

**И.В. Аникин**

**ПРОГРАММНО-АППАРАТНЫЕ СРЕДСТВА ЗАЩИТЫ  
ИНФОРМАЦИИ**

**И.В. Аникин**

**ПРОГРАММНО-АППАРАТНЫЕ СРЕДСТВА ЗАЩИТЫ  
ИНФОРМАЦИИ**

*Учебно-методическое пособие*

*Рекомендовано к изданию  
Учебно-методическим управлением КНИТУ-КАИ*

Казань  
2018

УДК 004.056.5

ББК 32.973.26-018.2

**A 95**

**Рецензенты:**

**Кафедра «Информационная безопасность»** (Казанский национальный исследовательский технологический университет)

**Ш.Т. Ишмухаметов** – доктор физ.-мат. наук, профессор кафедры Системного анализа и информационных технологий Казанского (Приволжского) федерального университета.

**Аникин И.В.**

**A 95**

Программно-аппаратные средства защиты информации: учебно-методическое пособие – Казань: Редакционно-издательский центр «Школа», 2018. – 147 с., ил.

**ISBN 978-5-906935-93-9**

Предлагаемое учебно-методическое пособие предназначено для организации учебного процесса при изучении дисциплины «Программно-аппаратные средства защиты информации» студентами направления 10.03.01 «Информационная безопасность».

УДК 004.056.5

ББК 32.973.26-018.2

**ISBN 978-5-906935-93-9**

© Аникин И.В., 2018

## **СПИСОК СОКРАЩЕНИЙ**

<b>АС</b>	–	Автоматизированная система
<b>ИБ</b>	–	Информационная безопасность;
<b>ИТ</b>	–	Информационные технологии;
<b>НСД</b>	–	Несанкционированный доступ
<b>ОЗУ</b>	–	Оперативное запоминающее устройство
<b>ОС</b>	–	Операционная система
<b>ПЗУ</b>	–	Постоянное запоминающее устройство
<b>ПО</b>	–	Программное обеспечение
<b>ПРД</b>	–	Правила разграничения доступа
<b>СЗПО</b>	–	Система защиты программного обеспечения
<b>СВТ</b>	–	Средство вычислительной техники
<b>ЭВМ</b>	–	Электронно-вычислительная машина

## **ВВЕДЕНИЕ**

Предлагаемое учебно-методическое пособие предназначено для организации учебного процесса при изучении дисциплины «Программно-аппаратные средства защиты информации» студентами направления 10.03.01 «Информационная безопасность».

В разделах 1-3 даются теоретические сведения по разделам курса «Защита программного обеспечения от несанкционированного копирования», «Защита программного обеспечения от несанкционированного исследования», «Защита автоматизированных систем от несанкционированного доступа».

Во четвертом разделе приводятся 8 лабораторных работ по выше представленным разделам дисциплины.

## **1. ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ НЕСАНКЦИОНИРОВАННОГО КОПИРОВАНИЯ**

В общем случае, система защиты программного обеспечения (ПО) от несанкционированного использования представляет собой комплекс средств, предназначенный для затруднения (а в идеале – предотвращения) нелегального копирования (исполнения) защищаемого программного модуля, с которым она ассоциирована.

Особенностью технических мер защиты ПО от несанкционированного использования является выделение (или введение) некоторого идентифицирующего элемента из среды окружения защищаемой программы, имеющего уникальные физические характеристики, на которые настраивается система защиты. В качестве идентифицирующего элемента, позволяющего отличать одну копию программного продукта от другой, могут выступать:

- серийный номер S/N;
- ключ;
- конфигурация аппаратуры;
- элементы электронного ключа, другие аппаратные устройства и т.д.

Основными требованиями к системе защиты ПО от несанкционированного использования являются следующие:

- 1) система защиты должна выявлять факт несанкционированного запуска программы;
- 2) система защиты должна реагировать на факт несанкционированного запуска программы;
- 3) система защиты должна противостоять возможным атакам злоумышленников, направленным на нейтрализацию системы защиты.

На рисунке 1.1 представлена иерархия модулей системы защиты программного обеспечения от несанкционированного использования, а также направления взаимодействия данных модулей. Система защиты состоит из двух основных частей: подсистемы внедрения механизмов системы защиты и внедряемого защитного кода. Последний блок, в свою очередь, состоит из подсистемы реализации защитных функций и подсистемы противодействия нейтрализации защитных механизмов.



Рисунок 1.1. Модульная архитектура системы защиты программного обеспечения от несанкционированного использования

Любая система защиты ПО от несанкционированного использования функционирует по следующему обобщённому алгоритму, отражающему взаимодействие перечисленных подсистем и модулей.

1) Разработчик программы внедряет защитные механизмы в защищаемую программу.

2) В защитные механизмы закладываются эталонные характеристики среды, которые идентифицируют конкретную копию программы, и относительно которых в дальнейшем будет проверяться легальность запуска.

3) При каждом запуске программы выполняются следующие действия:

- снимаются текущие характеристики среды;
- снятые характеристики среды сравниваются с эталонными;
  - если сравнение характеристик дало положительный результат, то программа запускается (либо продолжает работать);
  - если сравнение характеристик дало отрицательный результат, то запускается блок ответной реакции.

#### Подсистема внедрения механизмов защиты

Системы защиты программного обеспечения от несанкционированного использования по способу ассоциации (внедрения) защитного механизма можно подразделить на два типа:

- 1) встроенные системы (внедряются при создании ПО);
- 2) пристыковочные системы (подключаются к уже готовому ПО).

Во *встроенных защитах* подсистема внедрения защитных механизмов в явном виде отсутствует. Встраивание защиты производится непосредственно разработчиком в процессе создания ПО. При этом разработчик может применять либо собственные разработки элементов защиты, либо готовый набор модулей. В качестве примеров подобных защит можно привести использование разработчиком API функций электронных ключей HASP для защиты своего программного продукта.

*Пристыковочные защиты* внедряются в уже готовый исполняемый код, часто по вирусному принципу преобразуя код программы. При запуске защищённой программы производятся необходимые проверки после чего код защиты восстанавливает оригинальное начало файла и передаёт на него управление. В качестве примера подобной защиты можно привести использование модуля пакетной обработки защищаемых файлов HASP Envelope для электронных



ключей HASP, позволяющего в автоматическом режиме защитить исполняемый код.

#### Подсистема противодействия нейтрализации защитных механизмов

Противодействие нейтрализации защитных механизмов в первую очередь связано с противодействием анализу злоумышленником логики и принципов работы защитных механизмов, а также с противодействием возможному вмешательству в их функционирование. Нейтрализация защитных механизмов может вестись по двум основным направлениям – статическому и динамическому. При статической нейтрализации защищаемая программа сначала дизассемблируется и по полученному коду изучается логика её работы. При динамической нейтрализации изучение, реконструирование логики работы защитных механизмов, а также вмешательство в их работу часто осуществляется с помощью отладчиков.

#### Блок ответной реакции

Блок ответной реакции является одновременно самым простым и самым сложным при проектировании систем защиты ПО от несанкционированного использования. Именно ответная реакция ПО на некорректные характеристики среды часто позволяет локализовать злоумышленнику защитный код. С точки зрения безопасности предпочтительны ситуации, когда данный блок присутствует в коде защищаемой программы неявным образом. Например, информация, полученная от блока установки характеристик среды, может быть использована в качестве ключа для дешифрования кода программы. Если значения характеристик среды отличаются от эталонных, то при дешифровании получится «мусор», передача управления на который приведет к сбою в работе.

#### Блок сравнения характеристик среды

Данный блок часто подвергается атакам со стороны злоумышленников. Атаки направлены на модификацию блока таким образом, чтобы отключить вызов блока ответной реакции, либо запускать его только по одной из ветвей.

Для того чтобы систему защиты ПО от несанкционированного использования нельзя было нейтрализовать с помощью простейших приёмов, блок сравнения значений характеристик среды должен отвечать ряду требований.

1) Установка значений характеристик среды и их сравнение с эталонными значениями должны производиться многократно.

2) Сравнение текущих значений характеристик среды с эталонными должно производиться периодически в течение всего сеанса работы программы. Это делается для того, чтобы однократно проверенный в начале работы программы идентифицирующий элемент не мог быть использован для запуска других копий.

3) Получение результатов сравнения должно быть принудительно распределено в исполняемом коде программы.

Удачным приёмом против потенциального злоумышленника считается преобразование значения на выходе блока установки характеристик среды в некоторую переменную, которая затем может быть использована, например, как аргумент для обращения к управляющей таблице при вычислении значения адреса перехода или вызова подпрограммы. Также значения, полученные от блока установки характеристик среды, можно использовать в качестве ключа для дешифрования исполняемого кода, для которого (перед передачей ему управления) подсчитывается контрольная сумма.

#### Блок установки характеристик среды

Блок установки характеристик среды определяет наличие идентифицирующего элемента, его текущие параметры, и передаёт их значения блоку сравнения характеристик среды. Цель атаки на данный блок - выяснение идентифицирующей информации, которую «ждёт» блок сравнения характеристик среды.

Наиболее уязвимым местом в блоке установки характеристик среды является не его код, а способ передачи и объём передаваемых данных. Например, если все характеристики идентифицирующего элемента преобразовываются для дальнейшего сравнения в однобайтовую переменную, появляется возможность нейтрализации системы защиты путём перебора возможных её значений.

Для защиты от подобных атак эталонные характеристики среды не должны в явном виде присутствовать в коде программы. Данные характеристики должны закрываться, например, с использованием криптографически стойких функций хэширования.

К наиболее часто используемым средствам защиты ПО от несанкционированного копирования следует отнести электронные ключи.

## **2. ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ НЕСАНКЦИОНИРОВАННОГО ИССЛЕДОВАНИЯ**

Основными угрозами для программного продукта, защищённого от несанкционированного использования, являются угроза нарушения функциональности модуля защиты и угроза раскрытия эталонных характеристик среды. Реализация первой угрозы может заключаться в обходе либо полном отключении модуля защиты путём модификации кода программы. Реализация второй угрозы – в выяснении эталонных характеристик среды путём исследования программно-аппаратной среды функционирования защищённой программы.

В любом случае, присутствует ряд задач, которые злоумышленник должен решить при реализации данных угроз.

1. *Задача локализации кода защитного механизма в коде программы.* Следует отметить, что для современного ПО без использования специализированных программных средств эта задача злоумышленником не может быть решена за приемлемое время, что обусловлено следующими причинами.

- Код защитного механизма занимает сравнительно малый объем в общем объеме кода программы. Задача ручного поиска блоков установки, сравнения характеристик среды, и ответной реакции, занимающих размер 100-200 байт в коде программы, занимающем сотни мегабайт, без использования специализированных средств не может быть решена за приемлемое для злоумышленника время.

- Анализ кода программы злоумышленником в значительной степени затрудняется тем, что производится анализ не исходного текста на языке высокого уровня, а машинного кода, сформированного компилятором. На его разбор и понимание уходит значительное время даже у специалистов высокого класса.

Также, при анализе машинного кода исследователю приходится увязывать в единую логическую последовательность действий как минимум 10-20 команд, чтобы понять скрываемый за ними результат. Это усложняет анализ программного кода.

*2. Задача исследования модуля защиты и понимания принципов его действия.* Злоумышленник должен понять, каким образом построена защита, где она хранит эталонные характеристики среды, где сохраняет свои метки и ключи, на каком этапе принимается решение о регистрации программы либо об отклонении регистрации. При этом злоумышленник сталкивается с проблемой анализа машинного кода, что приводит к трудностям, описанным выше.

Одна из основных задач, решаемая злоумышленником при исследовании ПО – анализ логики работы программы, поиск в ней участков кода, отвечающих за реализацию защитных механизмов, детальное исследование принципов работы модулей защиты. При этом ставится задача представления машинного кода на как можно более высоком уровне абстракции с целью упрощения его понимания.

*Под обратным проектированием (reverse engineering)* понимают процесс исследования и анализа машинного кода, нацеленный на понимание общих механизмов функционирования программы, а также на его перевод на более высокий уровень абстракции вплоть до восстановления текста программы на исходном языке программирования.

Основными методами обратного проектирования являются отладка и дизассемблирование программ. При этом используются следующие основные средства (инструменты).

*Отладчики* – программные средства, позволяющие выполнять программу в пошаговом режиме, контролировать ее выполнение, вносить изменения в ход выполнения. Данные средства позволяют проследить весь механизм работы программы и являются средствами динамического исследования работы программ.

*Дизассемблеры* – программные средства, позволяющие получить листинг программы на языке ассемблера с целью его дальнейшего статического изучения. Дизассемблеры являются средствами статического исследования. По способу взаимодействия с пользователем существующие дизассемблеры можно разделить на две категории – автономные и интерактивные.

Автономные дизассемблеры требуют от пользователя задания всех указаний до начала дизассемблирования и не позволяют вмешиваться непосредственно в сам процесс. Если конечный результат окажется неудовлетворительным, то пользователь либо вручную правит полученный листинг, либо указывает дизассемблеру на его ошибки и повторяет всю вновь.

Интерактивные дизассемблеры обладают развитым пользовательским интерфейсом, благодаря которому приобретают значительную гибкость, позволяя человеку «вручную» управлять процессом анализа программы, помогая автоматическому анализатору там, где ему самому не справиться отличать адреса от констант, определять границы инструкций и т.д.

Примером автономного дизассемблера является SOURCER, а интерактивного IDA Pro.

*Мониторы событий* – программные средства, позволяющие отслеживать определенные типы событий, происходящие в системе.

*Редакторы кода* занимают отдельное место среди средств обратного проектирования. Данные средства, как правило, включают функции дизассемблирования, но позволяют также вносить изменения в код программы. Данные средства предназначены, в основном, для исследования программ, занимающих небольшой объём.

Наиболее общую классификацию средств обратного проектирования программного обеспечения можно представить в виде следующей схемы (рисунок 2.1).

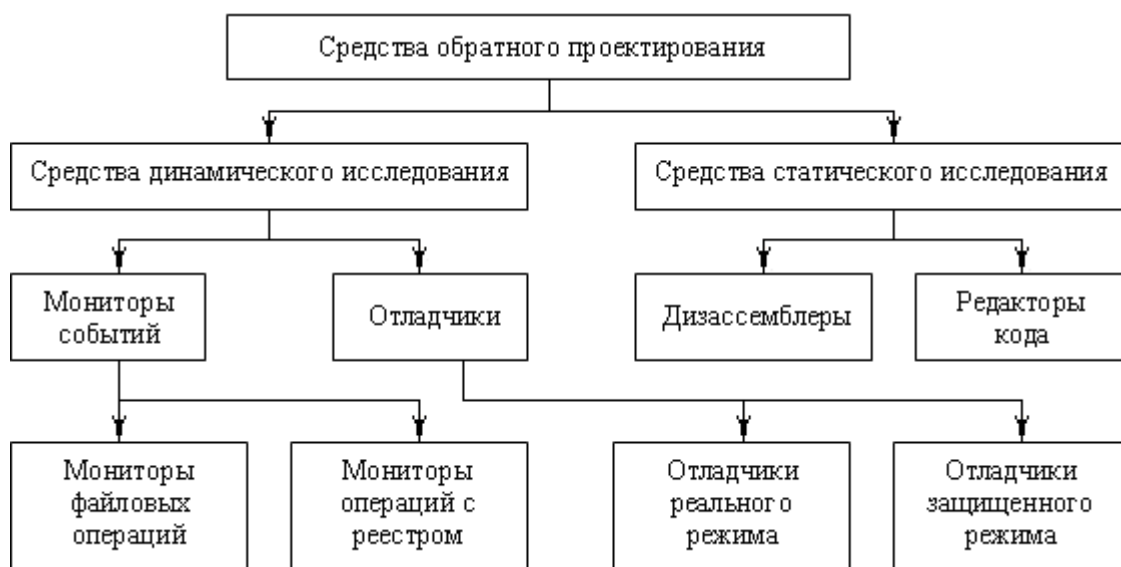


Рисунок 2.1. Классификация средств обратного проектирования

Таким образом, использование техники обратного проектирования позволяет злоумышленнику детально разобраться с принципами функционирования программы, в том числе и модулей защиты. Это поможет ему в дальнейшем атаковать отдельные модули системы защиты. Производитель ПО должен стремиться не допустить этого и противодействовать внутреннему исследованию кода своей программы злоумышленником. Таким образом, защита ПО от внутреннего исследования (в особенности критических модулей, ответственных за безопасность), является одной из актуальных задач для производителя ПО. Производитель должен уметь противостоять всем средствам обратного проектирования, облегчающим злоумышленнику задачу взлома.

#### Средства анализа и преодоления систем защиты программного обеспечения

В настоящее время существуют лишь неформальные классификации средств анализа и преодоления систем защиты программного обеспечения (СЗПО) (рисунок 2.2).



Рисунок 2.2 Средства анализа и преодоления систем защиты программного обеспечения



Все средства исследования программного обеспечения можно разбить на два класса: статические и динамические. Первые оперируют исходным кодом программы как данными и строят её алгоритм без исполнения. Вторые изучают программу, интерпретируя её в реальной или виртуальной вычислительной среде. Отсюда следует, что первые средства являются более универсальными в том смысле, что теоретически могут получить алгоритм работы всей программы, в том числе и тех блоков, которые никогда не получают управления. Динамические средства могут строить алгоритм работы программы только на основании конкретной её трассы, полученной при определённых входных данных. При динамическом исследовании можно говорить только о построении некоторой части алгоритма.

#### Отладчики и дизассемблеры

В состав стандартных функций отладчиков входят возможности пошагового выполнения объектного кода, установки точек останова (в том числе срабатывающих по условию), просмотра объектного кода программы в дизассемблированном виде, изменения последовательности выполнения объектного кода, редактирования памяти отлаживаемого процесса, отслеживания изменения данных процесса и др.

