентная защита, определить, когда было совершено последнее обновление антивируса (и удачно ли оно было совершено). Именно на основе отчетов можно понять, правильно ли антивирус выполняет свои функции, либо в его работе начались какие-либо неполадки, сбои. Увидеть это можно простым путем - если в отчетах стали появляться сообщения о неудачах работы антивируса, о его частых ошибках, либо отказе в каких-либо действиях, то можно быть уверенным, что были постоянные сбои в работе антивируса, и что он работает неправильно. Соответственно, в таком случае необходимо обязательно сообщить об этом в службу поддержки антивируса.

Пункт №7 (настроить антивирус должным образом). От правильной настройки антивируса зависит общая защищенность компьютера. Казалось бы, это очевидный факт, но многие пользователи забывают должным образом настроить свою антивирусную программу, а используют ее так, как она поставляется ему с сайта производителя. Это неверный подход, так как настройки современного антивируса достаточно богаты и можно "подогнать" антивирусное средство именно под свой компьютер, а также не упустить различные моменты в контролировании антивирусным средством компьютера. Настройки современных антивирусов позволяют задать четкое количество полных проверок компьютера, различные исключения для файлов (чтоб нужный файл нужной программы не попал в карантин антивируса), а также сделать так, чтобы антивирус полностью удовлетворял потребностям пользователя. Основной настройке подлежит планировщик заданий антивируса - важный компонент, который позволяет делать действия (проверять обновления, проверять компьютер) в определенные отрезки времени. Настроив его, пользователь может не беспокоиться о том, что может забыть проверить компьютер или скачать обновления для антивируса. Хотя стоит упомянуть о том, что периодически необходимо все же проверять правильность работы планировщика, чтобы неожиданно не обнаружить, что антивирусная программа не обновлена или полная проверка компьютера не была произведена.

Порядок выполнения работы .

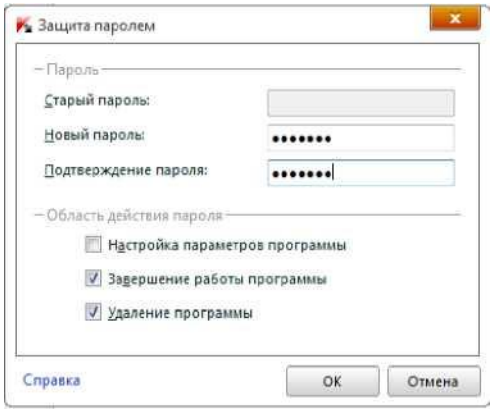
1. Запустите антивирусное программное обеспечение и перейдите в “Настройки”.

Данный пункт позволяет более детально настроить антивирусное программное обеспечение под нужды пользователя, что позволяет обеспечить защиту информации.



2. В основных параметрах включите защиту паролем на пункты:

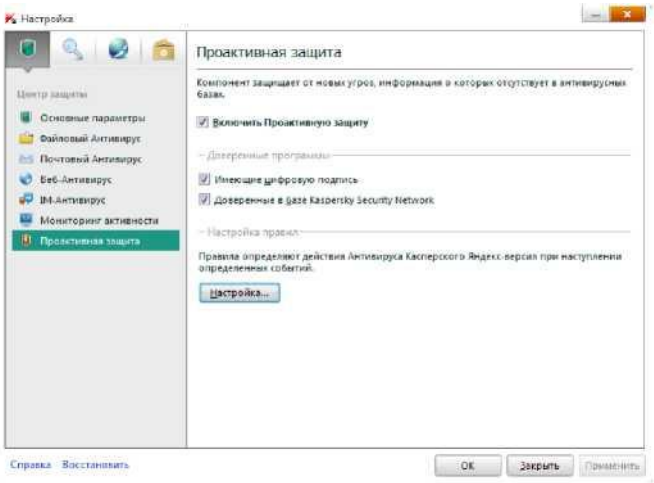
- Завершение работы программы
- Удаление программы



И теперь проверьте действие параметра путем отключения или удаления антивируса.