В рамках исследования СЗПО использование отладчиков реализует динамический анализ алгоритмов СЗПО. Преодоление практически любой СЗПО в большинстве случаев невозможно без использования отладочных средств. При этом большая часть отладочных функций реализуется в архитектуре центрального процессора ЭВМ.

Дизассемблеры предназначены для «детрансляции» объектных модулей из машинного кода в мнемокод ассемблера. При применении средств дизассемблирования производится статический анализ алгоритмов СЗПО по мнемокоду. Получение доступа к мнемокоду СЗПО даёт превосходную возможность детального анализа программного и алгоритмического исполнения процедур СЗПО, а также нахождения конкретных путей обхода или модификации ключе-

вых фрагментов СЗПО. Иногда появляется возможность использования элементов СЗПО во вновь создаваемых средствах их преодоления.

#### Программы-каталогизаторы (файловые оболочки операционных систем)

В стандартные возможности таких программ входят функции просмотра атрибутов файлов (тип, дата создания/модификации, размер, флаги доступа и др.), подсчета их количества и общего объема в каталоге приложения, просмотра файлов и т.п. При помощи этого типа программных средств, как правило, реализуется предварительный анализ защищенных продуктов и первичная локализация СЗПО.

Примером подобного использования может быть сравнение дат создания всех файлов в каталоге установленного приложения. В случае использования системой защиты каких-либо динамических библиотек, разница в датах их создания позволит легко локализовать файлы, относящиеся к СЗПО (как правило, даты создания "рабочих" файлов пакета совпадают, дата создания модулей СЗПО отличается от них). Аналогичным образом локализуются и файлы, хранящие счетчики количества запусков ПО. При помощи обычного текстового просмотра объектного модуля можно довольно легко определить тип и производителя СЗПО, так как обычно эта информация включается в тело защищенного модуля.

#### Программы поиска файлов, текстовых и двоичных последовательностей в текстовых и двоичных файлах

Данный тип программ позволяет производить поиск заданной последовательности (включая поиск по маске) в одном или сразу нескольких файлах с выдачей результатов в виде списка смещений относительно начала файла, по которым был найден искомый фрагмент. При помощи указанных средств реализуется вторичный анализ СЗПО и локализация ее ключевых фрагментов.

Обычно средства файлового поиска используются для следующих целей: поиска известных сигнатур СЗПО в объектных модулях, поиска строк с сообщениями СЗПО (например, «Программа не зарегистрирована!» или «Спасибо за регистрацию!»), поиска файлов СЗПО с известными именами/сигнатурами.

Первый и последний виды использования рассматриваемого типа ПО ориентированы на отыскание стандартных элементов СЗПО, исследованных ранее. Второй вид использования поисковых программ ориентирован на локализацию процедур СЗПО, отвечающих за идентификацию и аутентификацию легального пользователя ПО.

#### Программы-мониторы файловой системы (File Monitors)

Этот тип программ позволяет отслеживать изменения, происходящие в файловой системе при запуске приложений. В большинстве таких программ предусмотрена система фильтров для формирования протоколов работы отдельных приложений. При помощи данного типа средств реализуется анализ работы СЗПО с файлами.

Например, подобные программы позволяют выяснить, что именно и где именно изменяют модули СЗПО. Эта информация позволяет точно локализовать счетчики количества запусков ПО, скрытые файлы систем «привязки» ПО, «ключевые файлы», файлы с информацией о функциях ПО, разрешенных для использования в рамках конкретной лицензии на продукт и т.п. Примером подобной программы является Filemon.

#### Программы-мониторы реестра (Registry Monitors)

Программные средства этого типа предназначены для отслеживания изменений, вносимых приложениями в ключи реестра ОС Windows. Рассматриваемые средства позволяют определять, работает ли СЗПО с реестром ОС, какие изменения она туда вносит и какие данные использует. В результате подобного анализа становится возможным обнаружить скрытые счетчики количества запусков ПО, записи с лицензионными ограничениями функциональности ПО и т.п. Примером подобной программы является Regmon.

#### Программы-мониторы вызовов подпрограмм операционной системы (API Monitors)

ПО этого типа предназначено для отслеживания вызовов системных функций одним или несколькими приложениями с возможностью фильтрации/выделения групп отслеживаемых системных функций или приложений.

Применение таких программ позволяет проводить анализ использования СЗПО системных функций. Учитывая, что все действия ПО (и СЗПО), связанные с работой с файловой системой, работой с конфигурацией ОС, реализацией диалога с пользователем, работой с сетью и многим другим, реализуются посредством вызова функций ОС, анализ использования СЗПО системных функций позволяет подробно изучить механизмы работы систем защиты, найти их слабые места и разработать пути их обхода. Примером подобных программ являются API Monitor, APISpy.

#### Программы-мониторы обмена данными с системными устройствами (портами) (Port Monitors)

В современной архитектуре ОС доступ ко всем системным устройствам (их контроллерам) осуществляется через порты ввода/вывода. Все современные ОС виртуализируют эти порты, организуя таким образом совместный доступ нескольких приложений к одному и тому же порту, а также осуществляя контроль доступа к портам в целях обеспечения безопасности ОС. Использование программных средств этого типа позволяет проводить анализ взаимодействия СЗПО с системными устройствами. Контролируя доступ и обмен данными через порты ввода/вывода программной и аппаратной частей СЗПО, можно анализировать и преодолевать различные механизмы защит: привязку к электронным ключам, конфигурации аппаратуры и т.д. Примером подобной программы является PortMon.

#### Программы-мониторы сетевого обмена данными (Network Traffic Monitors)

Рассматриваемый тип программ предназначен для отслеживания сетевой активности приложений в рамках ОС. Использование сетевых мониторов позволяет проводить анализ сетевого обмена СЗПО.

Ряд современных программных продуктов реализует проверку аутентичности пользователя путем запроса данных о состоянии лицензии для данной рабочей станции с "сервера лицензий" в ЛВС. Также в последнее время появились программные продукты, проверяющие аутентичность пользователя или

срок своего использования через Интернет. Кроме указанных видов ПО, существуют также условно бесплатные программные продукты, в которых временные или функциональные ограничения заменены обязательным просмотром рекламной информации, получаемой через Интернет. Отслеживая сетевой обмен подобных программ, можно анализировать механизмы систем их защиты. К данному типу ПО можно отнести различные снифферы.

#### Программы-мониторы активных задач, процессов, потоков и окон (Process/Windows Managers)

Указанный тип программных средств предназначен для отслеживания и управления рядом объектов ОС: задачами, процессами, потоками, окнами и др.. Подобные программы обычно предоставляют возможности поиска необходимого объекта ОС, переключения на него управления, изменения его приоритета, уничтожения объекта, сохранения его параметров (а иногда и содержимого) на диске. Их применение дает возможность производить анализ модульной структуры СЗПО. Подобный анализ позволяет выяснить подробности организации СЗПО во время ее работы. Список динамически загружаемых процессом библиотек, данные о количестве создаваемых и уничтожаемых приложением потоков и окон, поведение программ при попытке принудительно завершить процесс, содержащий СЗПО, позволяют существенно дополнить картину анализа функционирования системы защиты. Примером подобных программ являются Handle, IceSword.

#### Программы-мониторы конвейеров данных, системных сообщений и высокоуровневого межпрограммного взаимодействия (Message/COM hooks)

Средства данного типа предназначены для отслеживания сообщений, данных и вызовов подпрограмм, которыми обмениваются объекты ОС. Применение мониторов сообщений позволяет производить анализ межмодульного взаимодействия в рамках СЗПО. В результате такого анализа можно получить доступ к информации о протоколах обмена данными между различными частями системы защиты, условиях ее срабатывания и отключения, динамике функционирования защиты.

### Программы копирования областей ОЗУ в ПЗУ (Memory Dumpers)

Указанный тип программных средств предназначен для сохранения областей оперативной памяти на жесткий диск. В рамках исследования СЗПО данные средства позволяют произвести принудительное сохранение образа памяти защищенного приложения. В случае использования механизмов шифрования/упаковки объектного кода защищаемого ПО, сохранение памяти активного процесса ОС позволяет получить копию кода защищаемого ПО в «открытом виде». В результате таких действий возможно либо сразу получить экземпляр незащищенного программного продукта, либо получить важную для дальнейшего анализа СЗПО информацию.

### Программы побайтового копирования гибких магнитных дисков, оптических дисков и жёстких магнитных дисков

Данный тип программных средств предназначен для создания максимально точных копий физической структуры носителей данных без учета их логической структуры. Чрезвычайной популярностью пользуются средства побайтового копирования оптических дисков и средства сохранения содержимого оптических дисков в виде файлов на жестких дисках.

### Программы-распаковщики/дешифраторы (Unpackers/Decryptors)

Средства распаковки/дешифрации объектных модулей позволяют получать копии указанных модулей в виде, в котором они были до их упаковки/шифрования. Как правило, распаковка/дешифрация защищенных объектных модулей ПО производится для получения возможности более глубокого дальнейшего анализа СЗПО. Для определения типа используемой защиты можно использовать программу Protection ID. Для снятия специализированных защит – программы ACStripper, Armadilo Find Protect, ASPack Unpacker.

### Средства декомпиляции объектных модулей программного обеспечения (Decompilers)

Декомпилирующие программы сходны дизассемблерам и даже иногда их используют в качестве своих подпрограмм. Задачей, стоящей перед данным типом программных средств, является "детрансляция" объектных модулей из ма-

шинного кода в исходный код на языке высокого уровня. Применение декомпиляторов к исследуемому ПО реализует статический анализ алгоритмов СЗПО по исходному коду. Большинство современных декомпиляторов ориентировано на обработку объектных модулей, написанных на языках интерпретирующего типа (FoxPro, Clipper, Visual Basic, Java), декомпиляторы для языков компилирующего типа встречаются крайне редко и обладают ограниченными возможностями в силу особенностей процесса компиляции. Декомпиляция ПО даёт доступ к его исходному коду (или его эквиваленту) и позволяет полностью распоряжаться программным продуктом, включая внесение в него функциональных изменений и повторную компиляцию. Примерами программных программ являются HEXRays, DJ Java Decompiler, Liatro SWF Decoder, DeDe.

#### Средства редактирования «ресурсов» объектных модулей

Подобные программы используются для редактирования текстовых, диалоговых, графических, аудио-, видео- и других ресурсов, содержащихся в области данных объектных модулей ПО. В рамках исследования СЗПО подобные средства позволяют производить редактирование их ресурсов. Модификация этих ресурсов позволяет изменить интерфейс программы, в том числе активировать отключенные в рамках данной лицензии на ПО пункты меню, предотвратить выдачу приложением предупреждающих надписей о необходимости приобретения ПО, изменить диалоговые окна и т.п. Примерами подобных программ являются XN Resource Editor, Restorator Resource Editor.

#### Средства загрузки объектных модулей и/или их динамической модификации в ОЗУ

Этот тип программных средств предназначен для модификации памяти процесса ОС во время его выполнения в ОЗУ. За исключением особенностей модификации объектного кода в оперативной памяти, данные средства функционально подобны средствам поиска и замены в файлах. При помощи таких программ осуществляется динамическая модификация кода СЗПО. Обычно средства динамической модификации кода используются при высокой сложности или нерациональности распаковки/дешифрации объектных модулей защи-

щённого программного обеспечения. Примером подобной программы является Process patcher.

Для защиты от использования представленных выше СЗПО необходимо следовать рекомендациям по безопасной реализации блоков установки и сравнения характеристик среды, блока ответной реакции, а также защищать модули защиты от внутреннего исследования.



### **3. ЗАЩИТА АВТОМАТИЗИРОВАННЫХ СИСТЕМ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА**

В 1992 году ФСТЭК России (в то время Гостехкомиссией) были разработаны и опубликованы пять руководящих документов, посвященных вопросам защиты информации от НСД в системах ее обработки.

1. Защита от несанкционированного доступа к информации. Термины и определения.

2. Концепция защиты СВТ и АС от НСД к информации.

3. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации.

4. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от НСД к информации.

5. Временное положение по организации разработки, изготовления и эксплуатации программных и технических средств защиты информации от НСД в автоматизированных системах и средствах вычислительной техники.

Третий документ устанавливает девять классов защищенности АС от НСД, распределенных по трем группам. Каждый класс характеризуется определенной совокупностью требований к средствам защиты. В пределах каждой группы соблюдается иерархия классов защищенности АС.

*Третья группа* включает АС, в которых работает один пользователь, допущенный ко всей информации АС, размещенной на носителях одного уровня конфиденциальности. Данная группа содержит два класса защищенности - ЗБ (низший) и ЗА (высший).

*Вторая группа* включает АС, в которых пользователи имеют одинаковые полномочия доступа ко всей информации, обрабатываемой и хранимой в АС на носителях различного уровня конфиденциальности. Данная группа содержит два класса защищенности - 2Б (низший) и 2А (высший).

*Первая группа* включает многопользовательские АС, в которых одновременно обрабатывается и хранится информация разного уровня конфиденциальности, не все пользователи имеют равные права доступа. Данная группа содержит пять классов защищенности - 1Д (низший), 1Г, 1В, 1Б и 1А (высший).

В таблице 3.1 приведены требования к подсистемам защиты для каждого класса защищенности.

Таблица 3.1. Показатели защищенности и требования к классам защиты АС от НСД

Подсистемы и требования	Классы								
	3Б	3А	2Б	2А	1Д	1Г	1В	1Б	1А
<b>I. Подсистема управления доступом</b>									
<b>А. Идентификация, проверка подлинности и контроль доступа субъектов:</b>									
• в систему	+	+	+	+	+	+	+	+	+
• к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ				+		+	+	+	+
• к программам				+		+	+	+	+
• к томам, каталогам, файлам, записям, полям записей				+		+	+	+	+
<b>В. Управление потоками информации</b>				+			+	+	+
<b>II. Подсистема регистрации и учета</b>									
<b>А. Регистрация и учет</b>									
• входа/выхода субъектов доступа в/из системы (узла сети)	+	+	+	+	+	+	+	+	+
• выдачи печатных (графических) выходных документов		+		+		+	+	+	+
• запуска/завершения программ и процессов (заданий, задач)				+		+	+	+	+
• доступа программ субъектов доступа к защищаемым файлам, включая их создание и удаление, передачу по линиям и каналам				+		+	+	+	+

Подсистемы и требования	Классы								
	ЗБ	ЗА	ЗБ	2А	1Д	1Г	1В	1Б	1А
связи									
• доступа программ субъектов доступа к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ, программам, томам, каталогам, файлам, записям, полям записей				+		+	+	+	+
• изменения полномочий субъектов доступа							+	+	+
• создаваемых защищаемых объектов доступа				+			+	+	+
<b>В. Учет носителей информации</b>	+	+	+	+	+	+	+	+	+
<b>С. Очистка (обнуление, обезличивание) освобождаемых областей оперативной памяти ЭВМ и внешних накопителей</b>		+		+		+	+	+	+
<b>Д. Сигнализация попыток нарушения защиты</b>							+	+	+
<b>III. Криптографическая подсистема</b>									
<b>А. Шифрование конфиденциальной информации</b>				+				+	+
<b>В. Шифрование информации, принадлежащей различным субъектам доступа (группам субъектов) на разных ключах</b>									+
<b>С. Использование аттестованных (сертифицированных) криптографических средств</b>				+				+	+
<b>IV. Подсистема обеспечения целостности</b>									
<b>А. Обеспечение целостности программных средств и обрабатываемой информации</b>	+	+	+	+	+	+	+	+	+
<b>В. Физическая охрана средств вычислительной техники и носителей информации</b>	+	+	+	+	+	+	+	+	+
<b>С. Наличие администратора (службы) защиты информации в АС</b>				+			+	+	+
<b>Д. Периодическое тестирование СЗИ НСД</b>	+	+	+	+	+	+	+	+	+
<b>Е. Наличие средств восстановления СЗИ НСД</b>	+	+	+	+	+	+	+	+	+
<b>Ф. Использование сертифицированных средств защиты</b>		+		+			+	+	+

Мероприятия по защите информации от НСД являются составной частью управленческой, научной, производственной (коммерческой) деятельности предприятия (учреждения, фирмы и т.д.), независимо от их ведомственной принадлежности и формы собственности, и осуществляются в комплексе с другими мерами по обеспечению установленного режима конфиденциальности. Практика организации защиты информации от НСД при ее обработке и хранении в АС должна учитывать соответствие уровня безопасности информации законодательным положениям и нормативным требованиям по охране сведений, подлежащих защите по действующему законодательству, в т.ч. выбор класса защищенности АС в соответствии с особенностями обработки информации (технология обработки, конкретные условия эксплуатации АС) и уровнем ее конфиденциальности. Средства защиты информации должны иметь сертификат, удостоверяющий их соответствие требованиям по безопасности информации.

Примерами средств защиты информации от НСД являются программно-аппаратные комплексы Secret Net, Аккорд, Dallas Lock.

#### Secret Net



Система Secret Net предназначена для защиты от НСД информации, составляющей коммерческую или государственную тайну в автоматизированных системах вплоть до класса 1В. Возможности Secret Net позволяют реализовать следующие функции:

- аутентификацию пользователей с использованием технических средств;
- обеспечение разграничения доступа к защищаемой информации и устройствам;
- создание изолированной программной среды;
- контроль каналов распространения конфиденциальной информации за счет реализации политики Белла-ЛаПадулы;
- контроль устройств компьютера и отчуждаемых носителей информации на основе централизованных политик, исключающих утечки конфиденциальной информации;

- централизованное управление политиками безопасности;
- мониторинг и аудит безопасности.

Возможно использование локальной или сетевой версии Secret Net. Аппаратные средства, поддерживаемые Secret Net, представлены в таблице 3.2.

Таблица 3.2. Виды аппаратной поддержки Secret Net

	<p>Электронный замок "Соболь" – плата с разъемом для подключения считывателя Touch Memory, аппаратным датчиком случайных чисел, 2-мя (4-мя) каналами физической блокировки устройств и внутренней энергонезависимой памятью. Используется в СЗИ НСД Secret Net для идентификации пользователей по электронным идентификаторам Touch Memory, а также для генерации криптографических ключей.</p>
	<p>Secret Net Touch Memory Card - плата с разъемом для подключения считывателя Touch Memory или считывателя бесконтактных радиокарт Proximity. Обеспечивает идентификацию пользователей по электронным идентификаторам Touch Memory или картам Proximity до загрузки операционной системы, хранение во внутренней памяти идентификатора (Touch Memory) дополнительной информации.</p>

В Secret Net возможно задать временные ограничения на вход в систему - интервал времени по дням недели (с дискретностью 30 мин), в который разрешена загрузка ЭВМ данным пользователем (субъектом доступа) (рисунок 3.1).

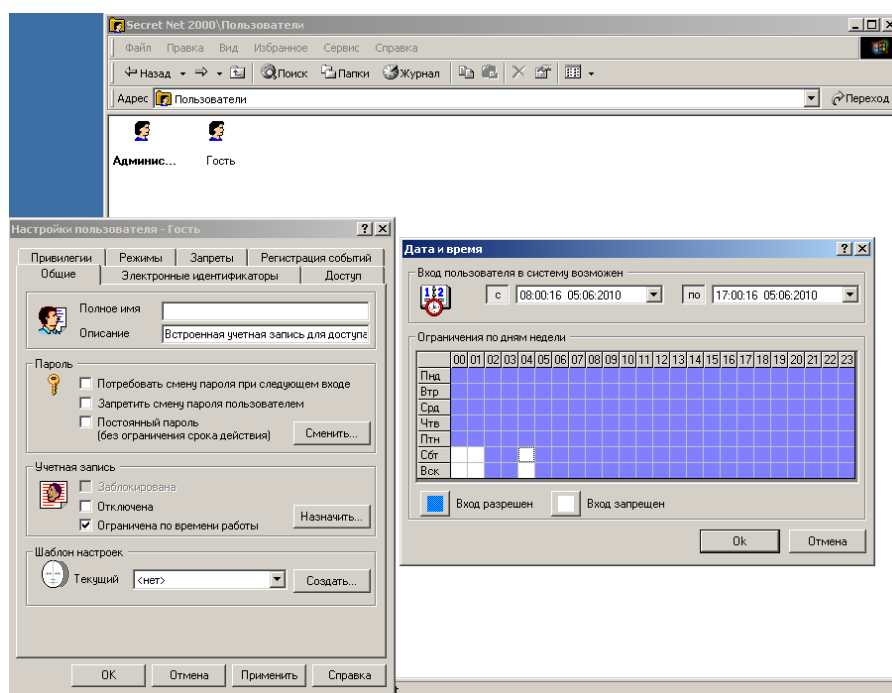


Рисунок 3.1. Задание временных ограничений на вход в систему

Разграничение доступа к дискам и портам так же может производиться применительно к группе, либо к конкретному пользователю (рисунок 3.2).

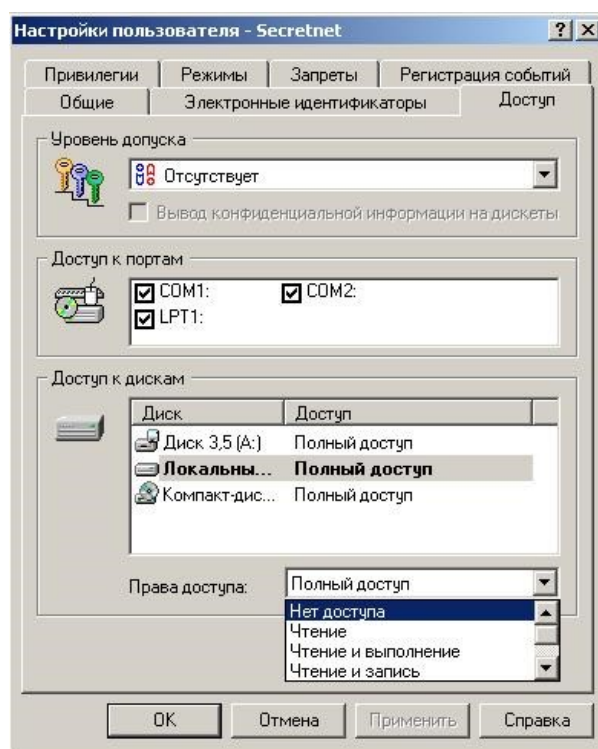


Рисунок 3.2 Разграничение доступа к дискам и к портам

Возможно задание количества циклов затирания данных и контроль целостности программных средств и обрабатываемой информации (рисунок 3.3).

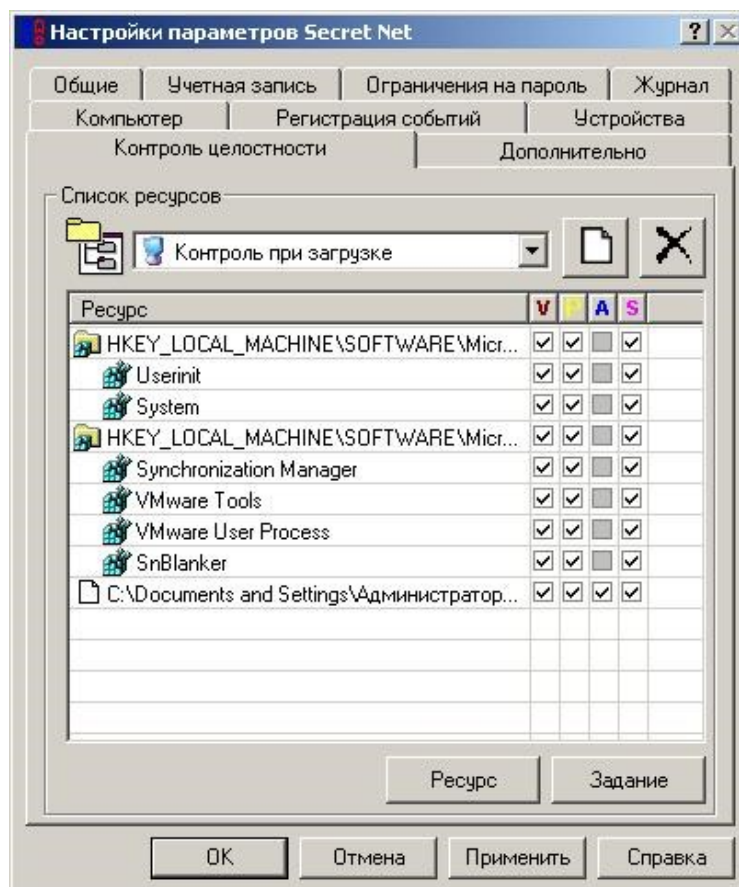


Рисунок 3.3 Контроль целостности программных средств и обрабатываемой информации

## АККОРД

Программно-аппаратный комплекс СЗИ НСД «Аккорд» предназначен для применения на ЭВМ, функционирующих под управлением ОС Windows, с целью обеспечения защиты от НСД к ПЭВМ и АС на их основе, при многопользовательском режиме эксплуатации. Комплекс СЗИ НСД «Аккорд» поставляется в программно-аппаратном исполнении и включает:

- программно-аппаратный комплекс СЗИ НСД «Аккорд-АМДЗ»;
- специальное программное обеспечение разграничения доступа в среде ОС Windows.

Защитные функции комплекса реализуются применением:

1. Защитой от НСД ПЭВМ, включая:

- идентификацию пользователя по уникальному ТМ-идентификатору;
- аутентификацию с учетом необходимой длины пароля и времени его жизни;

- аппаратный (до загрузки ОС) контроль целостности технических средств ПЭВМ, программ и данных на жестком диске (в том числе и системных областей диска);

- ограничение времени доступа субъекта к ПЭВМ в соответствии с установленным режимом работы пользователей;

- блокировку несанкционированной загрузки ПЭВМ с отчуждаемых носителей (FDD, CD-ROM, ZIP-drive).

2. Процедурой блокирования экрана и клавиатуры по команде пользователя или по истечению установленного интервала "неактивности" пользователя.

3. Разграничением доступа к ресурсам ПЭВМ в соответствии с установленными правилами разграничения доступа и определяемыми атрибутами доступа, которые устанавливаются администратором безопасности в соответствие каждой паре "субъект доступа - объект доступа" при регистрации пользователей.

Комплекс СЗИ НСД "Аккорд" позволяет использовать как дискреционный, так и мандатный принципы разграничения доступа.

4. Управлением процедурами ввода/вывода на отчуждаемые носители информации.

5. Контролем целостности критичных с точки зрения информационной безопасности программ и данных. Кроме процедур, выполняемых контроллером комплекса в программной части комплекса, возможна проверка целостности программ и данных по индивидуальному списку для каждого пользователя.

6. Средствами функционального замыкания информационных систем за счет использования защитных механизмов комплекса.

#### Подсистема управления доступом

Предназначена для защиты ПЭВМ от посторонних пользователей, управления доступом к объектам доступа и организации совместного их использования зарегистрированными пользователями в соответствии с установленными правилами разграничения доступа (ПРД).



Защита от посторонних пользователей обеспечивается процедурами идентификации (сравнение предъявленного ТМ-идентификатора с перечнем зарегистрированных на ПЭВМ) и аутентификации (подтверждение подлинности) с защитой от раскрытия пароля. Для идентификации (аутентификации) пользователей используются персональные идентификаторы DS-1992-1996.

В СЗИ НСД «Аккорд» реализованы принципы дискреционного и мандатного управления доступом. При использовании дискреционного управления зарегистрированному пользователю устанавливаются права доступа по принципу регистрации "белого списка" разрешенных к запуску программ (задач) и данных, а также "черного списка" запрещенных ресурсов, которые прописываются в ПРД. При использовании мандатного управления пользователю (субъекту) устанавливается уровень доступа, а объекту (данным или задаче) присваивается метка доступа (гриф). При запросе пользователя на доступ к объекту, в зависимости от уровня полномочий пользователя, разрешается или запрещается запрошенный тип доступа. Возможно использование одновременно двух механизмов доступа.

#### Подсистема регистрации и учета

Предназначена для регистрации в системном журнале событий, обрабатываемых комплексом «Аккорд-АМДЗ» (комплекс доверенной загрузки) и подсистемой разграничения доступа СЗИ НСД «Аккорд». При регистрации событий в системном журнале указываются:

- дата и время события;
- пользователь, осуществляющий регистрируемое действие;
- действия пользователя (сведения о входе/выходе пользователя в/из системы, запуске программ, фактах НСД и другие события).

#### Подсистема обеспечения целостности

Предназначена для исключения несанкционированных модификаций (как случайных, так и преднамеренных) конфигураций технических средств ПЭВМ, программной среды, обрабатываемой информации, обеспечивая при этом защиту ПЭВМ от внедрения программных закладок и вирусов.

Контроль целостности в комплексе реализуется:

- проверкой целостности конфигурации технических средств ПЭВМ перед каждым сеансом работы пользователя;
- проверкой целостности назначенных для контроля системных файлов, пользовательских программ и данных;
- исключением возможности использования ПЭВМ без контроллера комплекса;
- механизмом создания замкнутой программной среды, запрещающей запуск посторонних программ.

Функционирование подсистемы обеспечения целостности в комплексах семейства «Аккорд» основано на использовании следующих механизмов:

- при проверке на целостность вычисляется контрольная сумма файлов и сравнивается с эталонным (контрольным) значением, хранящимся в базе данных пользователей. Эти данные заносятся в энергонезависимую память контроллера комплекса при регистрации пользователя и могут изменяться в процессе эксплуатации ПЭВМ;
- для исключения фактов не обнаружения модификации файла используется алгоритм расчета контрольных сумм на основе вычисления значения их хэш-функций. Эталонное (контрольное) значение хэш-функции контрольной суммы хранится вне ПЭВМ - в энергонезависимой памяти контроллера, тем самым защищается от несанкционированной модификации.
- защита от модификации программы расчета хэш-функций обеспечивается тем, что она хранится в памяти контроллера комплекса;
- при контроле целостности индивидуального списка файлов пользователя результирующая хэш-функция хранится на жестком диске, но подписывается кодом аутентификации (КА) с использованием секретного ключа пользователя;
- секретный ключ пользователя формируется из последовательности случайных чисел и записывается в ТМ-идентификатор пользователя при регистрации. Этот секретный ключ используется при выработке кода аутентификации

хэш-функции и исключает возможность несанкционированной модификации файлов из индивидуального списка контролируемых файлов.

### Программно-аппаратные комплексы Secret Disk и Safe Disk

Данные средства предназначены для защиты конфиденциальной информации на жестких дисках и съемных носителях от несанкционированного доступа, копирования, повреждения, кражи или принудительного изъятия. Для защиты информации при хранении используется метод «прозрачного» шифрования с помощью стойких алгоритмов. При этом используется следующий принцип работы.

1. Создается контейнер в виде зашифрованного файла на диске или внешнем носителе и устанавливается способ его защиты: пароль, файл-ключ или электронный ключ.

2. При подключении контейнер отображается в системе как обычный диск, на который можно сохранять конфиденциальную информацию.

3. При сохранении информации данные прозрачно шифруются, при считывании - расшифровываются. Этот процесс не отнимает много времени и можно работать с документами в обычном режиме, но при этом информация надежно защищена.

4. При отключении контейнер перестает отображаться в системе, установить сам факт наличия конфиденциальной информации и получить к ней доступ невозможно.

5. Чтобы восстановить доступ к ранее сохраненной в контейнере конфиденциальной информации и продолжить работу с ней, необходимо подключить контейнер. Для этого необходим пароль, файл-ключ или электронный ключ, в зависимости от выбранного способа защиты.

Таким образом, записанные на жестком диске данные всегда зашифрованы, что делает доступ к ним невозможным для злоумышленника, даже в случае кражи или изъятия как отдельного диска, так и всего компьютера. Для получения доступа к зашифрованной информации используется двухфакторная аутентификация с помощью аппаратного средства аутентификации – USB-ключа или

смарт-карты. Для лиц, не прошедших процедуру аутентификации, скрывается сам факт наличия зашифрованной информации на компьютере.

Данные средства необходимо использовать в следующих случаях:

- конфиденциальная информация обрабатывается и хранится на ноутбуке, есть риск его кражи или несанкционированного использования посторонними;
- необходимо исключить возможность загрузки ОС неавторизованными пользователями;
- за компьютером работает несколько пользователей и есть риск несанкционированного доступа, случайной или преднамеренной порчи, искажения информации;
- конфиденциальная информация переносится на съемных носителях и есть риск их утери или кражи;
- работник IT-отдела, обладая административными привилегиями, необходимыми для обслуживания компьютеров организации, может получить доступ к конфиденциальной информации на жестком диске компьютера.
- необходимо защитить временные и системные файлы, которые хранят информацию о сеансах работы пользователя в различных прикладных программах, историю работы пользователя в сети Интернет, включая пароли доступа к различным сетевым ресурсам, историю переписки пользователя по электронной почте или в системах обмена мгновенными сообщениями;
- конфиденциальная информация находится на жестком диске компьютера или сервера, который передается для технического обслуживания в IT-отдел или внешнюю организацию;
- необходимо обеспечить доступ к конфиденциальной информации лишь одному или нескольким сотрудникам и не допустить ее попадания в чужие руки, а также скрыть сам факт наличия определенных программ и данных;
- необходимо обеспечить экстренное прекращение доступа к данным или уничтожение данных на сервере в случае возникновения нештатных ситуаций (проникновение в офис злоумышленников).

### Электронный замок «Соболь»

Электронные замки «Соболь» предназначены для защиты ресурсов компьютера от несанкционированного доступа.

Электронный замок «Соболь» может применяться как устройство, обеспечивающее защиту автономного компьютера, а также рабочей станции или сервера, входящих в состав локальной вычислительной сети.

К основным возможностям электронного замка относятся:

- идентификация и аутентификация пользователей;
- регистрация попыток доступа к ПЭВМ;
- запрет загрузки ОС со съемных носителей;
- контроль целостности программной среды.

Возможности по идентификации и аутентификации пользователей, а также регистрация попыток доступа к ПЭВМ не зависят от типа используемой операционной системы.

Каждый пользователь компьютера регистрируется в системе электронного замка «Соболь», установленной на данном компьютере. Регистрация пользователя осуществляется администратором и состоит в определении имени регистрируемого пользователя, присвоении ему персонального идентификатора и назначении пароля.

Действие электронного замка «Соболь» состоит в проверке персонального идентификатора и пароля пользователя при попытке входа в систему. В случае попытки входа в систему не зарегистрированного пользователя, электронный замок «Соболь» регистрирует попытку НСД, осуществляет аппаратную блокировку до 4-х внешних устройств, например: FDD, CD-ROM, ZIP.

В электронном замке используются идентификаторы Touch Memory. Загрузка операционной системы с жесткого диска осуществляется только после предъявления зарегистрированного идентификатора. Служебная информация о регистрации пользователя (имя, номер присвоенного персонального идентификатора и т.д.) хранится в энергонезависимой памяти электронного замка.

Электронный замок «Соболь» осуществляет ведение системного журнала, записи которого хранятся в специальной энергонезависимой памяти. Электронный замок фиксирует в системном журнале вход пользователей, попытки входа, попытки НСД и другие события, связанные с безопасностью системы.