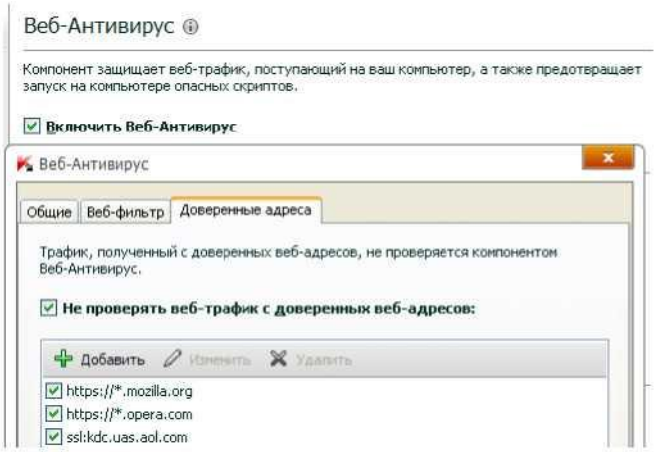
Данные параметры позволяют обезопасить антивирус от воздействия посторонних программ или пользователей.

3. Проактивная защита позволяет защитить компьютер от новых угроз, информации о которых еще нет в базах:

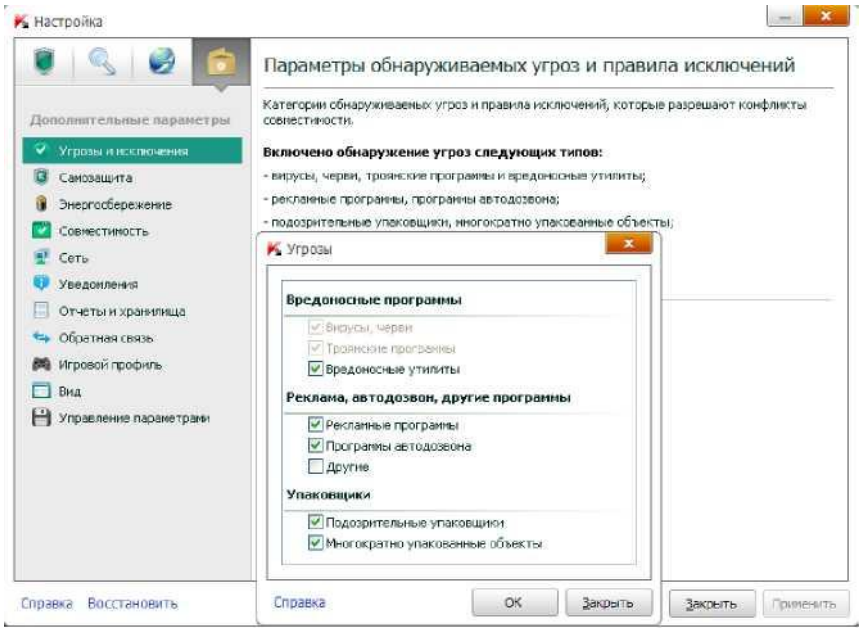


4. Мониторинг активности отслеживает работу имеющихся программ для своевременного обнаружения непредусмотренных функций:

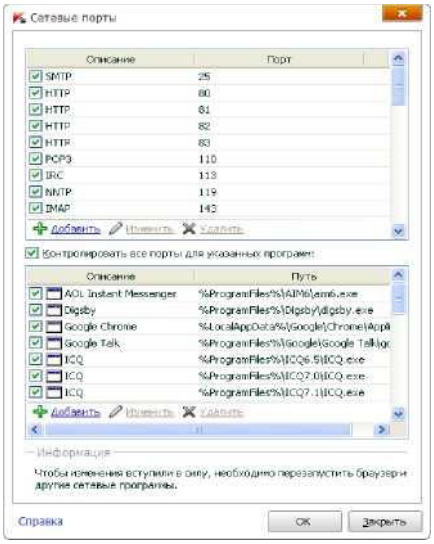
5. Веб-антивирус предусматривает управление доверенными веб-адресами:



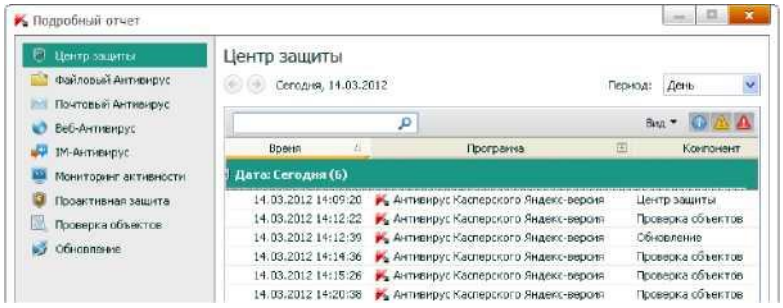
6. Возможна настройка исключений для обнаруживаемых антивирусом угроз:



7. Управление доверенными и контролируемыми портами:



8. Отчет о работе программы за выбранный период:



3. Содержание отчета:

1) титульный лист;
2) формулировку цели работы;
3) описание результатов выполнения;
4) выводы, согласованные с целью работы.