В системном журнале хранится следующая информация: дата и время события, имя пользователя и информация о типе события, например:

- факт входа пользователя;
- введение неправильного пароля;
- предъявление не зарегистрированного идентификатора пользователя;
- превышение числа попыток входа в систему;
- другие события.

Таким образом, электронный замок «Соболь» предоставляет информацию администратору обо всех попытках доступа к ПЭВМ.

Подсистема контроля целостности расширяет возможности электронного замка «Соболь». Контроль целостности системных областей дисков и наиболее критичных файлов производится по алгоритму ГОСТ 28147-89 в режиме имитовставки. Администратор имеет возможность задать режим работы электронного замка, при котором будет блокирован вход пользователей в систему при нарушении целостности контролируемых файлов.

Подсистема запрета загрузки с гибкого диска и CD обеспечивает запрет загрузки операционной системы с этих съемных носителей для всех пользователей компьютера, кроме администратора. Администратор может разрешить отдельным пользователям компьютера выполнять загрузку операционной системы со съемных носителей.

## **4. ЛАБОРАТОРНЫЕ РАБОТЫ**

### **4.1. Лабораторная работа № 1**

**Наименование лабораторной работы** – Защита программного обеспечения с помощью электронных ключей HASP.

**Время на выполнение** – 4 часа.

**Цель** – изучение способов защиты программного обеспечения от несанкционированного копирования на примере электронных ключей HASP.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками защиты программного обеспечения от несанкционированного копирования с применением электронных ключей семейства HASP.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, электронные ключи HASP 4.

#### **Краткие теоретические сведения:**

Электронные ключи – физические устройства, предназначенные для защиты программных продуктов от несанкционированного использования, а именно от нелегального копирования, распространения, нарушения лицензионных соглашений и т.д. Устройство HASP представляет собой электронный ключ, разработанный фирмой Aladdin. Передовые возможности шифрования и расшифрования информации с помощью аппаратных средств HASP позволяют осуществлять тесную интеграцию аппаратных средств с защищаемым ПО. Возможности самого ключа позволяют критическим функциям защищаемого приложения быть доступными в зависимости от наличия корректного ключа (в противном случае эти функции будут недоступны).

Любые данные, используемые приложением, могут быть зашифрованы. Полученная зашифрованная информация является функцией от данных, посланных на HASP, и уникального кода разработчика.

Все утилиты HASP используют собственные системы защиты кода и алгоритмов. В дополнение к этому, HASP использует передовые технологии защиты от декомпиляции. Специальные антихакерские уловки, применяемые в системе HASP, создают достаточно серьезные препятствия для взломщика. Обмен данными между ключом и защищенным приложением зашифрован.

Во время работы приложение запрашивает HASP, подключенный к ПЭВМ. Если HASP возвращает ответ с использованием корректного алгоритма, то приложение выполняется. В противном случае приложение не загружается, либо может переключиться в демонстрационный режим, закрыть свои основные возможности, или выполнить иное действие на усмотрение разработчика.

При заказе ключей в компании Aladdin разработчик ПО получает ключи HASP со встроенной информацией о разработчике. Эта информация используется для того, чтобы отличать ключи разных производителей. Код разработчика представляет собой уникальный код, который компания Aladdin присваивает каждому разработчику ПО. Код прошивается на чипе, чтобы не допустить декомпиляции. Таким образом повышается степень защищенности приложений.

Пароли HASP – это два целых 16-битовых числа, присвоенные каждому разработчику. Пароль базируется на коде разработчика. Только зная пароль можно получить доступ к HASP, защищать приложения и получать доступ к утилитам HASP. Пароль прошит в аппаратный контроллер ключа, зашифрован и не может быть изменен. Это обусловлено требованиями защиты от декомпиляции.

Присутствие корректного ключа HASP может быть проверено следующими методами:

- 1) с использованием механизма шифрования, основывающемся на присутствии ключа;
- 2) проверкой специфичного для ключа ID-номера;



3) с использованием функций памяти ключа.

#### Использование механизма шифрования

Система HASP производит эту проверку методом шифрования и расшифрования данных с использованием самого ключа. Для того, чтобы расшифровать данные, нужно послать уже зашифрованные данные ключу. Далее можно проверить, корректны ли расшифрованные данные. Если они корректны, значит ключ присутствует в системе.

Расшифрованные данные могут быть проверены с использованием данных самого приложения. Зашифрованные данные являются функцией уникального кода разработчика и собственных данных программы. Таким образом, шифрование одной и той же строки ключами разных разработчиков приведет к разным результатам.

#### Проверка ID-номера HASP

Каждый ключ HASP имеет свой уникальный ID-номер, который защищаемое приложение может проверить. Ключи HASP со встроенными ID-номерами помогают различать пользователей ПО. Проверяя данный номер, всегда можно точно сказать, присутствует ли конкретный ключ в системе или нет.

#### Использование функций памяти ключа

Все ключи HASP (кроме HASP Standard) содержат внутреннюю память. Эту память можно использовать, чтобы:

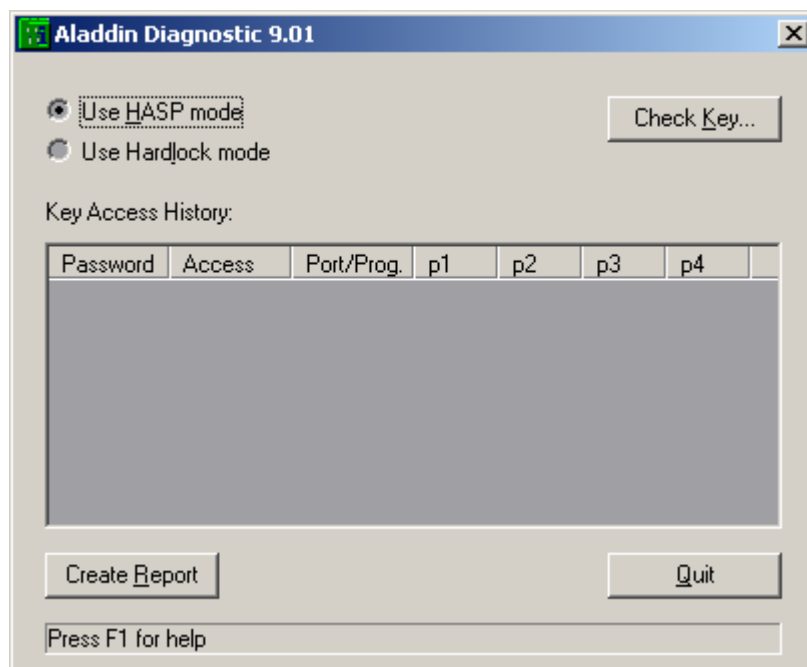
- 1) контролировать доступ к модулям ПО или различным приложениям;
- 2) присваивать уникальный ID-номер каждому пользователю;
- 3) распространять демонстрационные версии, которые могут быть активированы ограниченное количество раз;
- 4) сохранять пароли, код программы, переменные или данные.

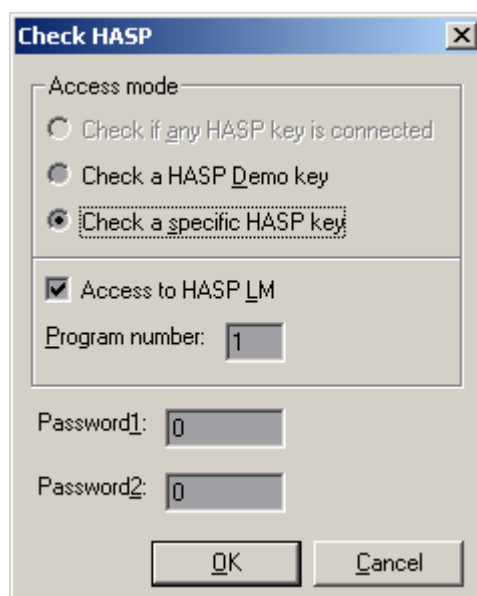
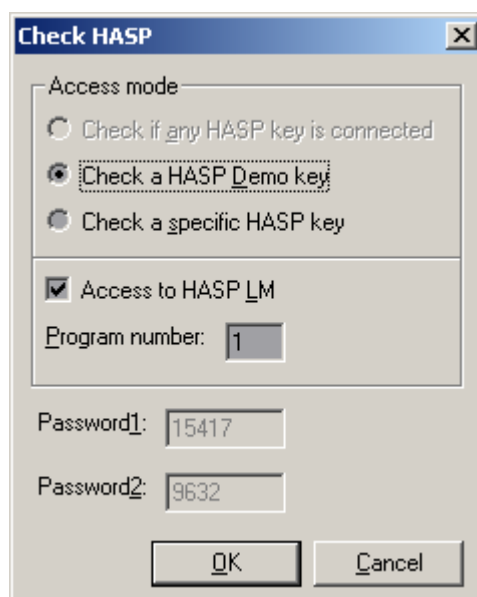
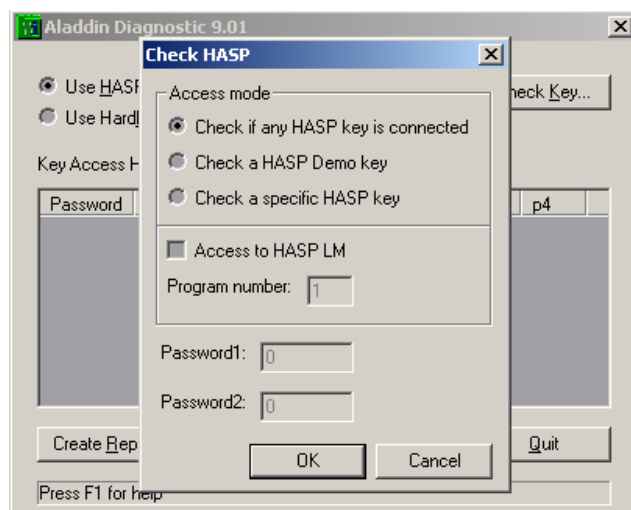
### Порядок выполнения лабораторной работы:

Пароли доступа к ключам HASP:	
Код ключа: <b>RAOMG</b>	Пароль №1: <b>21746</b> . Пароль №2: <b>5538</b> .
Код ключа: <b>IXGGR</b>	Пароль №1: <b>32113</b> . Пароль №2: <b>21797</b> .
Код ключа: <b>DEMOMA</b>	Пароль №1: <b>15417</b> . Пароль №2: <b>9632</b> .

#### 1. Работа с утилитой Aladdin Diagnostic.

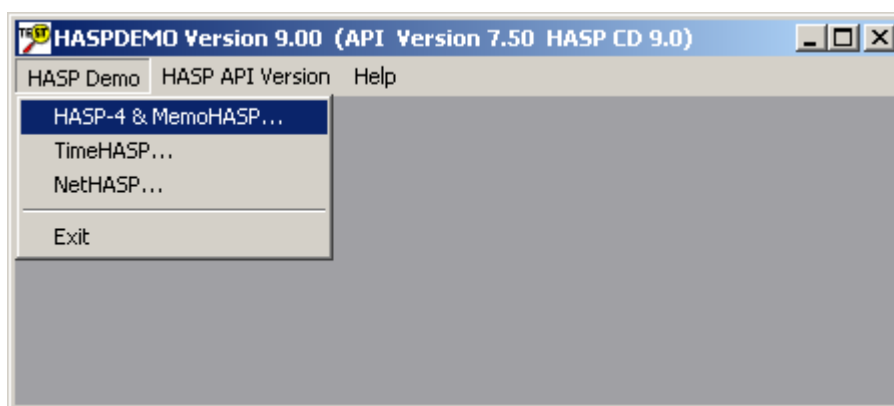
- 1.1. Подключить к LPT порту ключ HASP4.
- 1.2. Запустить утилиту Aladdin Diagnostic и попытаться диагностировать на ЭВМ ключ HASP. Вначале – любой, затем – демонстрационный, затем - с заданным кодом доступа (верным и неверным). Какова реакция системы на данные действия (внесите в отчет)? Внесите в отчет верные пароли к ключам HASP, которые вы использовали.



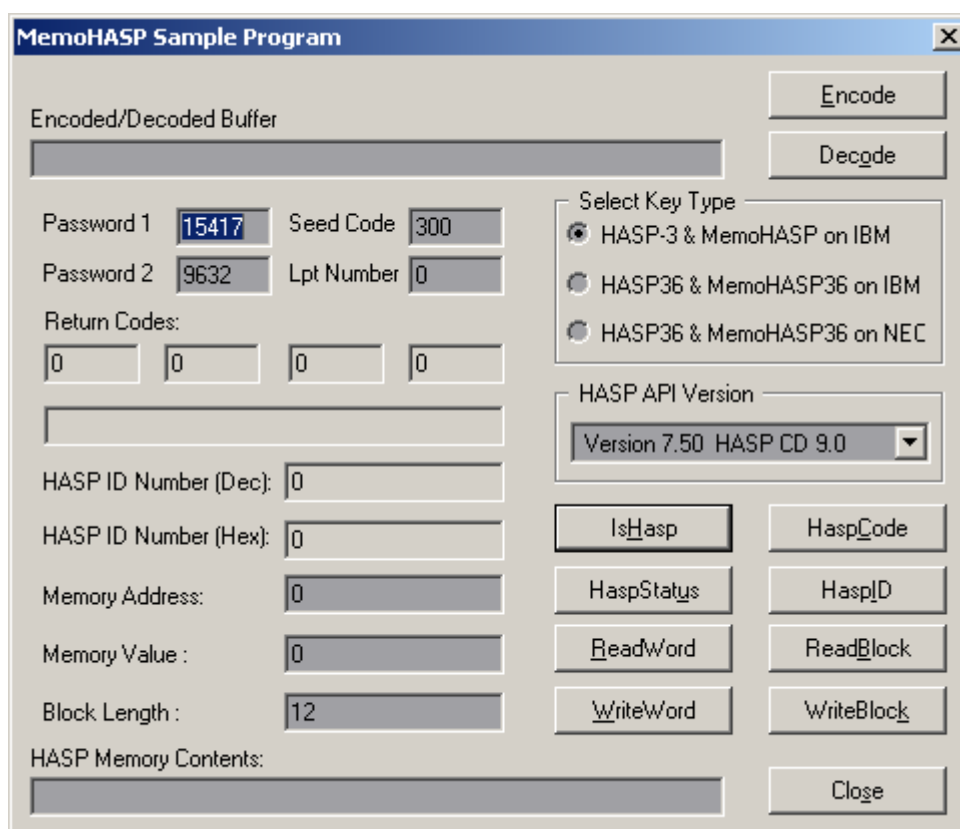


2. Работа с утилитой HASP Test. С помощью этой утилиты возможна проверка подключения конкретного HASP к ПК и тестирование всех основных функций ключа.

- 2.1. Подключить HASP4 Standard к ПК, запустить утилиту HASP Test for Win32 и выбрать соответствующий тип ключа HASP.



- 2.2. В полях Password-1 и Password-2 ввести пароли.



- 2.3. Какие функции доступны для HASP4 Standard (внести в отчет)?
- 2.4. Введите в поле SeedCode произвольное значение и получите отклик аппаратной функции ключа при послылке в нее данного значения. Сколько двойных слов возвращает данная функция (внести в отчет)?

**MemoHASP Sample Program**

Encoded/Decoded Buffer

Encode

Decode

Select Key Type

☒ HASP-3 & MemoHASP on IBM

☐ HASP36 & MemoHASP36 on IBM

☐ HASP36 & MemoHASP36 on NEC

HASP API Version

Version 7.50 HASP CD 9.0

IsHasp

HaspCode

HaspStatus

HaspID

ReadWord

ReadBlock

WriteWord

WriteBlock

Close

Password 1: 15417 Seed Code: 777

Password 2: 9632 Lpt Number: 0

Return Codes: 1 0 0 9200

Incorrect Password

HASP ID Number (Dec): 0

HASP ID Number (Hex): 0

Memory Address: 0

Memory Value : 0

Block Length : 12

HASP Memory Contents:

- 2.5. Получите отклик ключа HASP на произвольные 6 значений SeedCode. Возвращенные значения внесите в отчет.
- 2.6. Зашифруйте с помощью ключа HASP4 Standard любую фразу и расшифруйте ее. Результаты шифрования и шифруемую фразу внесите в отчет.

**МемоHASP Sample Program**

Encoded/Decoded Buffer: Кафедра СИБ

Encode Decode

Password 1: 21746 Seed Code: 777  
 Password 2: 5538 Lpt Number: 0

Return Codes: 1 0 0 9200

Incorrect Password

HASP ID Number (Dec): 0  
 HASP ID Number (Hex): 0

Memory Address: 0  
 Memory Value: 0  
 Block Length: 12

HASP Memory Contents:

Select Key Type:  
☒ HASP-3 & MemoHASP on IBM  
☐ HASP36 & MemoHASP36 on IBM  
☐ HASP36 & MemoHASP36 on NEC

HASP API Version: Version 7.50 HASP CD 9.0

IsHasp HaspCode  
 HaspStatus HaspID  
 ReadWord ReadBlock  
 WriteWord WriteBlock

Close

**МемоHASP Sample Program**

Encoded/Decoded Buffer: r8b3ubCP

Encode Decode

Password 1: 21746 Seed Code: 777  
 Password 2: 5538 Lpt Number: 0

Return Codes: 0 0 0 0

Encode Data was successful.

HASP ID Number (Dec): 0  
 HASP ID Number (Hex): 0

Memory Address: 0  
 Memory Value: 0  
 Block Length: 12

HASP Memory Contents:

Select Key Type:  
☒ HASP-3 & MemoHASP on IBM  
☐ HASP36 & MemoHASP36 on IBM  
☐ HASP36 & MemoHASP36 on NEC

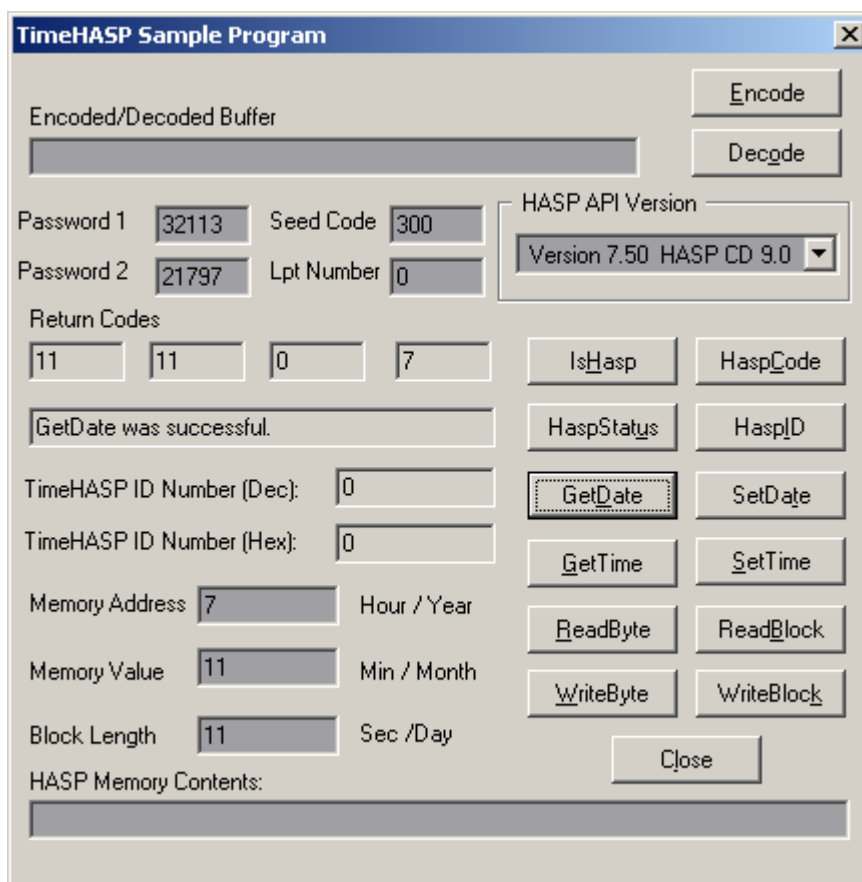
HASP API Version: Version 7.50 HASP CD 9.0

IsHasp HaspCode  
 HaspStatus HaspID  
 ReadWord ReadBlock  
 WriteWord WriteBlock

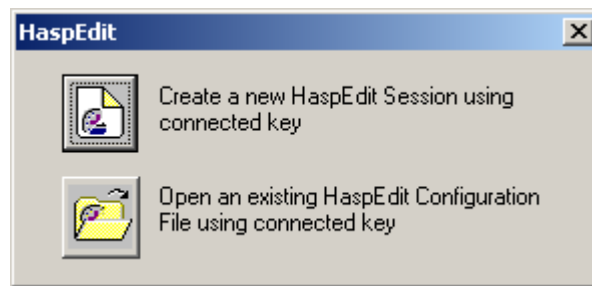
Close

- 2.7. Извлеките HASP4 Standard из ПК и подключите к нему HASP4 Time. Какие дополнительные функции добавились при этом?

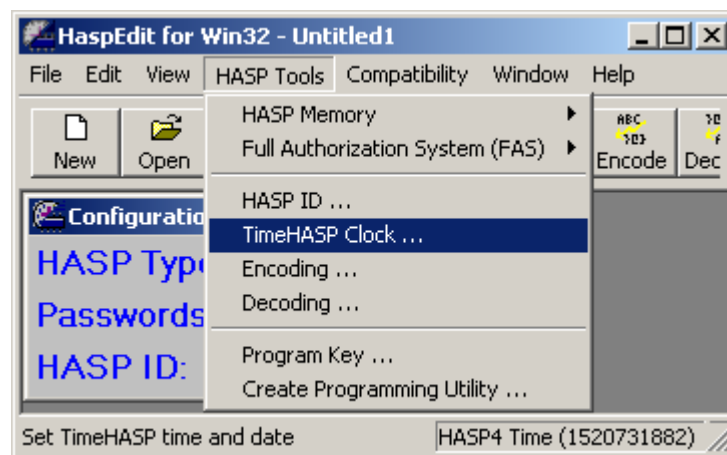
- 2.8. Исследуйте работу функций HASP4 Time с датой и временем. Получите текущую дату и время. Измените эти параметры на другие. Внесите эти данные и использованные команды в отчет.



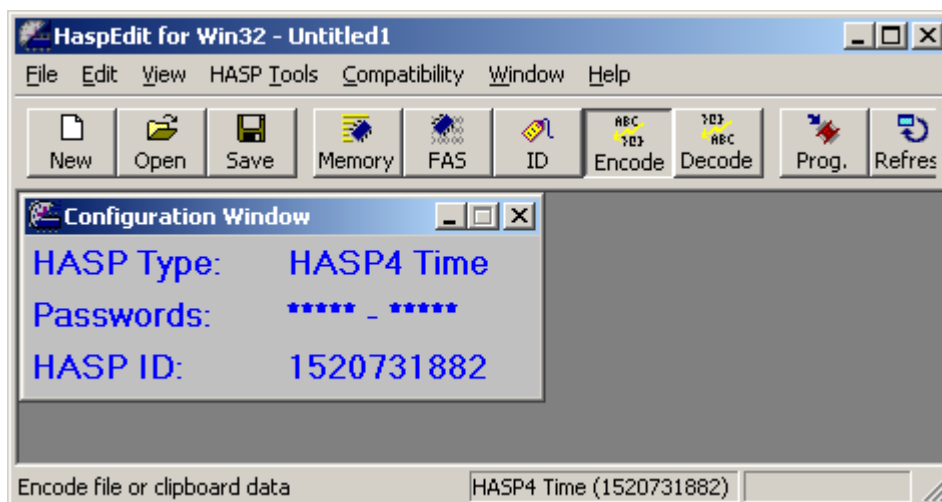
- 2.9. Исследовать работу с энергонезависимой памятью HASP4 Time. Прочитать байт из памяти ключа по адресу 1 (внести в отчет). Записать по этому адресу значение 100. Осуществить чтение заново и убедиться, что данное значение записалось. Использованные при этих операциях команды внести в отчет.
- 2.10. Прочитать из памяти блок длиной 16 байт начиная с адреса 1. Записать туда выражение «Ключ\_доступа». Осуществить чтение заново и убедиться, что данное выражение записалось.
- 2.11. Каков номер ID Вашего электронного ключа HASP (внести в отчет)?
3. Работа с утилитой HASP Edit. Данная утилита позволяет подготовить ключи HASP к распространению и внести в них необходимые данные.
- 3.1. Подключить электронный ключ HASP4 Time к ПК и запустить утилиту HASP Edit for Win32.



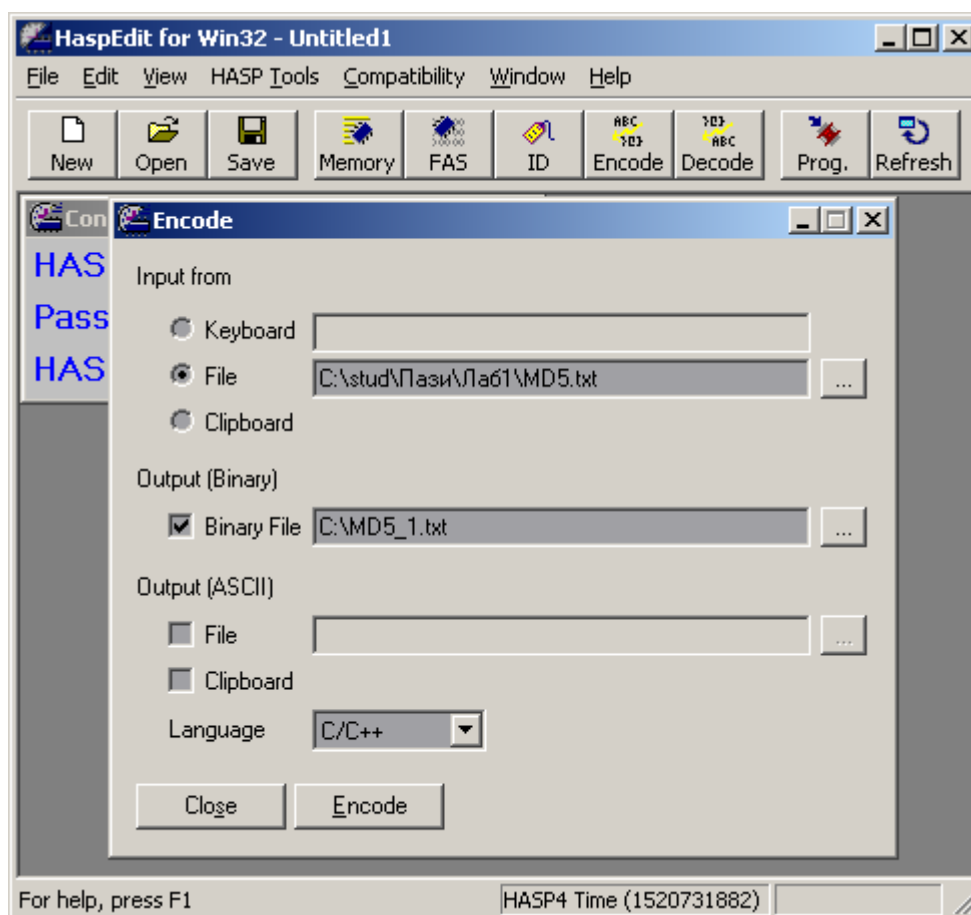
- 3.2. Создать новую сессию и ввести корректные коды доступа.
- 3.3. Познакомиться с основными функциями HASP4 Time, предлагаемые HASP Edit. Внесите их в отчет.
- 3.4. Получите ID номер HASP4 Time.
- 3.5. Вызовите функцию установки даты и времени HASP4 Time. Соответствуют ли данные установки компьютерным? Установите дату и время HASP4 Time реальным.



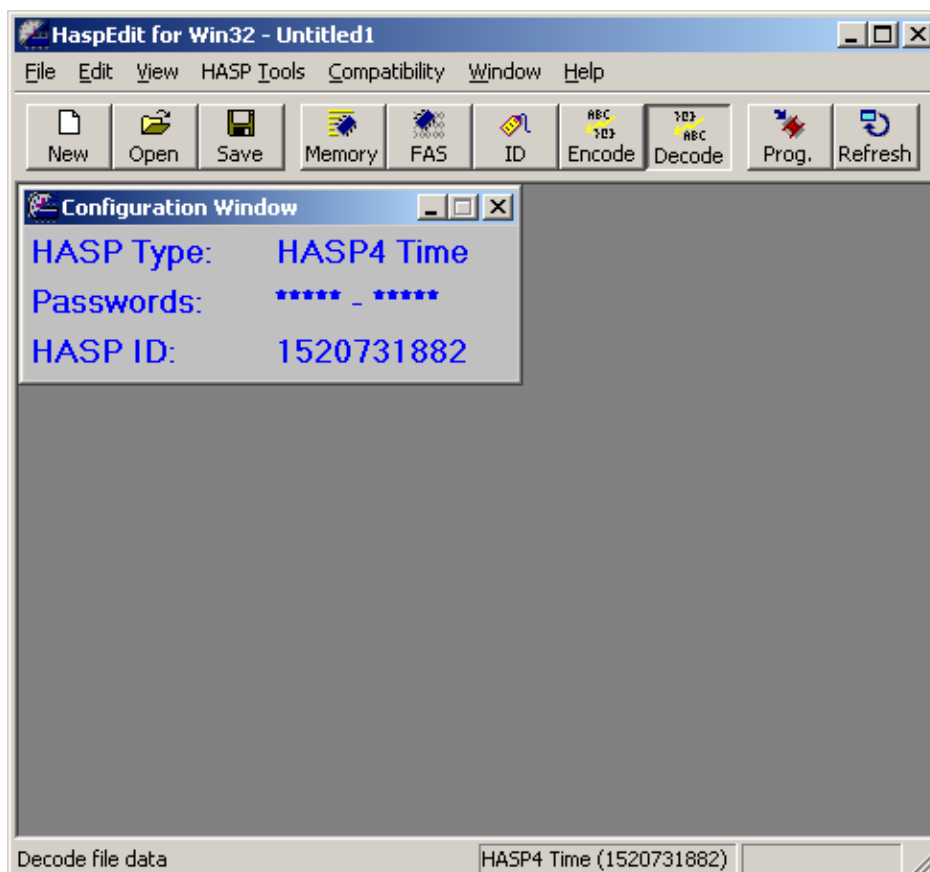
- 3.6. Открыть меню Encoding. Зашифровать с помощью HASP файл MD5.txt, в качестве выходного файла указать бинарный файл MD5\_1, изучить содержимое обоих файлов. Зашифрован ли файл?

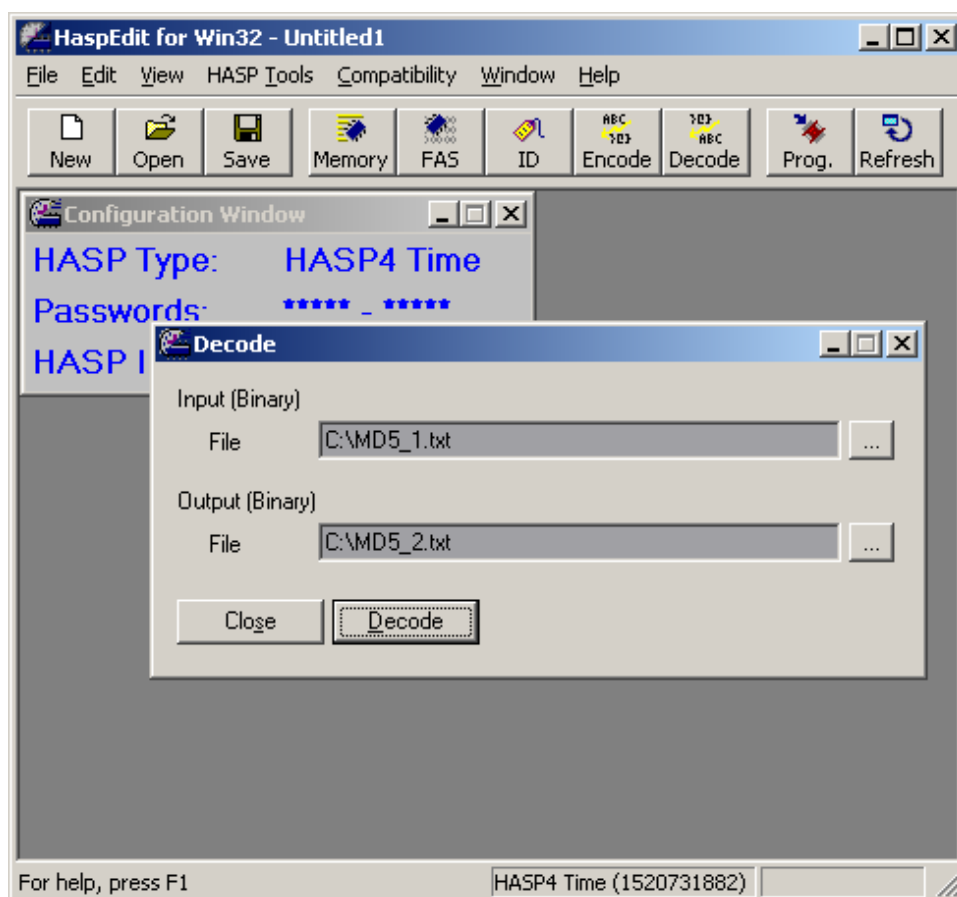




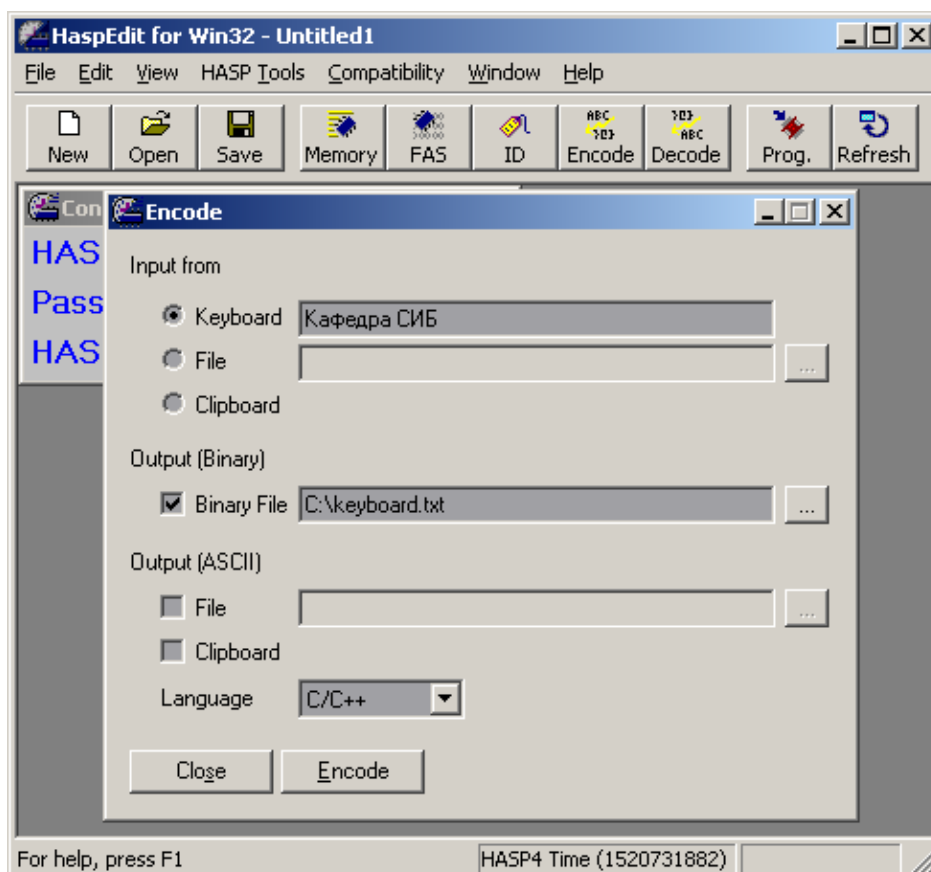


- 3.7. Расшифровать зашифрованный файл и изучить его содержимое. Корректно ли расшифрован файл?