4. Контрольные вопросы

1. Что такое антивирус?
2. Что такое вирус?
3. Какие виды вирусов существуют?
4. Что такое проактивная защита?
5. На какие классы делятся вирусы?
6. Механизм работы современных антивирусов
7. Надежность современных антивирусов
8. Основные моменты использования современных антивирусных программ

ЛАБОРАТОРНАЯ РАБОТА № 8

Изучение функциональных возможностей системы обнаружения вторжений Snort

Цель работы: Практическое ознакомление с возможностями и настройкой системы обнаружения вторжений «Snort».

1. Общие сведения

Почти все информационные системы имеют уязвимости перед внешними и внутренними атаками. В большинстве случаев пользователи даже не подозревают, что в их компьютере или сегменте компьютерной сети содержатся уязвимости, позволяющие недоброжелателю осуществить несанкционированное проникновение в информационную систему. Реальное представление о недостатках защиты пользователи получают как правило уже после совершения взломов, проникновения вирусов или вывода из строя различных сетевых узлов. Таким образом, это одна из актуальных проблем защиты на сегодняшний день, для решения которой используется сканирование и анализ сетевого трафика.

Типичным решением этой проблемы является сканирование и анализ сетевого трафика, проходящего через сетевые узлы корпоративной системы. Специальные сетевые сканеры устанавливаются на сервер, через который идет обмен трафика информационной системы и сети Internet. В случае обнаружения вторжения, сканер выдаст предупреждение или же остановит несанкционированное действие.

Системами обнаружения вторжений (IDS - Intrusion Detection Systems) называют множество различных программных и аппаратных средств, объединяемых одним общим свойством - они занимаются анализом использования вверенных им ресурсов и, в случае обнаружения каких-либо подозрительных или просто нетипичных событий, способны предпринимать некоторые самостоятельные действия по обнаружению, идентификации и устранению их причин.

Но системы обнаружения вторжений (СОВ) лишь один из инструментов защитного арсенала и он не должен рассматриваться как замена для любого из других защитных механизмов. Защита информации наиболее эффективна, когда в интрасети поддерживается многоуровневая защита. Она складывается из следующих компонентов:

- Политика безопасности интрасети организации;

- Система защиты хостов в сети;
- Сетевой аудит;
- Защита на основе маршрутизаторов;
- Межсетевые экраны;
- Системы обнаружения вторжений;
- План реагирования на выявленные атаки.

Следовательно для полной защиты целостности сети необходима реализация всех вышеперечисленных компонентов защиты. И использование многоуровневой защиты является наиболее эффективным методом предотвращения несанкционированного использования компьютерных систем и сетевых сервисов. Таким образом, система обнаружения вторжений - это одна из компонент обеспечения безопасности сети в многоуровневой стратегии её защиты.

Классификация Систем обнаружения вторжений

Для проведения классификации IDS необходимо учесть несколько факторов (рисунок 8.1).

Метод обнаружения описывает характеристики анализатора. Когда IDS использует информацию о нормальном поведении контролируемой системы, она называется поведенческой. Когда IDS работает с информацией об атаках, она называется интеллектуальной.

Рисунок 8.1 - Характеристики систем обнаружения вторжений

Поведение после обнаружения указывает на реакцию IDS на атаки. Реакция может быть активной - IDS предпринимает корректирующие (устраняет лазейки) или действительно активные (закрывает доступ для возможных нарушителей, делая недоступными сервисы) действия. Если IDS только выдаёт предупреждения, её называют пассивной.

Расположение источников результата аудита подразделяет IDS в зависимости от вида исходной информации, которую они анализируют. Входными данными для них могут быть результаты аудита, системные регистрационные файлы или сетевые пакеты.

Частота использования отражает либо непрерывный мониторинг контролируемой системы со стороны IDS, либо соответствующие периодическим запускам IDS для проведения анализа.