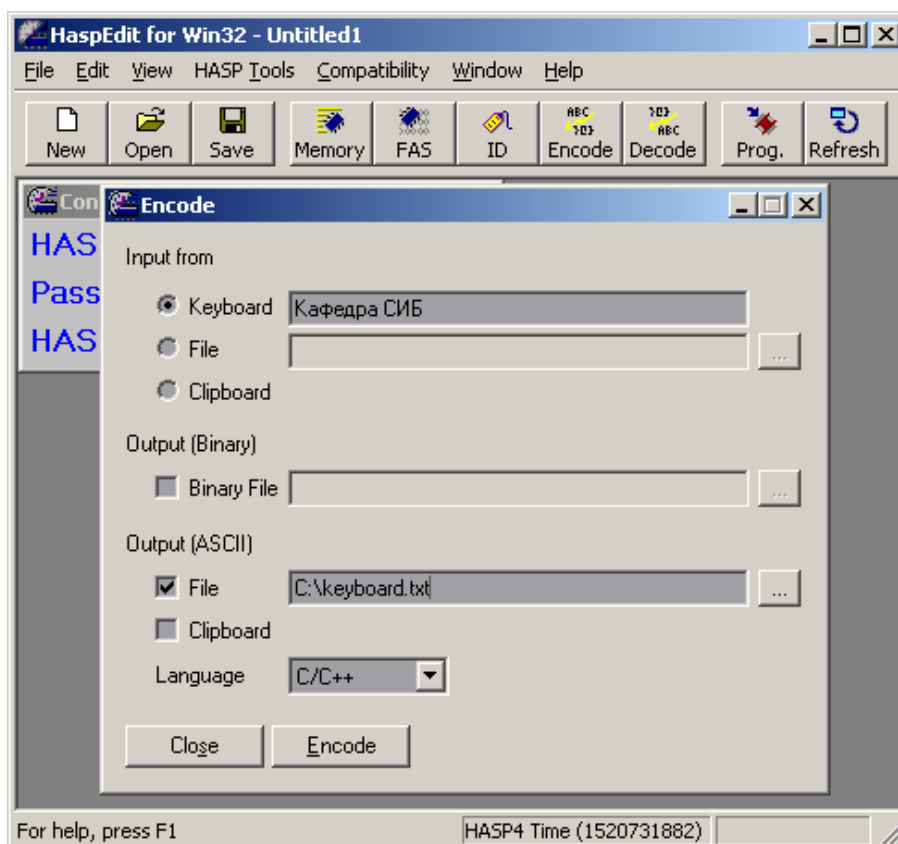




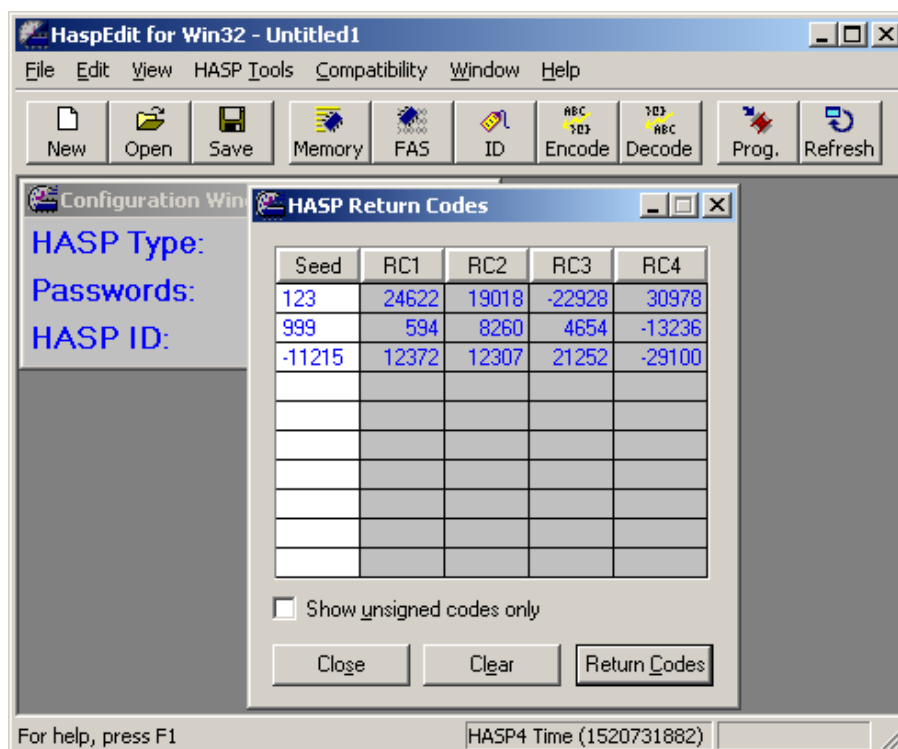
- 3.8. Набрать в строке Keyboard некоторое выражение и зашифровать его. Изучить содержимое зашифрованного файла. Расшифровать его.



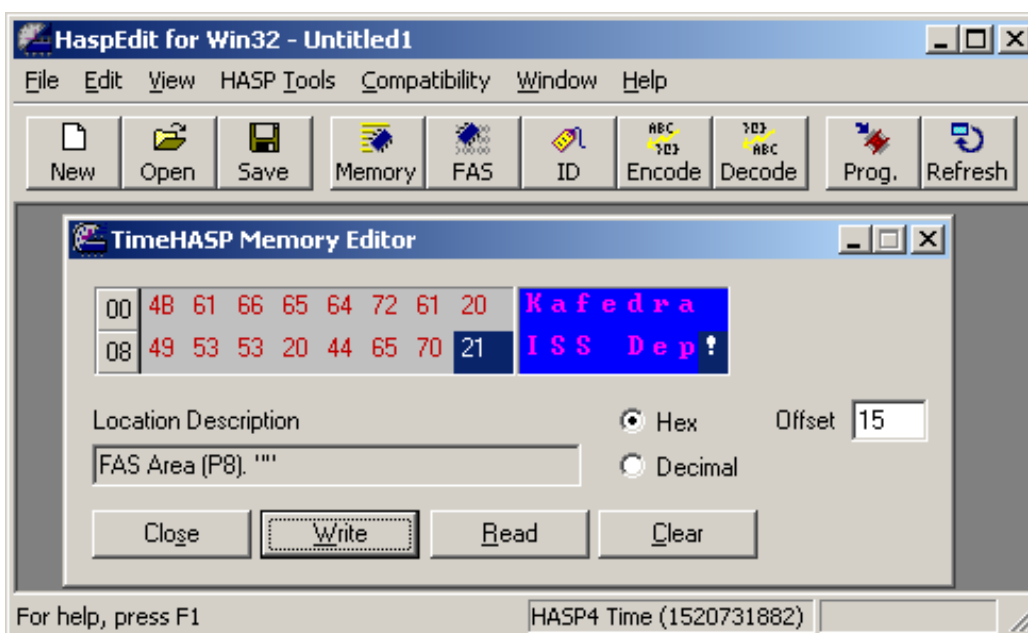
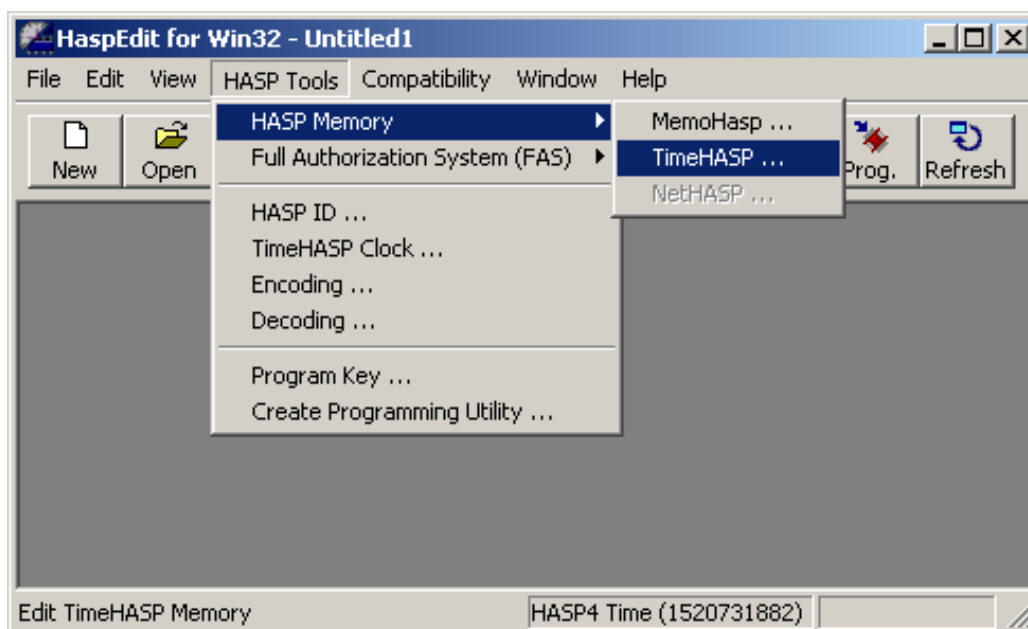
- 3.9. Заменить бинарный вывод на текстовый и повторить пункты 3.6-3.8. В чем особенность зашифрованных файлов в данном случае (внести в отчет)?



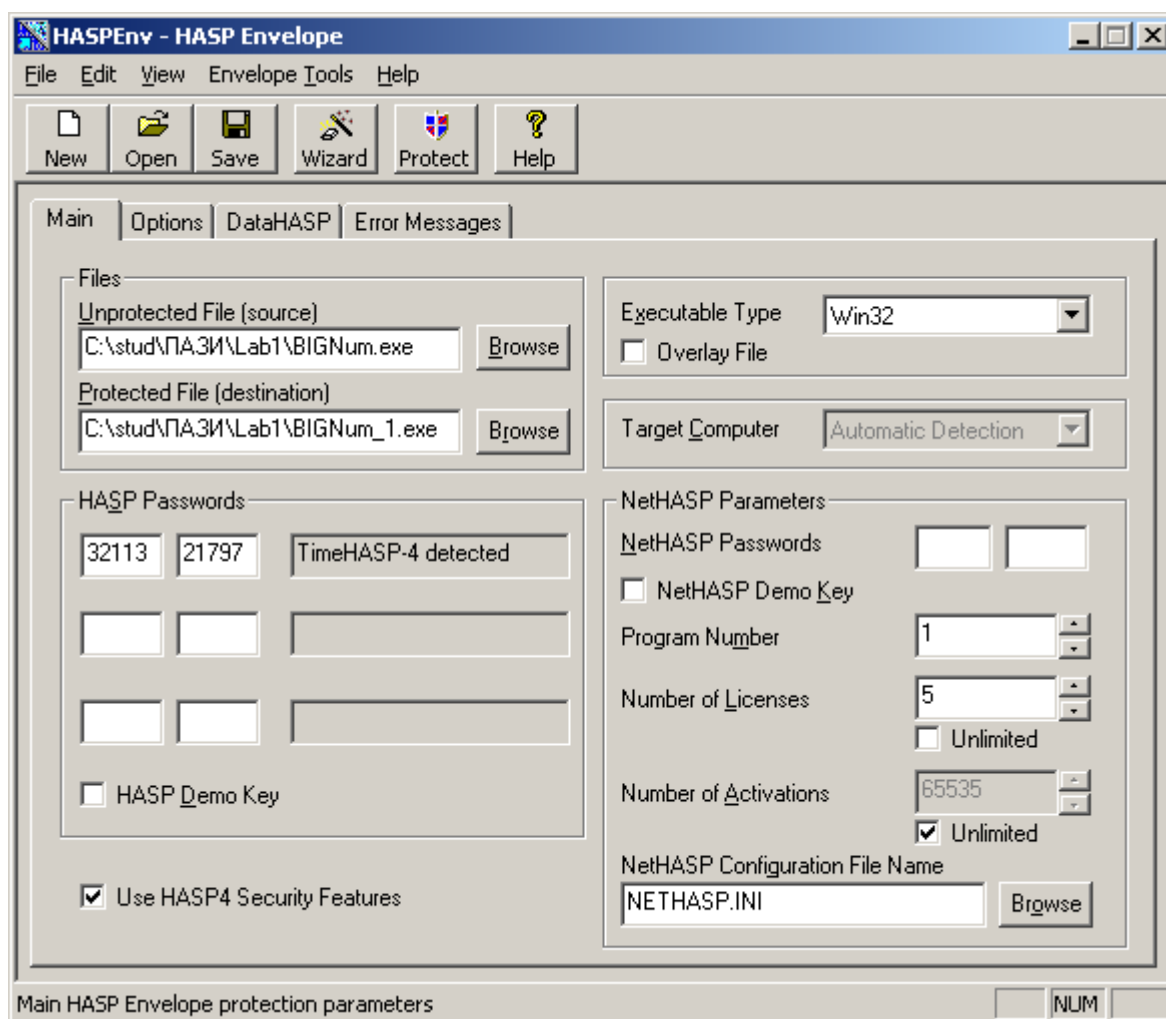
- 3.10. Вызвать меню COMPATIBILITY – RETURN CODES и изучить отклики HASP на различные посылаемые в качестве параметров значения.



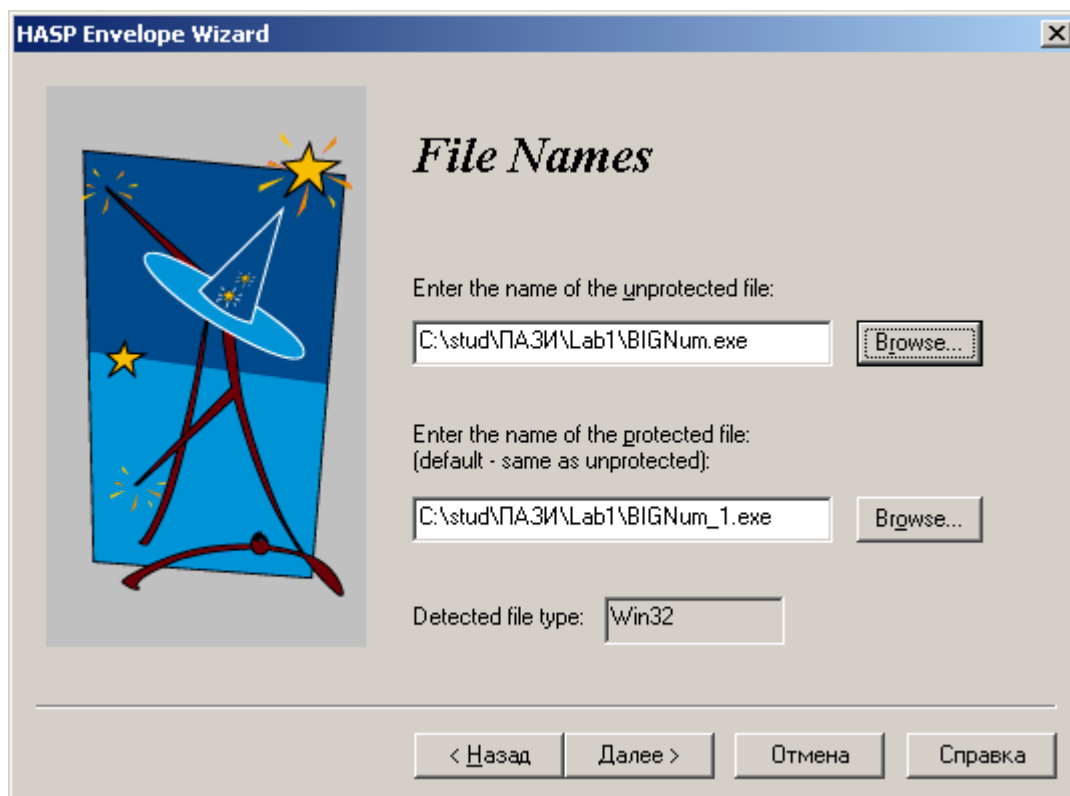
- 3.11. Исследуйте функцию работы с защищенной энергонезависимой памятью ключа HASP4 Time (меню HASP TOOLS – HASP MEMORY – MEMOHASP). Запишите туда некоторый секретный ключ, который должен будет контролироваться защищаемой программой.