Классифицировать IDS можно также по нескольким параметрам. По способам реагирования различают статические и динамические IDS. Статические средства делают «снимки» (snapshot) среды и осуществляют их анализ, разыскивая уязвимое ПО, ошибки в конфигурациях и т.д. Статические IDS проверяют версии работающих в системе приложений на наличие известных уязвимостей и слабых паролей, проверяют содержимое специальных файлов в директориях пользователей или проверяют конфигурацию открытых сетевых сервисов. Статические IDS обнаруживают следы вторжения. Динамические IDS осуществляют мониторинг в реальном времени всех действий, происходящих в системе, просматривая файлы аудита или сетевые пакеты, передаваемые за определённый промежуток времени. Динамические IDS реализуют анализ в реальном времени и позволяют постоянно следить за безопасностью системы.

По способу сбора информации различают сетевые и системные IDS. Сетевые (NIDS)контролируют пакеты в сетевом окружении и обнаруживают попытки злоумышленника проникнуть внутрь защищаемой системы или реализовать атаку «отказ в обслуживании». Эти IDS работают с сетевыми потоками данных. Типичный пример NIDS - система, которая контролирует большое число TPC-запросов на соединение (SYN) со многими портами на выбранном компьютере, обнаруживая, таким образом, что кто-то пытается осуществить сканирование TCP-портов. Сетевая IDS может запускаться либо на отдельном компьютере, который контролирует свой собственный трафик, либо на

выделенном компьютере, прозрачно просматривающим весь трафик в сети (концентратор, маршрутизатор). Сетевые IDS контролируют много компьютеров, тогда как другие IDS контролируют только один. IDS, которые устанавливаются на хосте и обнаруживают злонамеренные действия на нём называются хостовыми или системными IDS. Примерами хостовых IDS могут быть системы контроля целостности файлов (СКЦФ), которые проверяют системные файлы с целью определения, когда в них были внесены изменения. Мониторы регистрационных файлов (Log-file monitors, LFM), контролируют регистрационные файлы, создаваемые сетевыми сервисами и службами. Обманные системы, работающие с псевдосервисами, цель которых заключается
в воспроизведении хорошо известных уязвимостей для обмана злоумышленников.

По методам анализа IDS делят на две группы: IDS, которые сравнивают информацию с предустановленной базой сигнатур атак и IDS, контролирующие частоту событий или обнаружение статистических аномалий.

Анализ сигнатур был первым методом, примененным для обнаружения вторжений. Он базируется на простом понятии совпадения последовательности с образцом. Во входящем пакете просматривается байт за байтом и сравнивается с сигнатурой (подписью) - характерной строкой программы, указывающей на характеристику вредного трафика. Такая подпись может содержать ключевую фразу или команду, которая связана с нападением. Если совпадение найдено, объявляется тревога.

Второй метод анализа состоит в рассмотрении строго форматированных данных трафика сети, известных как протоколы. Каждый пакет сопровождается различными протоколами. Авторы IDS, зная это, внедрили инструменты, которые разворачивают и осматривают эти протоколы, согласно стандартам. Каждый протокол имеет несколько полей с ожидаемыми или нормальными значениями. Если что-нибудь нарушает эти стандарты, то вероятна злонамеренность. IDS просматривает каждое поле всех протоколов входящих пакетов: IP, TCP, и UDP. Если имеются нарушения протокола, например, если он содержит неожиданное значение в одном из полей, объявляется тревога.

Архитектура IDS.

У систем обнаружения вторжений целесообразно различать локальную и глобальную

архитектуру. В рамках локальной архитектуры реализуются элементарные составляющие, которые затем могут быть объединены для обслуживания корпоративных систем.

Основные элементы локальной архитектуры и связи между ними показаны на рисунке 8.2. Первичный сбор данных осуществляют агенты, называемые также сенсорами. Регистрационная информация может извлекаться из системных или прикладных журналов (технически несложно получать ее и напрямую от ядра ОС), либо добываться из сети с помощью соответствующих механизмов активного сетевого оборудования или путем перехвата пакетов посредством установленной в режим мониторинга сетевой карты.

Рисунок 8.2 - Основные элементы локальной архитектуры систем обнаружения вторжений

На уровне агентов (сенсоров) может выполняться фильтрация данных с целью уменьшения их объема. Это требует от агентов некоторого интеллекта, но зато разгружает остальные компоненты системы.