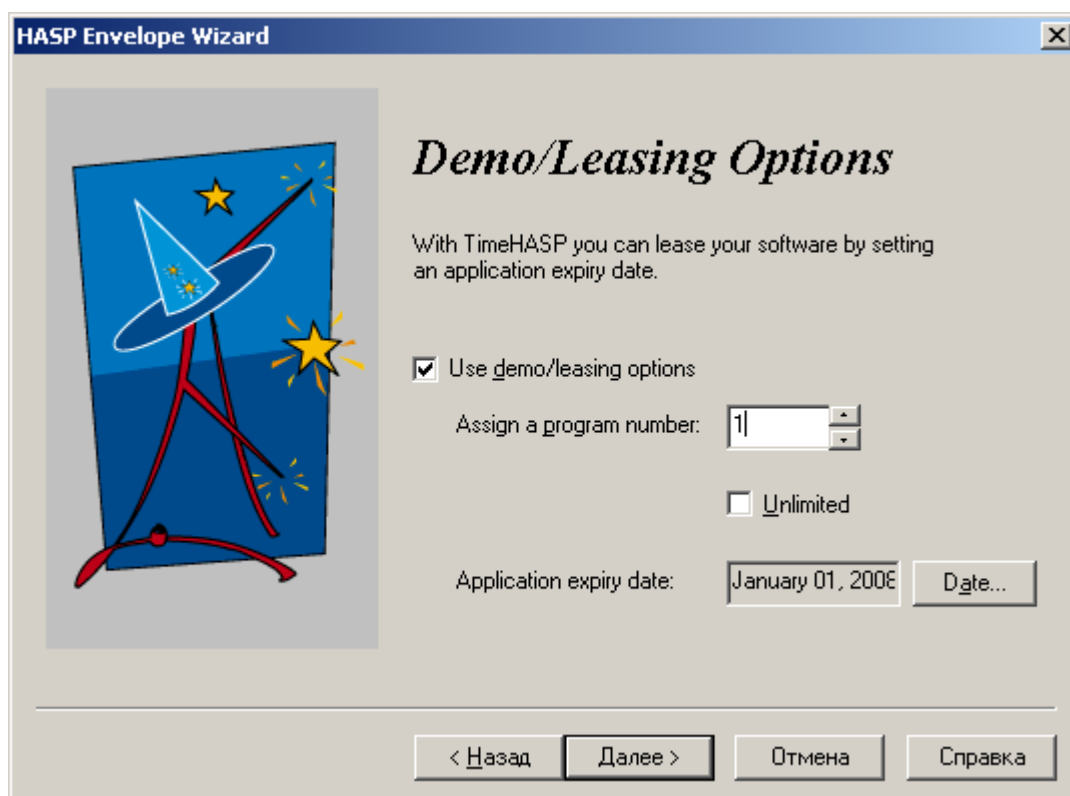
4. Работа с утилитой HASP Envelope. Данная утилита обеспечивает пакетный (пристыковочный) режим защиты исполняемого кода ПО от несанкционированного использования.
- 4.1. Подключить электронный ключ HASP4 Time к ПК и запустить утилиту HASP Envelope.



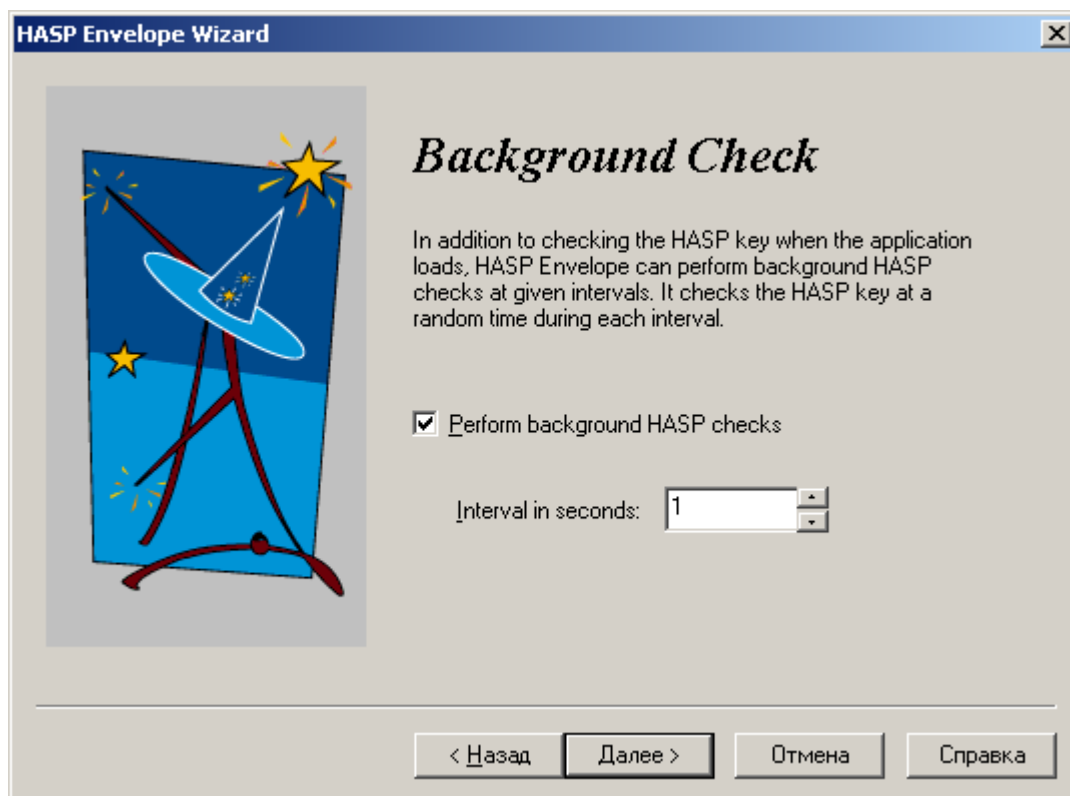
- 4.2. Нажмите кнопку HASP WIZARD.
- 4.3. В качестве защищаемого файла выбрать BIGNum.exe, в качестве выходного – BIGNum\_1.exe.



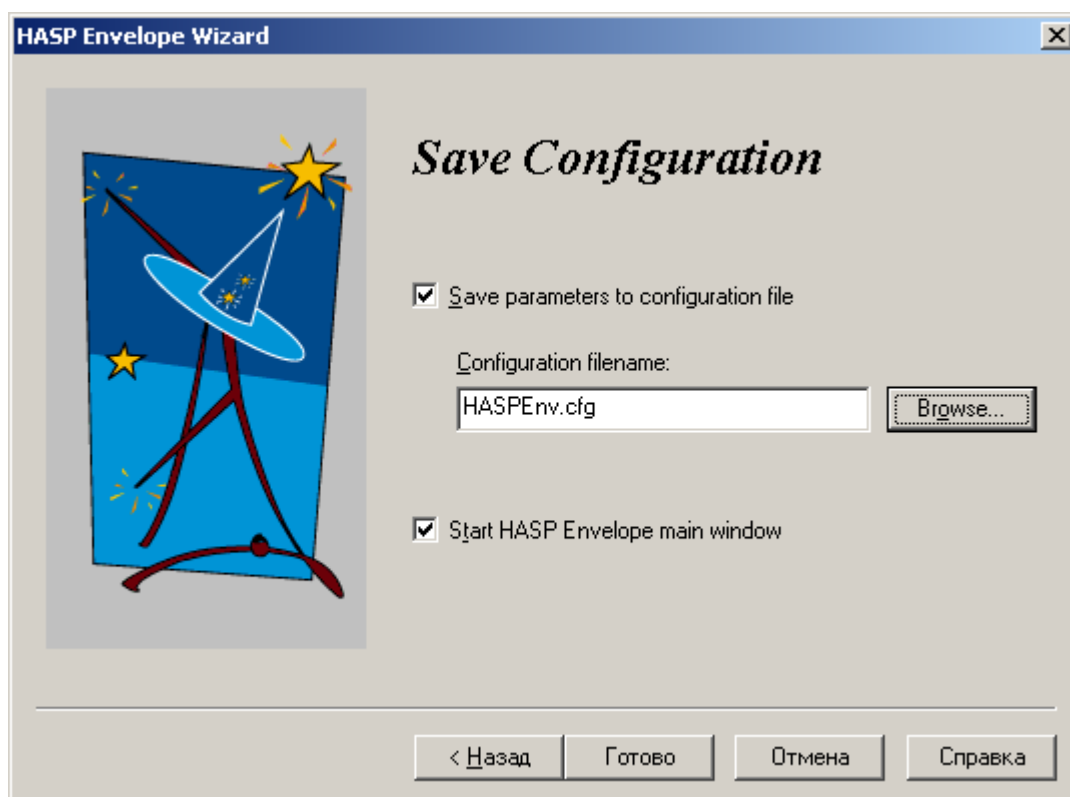
- 4.4. Выбрать опцию STAND-ALONE HASP ONLY.
- 4.5. Ввести код доступа к HASP.
- 4.6. Присвоить программе номер 1 и указать, что она может использоваться до 1 января следующего года.

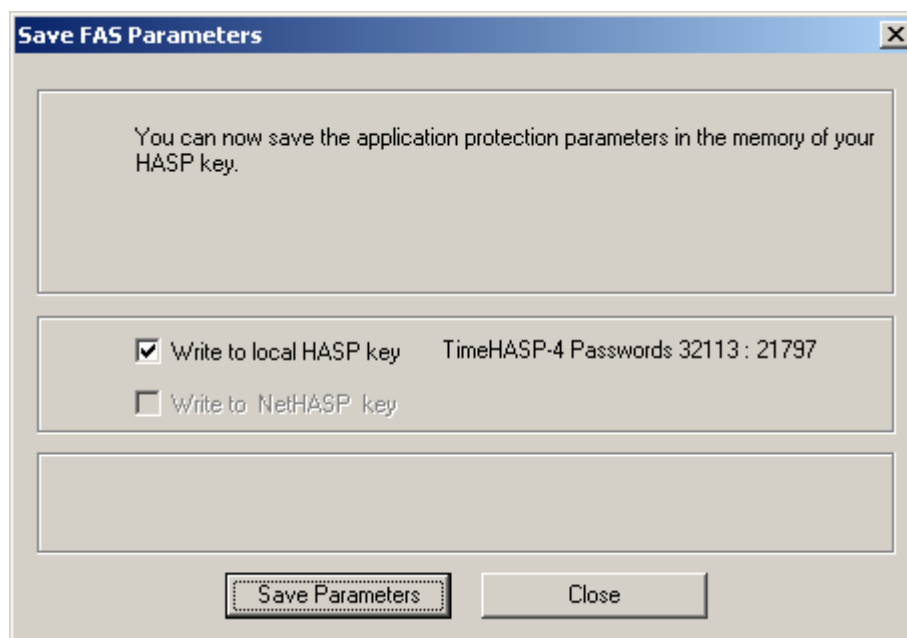
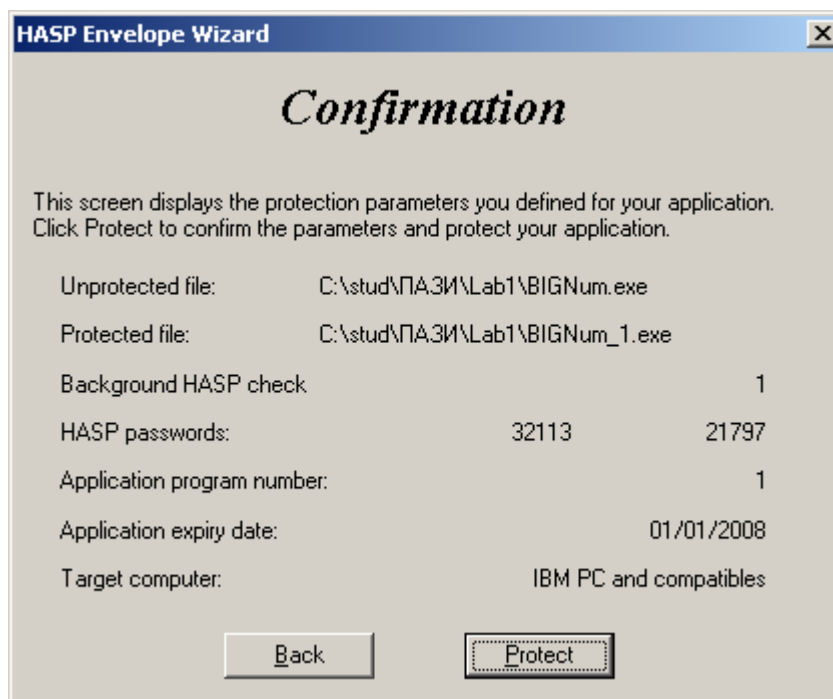


- 4.7. Задать функцию проверки наличия HASP один раз в секунду.

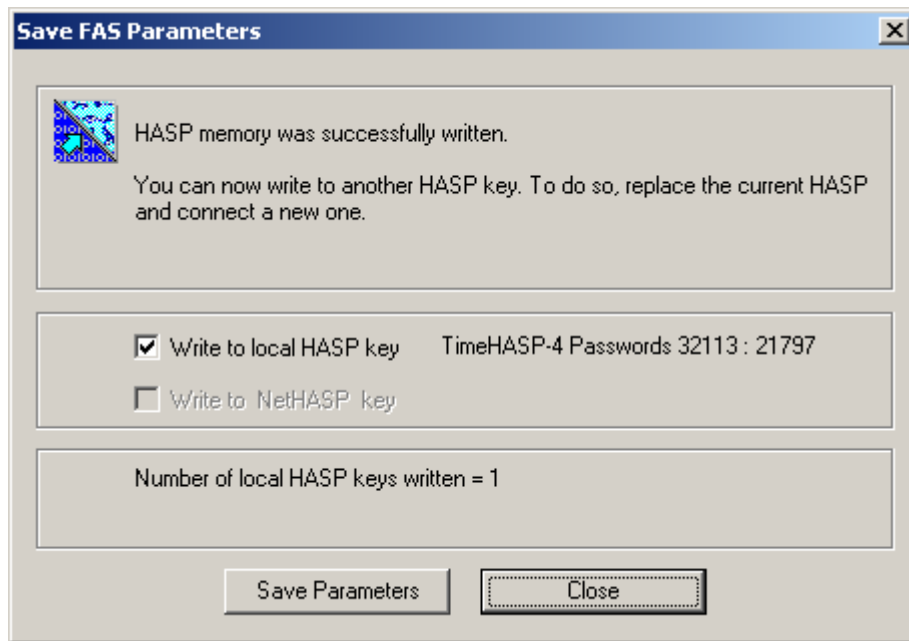


- 4.8. Произвести защиту программы и записать конфигурацию в память HASP.

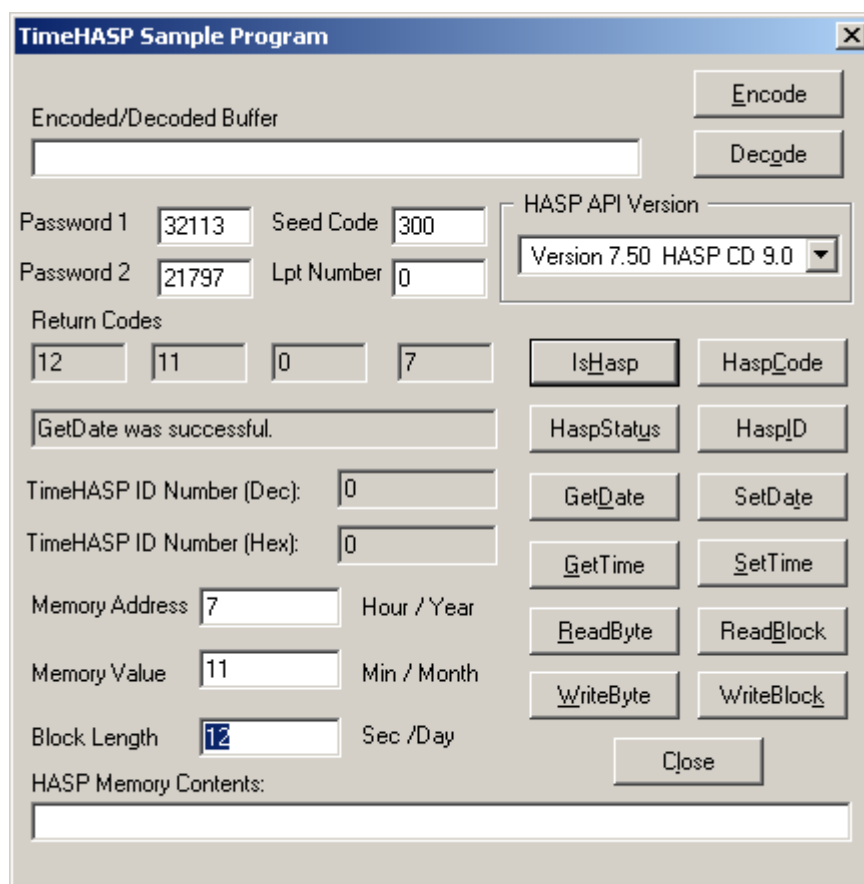








- 4.9. Произвести запуск программы BIGNum.exe с HASP и без него. Убедиться в том, что без HASP программа становится не работоспособной.
- 4.10. Запустить программу с HASP, затем извлечь его. Проследить за реакцией программы (внести в отчет).
- 4.11. Изменить компьютерную дату, например, на год вперед и запустить защищенную программу. Будет ли она работать (внести ответ в отчет)? Вернуть реальную дату.
- 4.12. Изменить дату в самом ключе HASP на 2 января следующего года или более позднюю и запустить защищенную программу. Будет ли она работать (внести ответ в отчет)? Вернуть реальную дату. Для установки даты (или времени) используйте программу HASP TEST. Введите дату (или время) и нажмите кнопку SetDate (или SetTime).



4.13. Изучить в меню OPTIONS главного меню возможные опции защиты.

Прокомментировать их (внести в отчет).

Изучить опции DATAHASP и объяснить их суть.

### Контрольные вопросы:

1. Что такое электронный ключ?
2. Что представляет собой устройство HASP?
3. Какие существуют варианты реализации ключей HASP?
4. Какие возможности защиты самого ключа и защищаемого им приложения предоставляет система HASP?
5. В чем преимущества использования системы HASP?
6. Объясните принцип функционирования ключей HASP.
7. Что из себя представляют код разработчика и пароли HASP? Для каких целей они служат?
8. Какими методами можно проверить присутствие корректного ключа? Опишите каждый из них.