Агенты передают информацию в центр распределения, который приводит ее к единому формату, возможно, осуществляет дальнейшую фильтрацию, сохраняет в базе данных и направляет для анализа статистическому и экспертному компонентам. Один центр распределения может обслуживать несколько сенсоров.

Содержательный активный аудит начинается со статистического и экспертного компонентов. Если в процессе статистического или экспертного анализа выявляется подозрительная активность, соответствующее сообщение направляется решателю, который определяет, является ли тревога оправданной, и выбирает способ реагирования. Система

обнаружения и предотвращения вторжений Snort

Snort — свободная сетевая система предотвращения вторжений (IPS) и обнаружения вторжений (IDS) с открытым исходным кодом, способная выполнять регистрацию пакетов и в реальном времени осуществлять анализ трафика в IP сетях. Snort была написана Мартином Рёшем и в настоящее время разрабатывается компанией Sourcefire, основателем и техническим директором которой является Рёш. Sourcefire предлагает так же коммерческие версии систем для предприятий, специализированные аппаратные платформы и услуги по поддержке.

Snort выполняет протоколирование, анализ, поиск по содержимому, а также широко используется для активного блокирования или пассивного обнаружения целого ряда нападений и зондирований, таких как переполнение буфера, стелс-сканирование портов, атаки на веб-приложения, SMB- зондирование и попытки определения ОС. Программное обеспечение в основном используется для предотвращения проникновения, блокирования атак, если они имеют место.

Как следует из представленного выше описания, Snort состоит не столько из exe-файла самой программы, сколько из довольно-таки внушительного набора правил, реализующих реагирование программы в соответствии с возникшими обстоятельствами. Эти правила относятся к распознаванию различного рода атак и реагированию на них, поведению при получении пакетов от ip-адресов, отнесенных ранее к белому или черному списку контактов, правильной работе препроцессоров и многому другому. Также программа включает в себя ряд конфигурационных файлов, один из которых в том числе нам придется изменять, чтобы подстроить работу программы под наши нужды и нашу операционную систему. Кроме того, в состав программы входит ряд библиотек, помогающих выполнять функции препроцессоров и определять параметры системы и сети.

Система Snort способна работать:

- Как пакетный "снифер" (анализатор сетевого трафика, аналог tcpdump). В режиме сохранения информации обо всех переданных и полученных пакетах (удобно для диагностики сети).
- В качестве полнофункциональной сетевой системы обнаружения вторжения.
-

Порядок выполнения работы.

1. Скачать с файлового сервера \\srv02\students_RX систему обнаружения вторжений Snort. Затем установить на виртуальную машину загруженное программное обеспечение;
2. Запустить программу Snort в режиме снифера. Программа должна перехватывать весь трафик, направленный на ваш компьютер, и выводить протокол в указанный каталог;
3. Запустить программу Snort в режиме обнаружения атак с использованием стандартного файла конфигурации. Программа должна перехватывать весь трафик, направленный на ваш компьютер, и выводить протокол и предупреждения в указанный каталог;
4. Сделать анализ полученного протокола предупреждений. Команду на запуск программы и часть полученного протокола поместить в отчет по лабораторной работе. В отчете по лабораторной работе указать, какие виды нарушений безопасности или подозрительных действий обнаружены программой Snort;
5. Создать свой собственный файл конфигурации, в котором указать ссылку только на правила обнаружения сканирования вашего компьютера *scan rules.* Запустить Snort с этим файлом конфигурации. На соседнем компьютере запустить сканер *Superscan,* сканирующий ваш компьютер. Правильно настроенный Snort должен обнаружить сканирование. Полученные предупреждения о сканировании вставить в отчете;
6. Создать свой собственный файл конфигурации, в котором указать ссылку только на правила обнаружения подозрительного ICMP-трафика *icmp rules.* Проверить, предупреждает ли программа Snort об исследовании маршрута к вашему компьютеру с помощью утилиты *tracert;*

3. Содержание отчета:

1. титульный лист;
2. формулировка цели работы;
3. описание результатов выполнения;
4. выводы, согласованные с целью работы.

4. Контрольные вопросы

1. Что такое системы обнаружения вторжений?
2. Для чего применяют СОВ?
3. Классификация систем обнаружения вторжений.
4. Архитектура СОВ.
5. Система обнаружения вторжений «Snort».