9. Какие основные модели включает в себя семейство ключей HASP4? В чем их основные сходства и различия?
10. Что представляет собой механизм защиты HASP Envelope?
11. Как изменяется порядок запуска программы, защищенной системой HASP Envelope?
12. Какие достоинства и недостатки у пристыковочного механизма защиты ПО?
13. В чем суть защиты ПО с использованием механизма внедрения?
14. Какие достоинства и недостатки у защиты с помощью HASP API?

## **4.2. Лабораторная работа № 2**

**Наименование лабораторной работы** – Применение отладчика OllyDbg для исследования механизмов защиты программного обеспечения.

**Время на выполнение** – 4 часа.

**Цель** – познакомиться на практике с методами динамического исследования эффективности механизмов защиты.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками динамического исследования эффективности механизмов защиты с применением отладчиков.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, отладчик OllyDbg.

### **Краткие теоретические сведения:**

К достоинствам отладчика OllyDbg можно отнести простоту использования, наличие достаточно большого количества плагинов, функциональность и т.п. Все это и определяет его популярность в среде реверсеров. При запуске отладчика открывается рабочее окно (рис. 4.1). Исследуемый процесс может быть выбран как в виде исполняемого PE-файла (File – open, либо F3), так и в виде уже запущенного процесса (File - attach).

На рисунке представлены фрейм с дизассемблированным листингом отлаживаемой программы, фрейм дампа памяти, фрейм регистров CPU, фрейм стека.

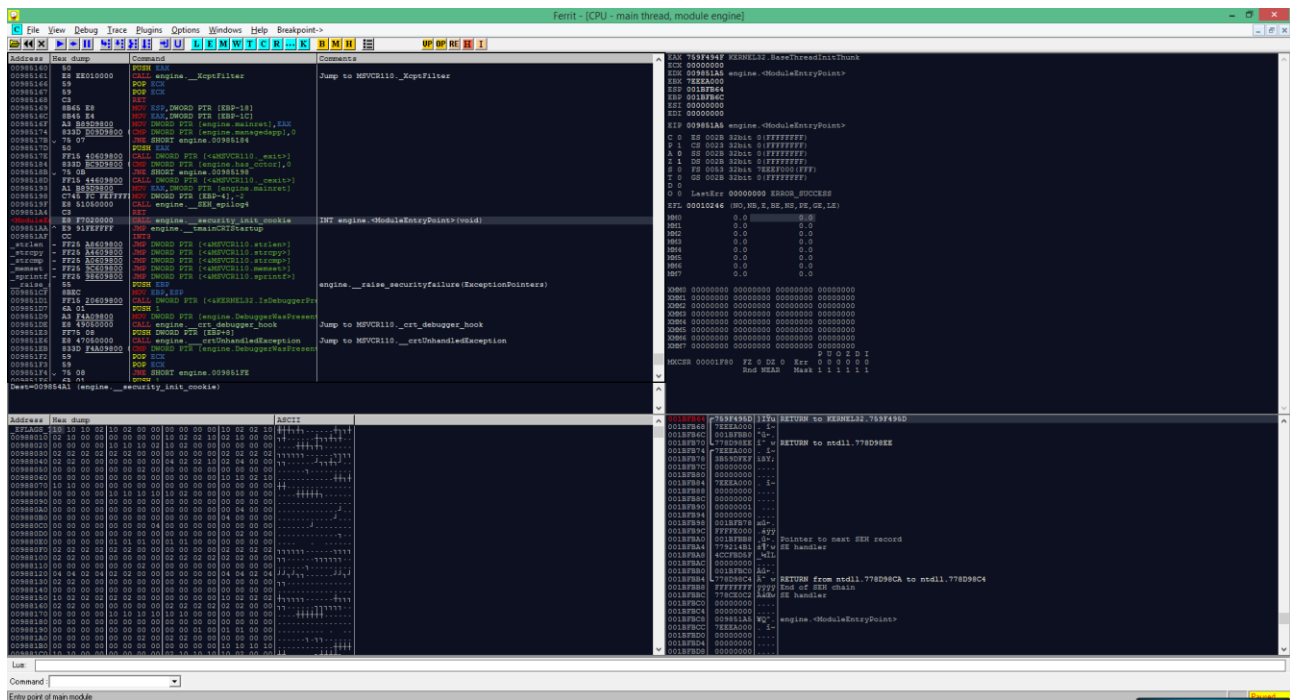


Рис. 4.1. Внешний вид отладчика OllyDbg

### Фрейм отлаживаемого кода

В данном фрейме с кодом программы рекомендуется установить режим подсказки для условных и безусловных переходов (Options – Debugging Options – CPU – поставить галки на Show jump path, Show grayed path if jump is not taken, Show jumps to selected command).

Трассировка программы выполняется следующими клавишами: F7 – с заходом в процедуры CALL address, F8 – без захода в процедуры. Также возможно выполнение программы (клавиша F9) полностью, либо до установленной точки останова, либо автоматическое выполнение кода между двумя точками останова.

Точки останова можно установить следующими способами: ручной режим (выбрать строчку программы и нажать F2) либо через команды BP, BPX. Установленный breakpoint подсвечивается в колонке адресации.

Двойной клик в окне по команде (либо же правый клик на команде Assemble, либо пробел) вызывает окно редактирования кода, что иногда помогает отключить некоторые вызовы функций, изменить направление ветвления, Следует только учитывать, что после перезагрузки программы в отладчике (Debug - Restart) все изменения сбрасываются.

### Фрейм дампа памяти

В данном фрейме также возможны установки точек останова на чтение-запись-выполнение (через правую кнопку мыши), перемещение по адресам и т.п. Через контекстное меню можно задавать любой требуемый для выполнения текущей задачи вид дампа памяти. Дамп может быть представлен в виде HEX кодировки, в виде ASCII и UNICODE строк, в виде Short, Long, Float чисел или же может быть показан PE-заголовок.

### Фрейм регистров процессора

В данном фрейме нам представлены все регистры процессора. В стандартном здесь находятся основные регистры, флаговый регистр, регистры FPU. Значения основных регистров (кроме EIP) могут быть изменены на любом этапе отладки. Для этого достаточно щелкнуть на нужном регистре правой кнопкой мыши, выбрать опцию Modify и в появившемся окне ввести необходимое значение. Значения регистров и флагов, которые в ходе выполнения последней машинной команды были модифицированы, подсвечиваются красным цветом.

### Фрейм стека

В данном фрейме предлагается работа со стеком и реализованы ранее описанные опции.

### **Порядок выполнения лабораторной работы:**

1. Установить программу WinZip 7.0 (из каталога `\DATA\WinZip_7_0\`).
2. Запустить программу WinZip и нажать кнопку «Enter Registration Code...», в результате чего откроется окно регистрации.
3. Попробуйте ввести произвольное имя и серийный номер программы. Какова реакция программы при этом (внести в отчет)?
4. Далее попытаемся проанализировать регистрационную защиту архиватора WinZip 7.0. При выполнении задания будем пользоваться точками прерывания по чтению информации, введенной пользователем в диалоговом окне регистрации (GetDlgItemTextA). Установите точку прерывания на вызов функции GetDlgItemTextA.
5. Выполнить процедуру регистрации WinZip.

6. По какому адресу вызывается функция `GetDlgItemTextA`? (команда `CALL`). Внесите адрес в отчет.
7. Какая информация (аргументы) передается в данную функцию, и чему равны значения данных аргументов? Внесите эту информацию в отчет.
8. В какой регистр записывается адрес хранения идентификатора пользователя? Внесите в отчет адрес и регистр.
9. Каково значение данного регистра при выходе из процедуры чтения информации? Вывести содержимое памяти по содержимому этого регистра на экран и удостовериться в правильности этой гипотезы.
10. Найти сразу же в программном коде (чуть ниже) место вызова аналогичной функции, предназначенной для чтения регистрационного номера. Каков адрес вызова данной функции? Внесите адрес в отчет.
11. В какой регистр записывается адрес хранения введенного регистрационного номера пользователя? Внесите адрес в отчет.
12. Каково значение данного регистра при выходе из процедуры чтения информации? Внести это значение в отчет. Вывести содержимое памяти по содержимому этого регистра на экран и удостовериться в правильности этой гипотезы.
13. В пошаговом режиме выполнить чтение идентификатора пользователя и регистрационного номера.
14. Что по вашему мнению выполняют команды по адресам `0167:00408049` и `0167:00408053` (с точки зрения `WinZip`)? Внесите эти сведения в отчет. Смоделировать на компьютере ситуацию, когда эти условия не выполняются.
15. Что по вашему мнению осуществляет процедура по адресу `00407B4B` (вызов `CALL 00407B4B` по адресу `0167:0040805C`). При ответе руководствоваться командами, следующими за данным вызовом. Внесите эти сведения в отчет. Осуществите жесткий взлом программы путем модификации регистра флагов до выполнения команды по адресу `0167:00408063`. Покажите результат

регистрации программы преподавателю. Запускается ли Winzip нормально после такой регистрации, удался ли жесткий взлом? Сделайте выводы.

16. Перейти к пошаговому исследованию процедуры по адресу 00407B4B (вызов CALL 00407B4B по адресу 0167:0040805C).

17. Что осуществляется с точки зрения WinZip командой по адресу 0167:00407B58? При ответе на вопрос руководствоваться предварительно полученной информацией (в том числе относительно адреса 47D928). Внесите эти сведения в отчет.

18. Какие аргументы передаются в процедуру (CALL 00407E31), вызываемую по адресу 0167:00407B83? Указать их значения и назначение в отчете.

19. Какая информация и в каких регистрах передается в процедуру (CALL 00407CC6), вызываемую по адресу 0167:00407C16? Внесите эти сведения в отчет. Проанализировать код, следующий за вызовом данной процедуры (адреса памяти 0167:00407C1B – 0167:00407C2A).

20. Какие аргументы подаются на вход процедуры (CALL 00457900), вызываемой по адресу 0167:00407C2A, каковы значения данных аргументов, что находится по соответствующим адресам? Сделать вывод о том, что делает данная процедура. Внесите эти сведения в отчет.

21. Можете ли вы сказать регистрационный код программы? Внесите его в отчет.

22. Зарегистрировать WinZip.

### **Контрольные вопросы:**

1. Перечислите основные угрозы программному продукту, защищенному от несанкционированного использования путем ввода ключевой информации?

2. Какие задачи решает злоумышленник в процессе реализации данных угроз программному продукту?

3. Что понимают под обратным проектированием?

4. Для чего служат отладчики?

5. Для чего служат мониторы событий?

6. Для чего служат дизассемблеры?



7. Для чего служат редакторы кода?
8. Какова классификация средств обратного проектирования?
9. Опишите приемы, используемые злоумышленником при отладке и дизассемблировании ПО.
10. Для каких целей используют перехват Win API с помощью средств отладки?
11. Какие API-функции может перехватывать злоумышленник с целью обнаружения ключевой информации?

### **4.3. Лабораторная работа № 3**

**Наименование лабораторной работы** – Применение дизассемблера IDA Pro.

**Время на выполнение** – 4 часа.

**Цель** – познакомиться на практике с методами статического исследования эффективности механизмов защиты.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками статического исследования эффективности механизмов защиты с применением дизассемблеров.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, интерактивный дизассемблер IDA Pro, редактор кода Niew.

#### **Краткие теоретические сведения:**

Одним из способов изучения программного обеспечения в условиях отсутствия исходных текстов является дизассемблирование – процесс перевода двоичных команд процессора в удобочитаемые мнемоники этих команд.

Качественное дизассемблирование объемного программного продукта является достаточно трудной задачей, так как ассемблирование представляет собой однонаправленный процесс с потерями. Полное автоматическое восстановление исходного текста программы, как правило, невозможно.

По типу реализации интерфейса взаимодействия с пользователем, существующие дизассемблеры можно разделить на две группы – автономные и интерактивные.

Автономные дизассемблеры требуют от пользователя всех указаний до начала процесса дизассемблирования и не позволяют вмешиваться непосредственно в сам процесс. Если же конечный результат окажется неудовлетворительным, то пользователь либо вручную правит полученный листинг, либо ука-

зывает дизассемблеру на его ошибки и повторяет всю процедуру вновь и вновь. Такой способ общения человека с дизассемблером непроизводителен и неудобен, но его легче запрограммировать.

Интерактивные дизассемблеры обладают развитым пользовательским интерфейсом, благодаря которому они приобретают значительную гибкость, позволяя человеку «вручную» управлять разбором программы, помогая автоматическому анализатору там, где ему самому не справиться – отличать адреса от констант, определять границы инструкций, области кода программы и данных и т.п.

Примерами автономных дизассемблеров является SOURCER и встроенный дизассемблер редактора машинных кодов HIEW, а интерактивного – IDA Pro (рисунок 4.2)

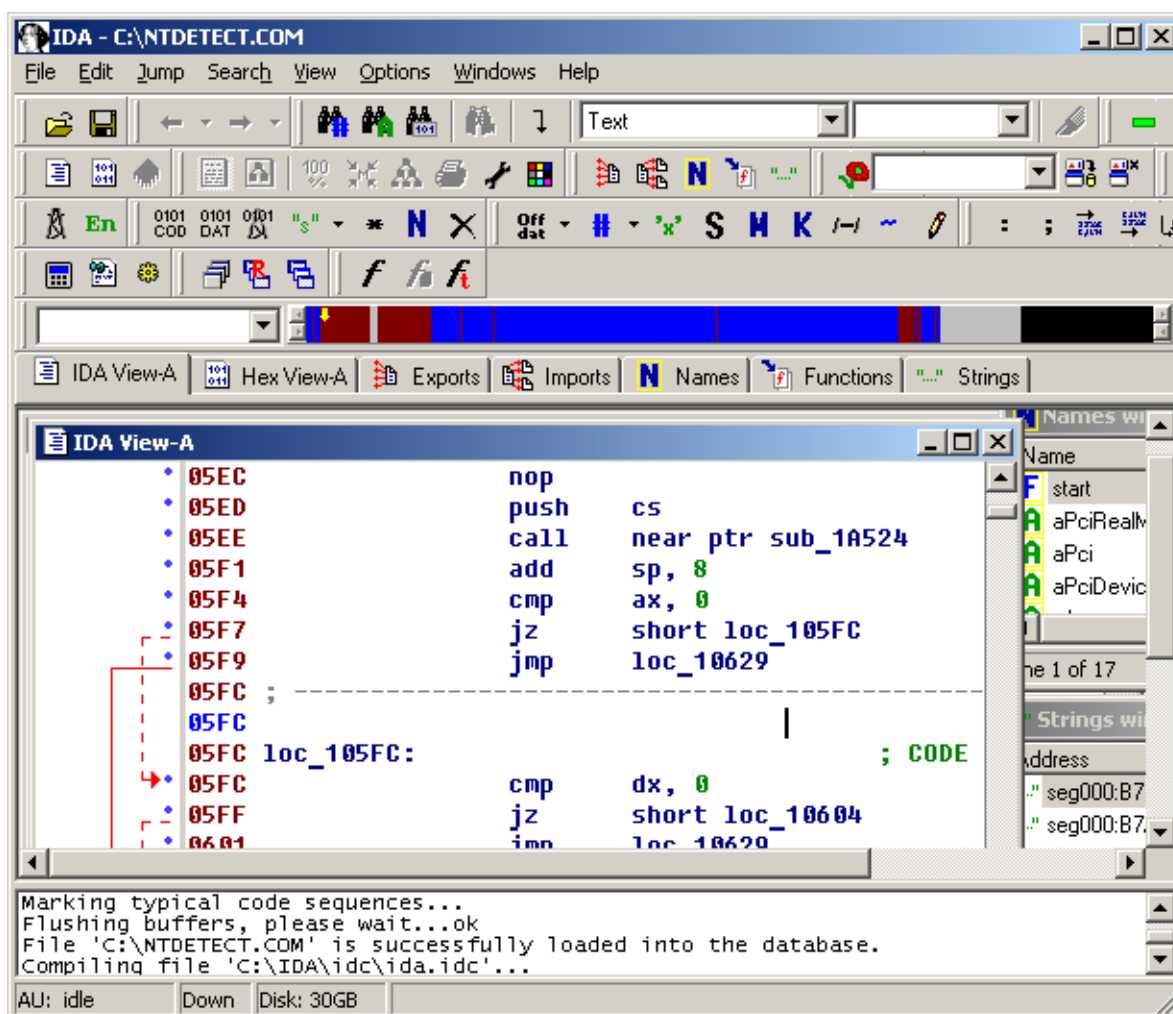


Рис. 4.2. Интерфейс интерактивного дизассемблера IDA Pro

Интерактивный дизассемблер IDA Pro предназначен для дизассемблирования кода программы в мнемонические инструкции на языке ассемблера. Данный дизассемблер изначально проектировался как интерактивная среда, предназначенная для совместной работы с человеком. Преимуществом интерактивных дизассемблеров является то, что их работу сложнее нарушить каким-либо хитрым приемом, что возможно для дизассемблеров, работающих в автоматическом режиме.

IDA Pro поддерживает инструкции различных процессоров. Среди них все версии процессоров INTEL, Motorola, Z80 и многих других.

В IDA Pro возможно непосредственно в процессе интерактивного диалога указывать, является ли код программы по заданному адресу действительно кодом либо данными, т.к. точно это понять способен лишь человек.

Автономные дизассемблеры плохо справляются с анализом больших файлов, а с зашифрованным кодом не справляются вообще. Применение интерактивных дизассемблеров позволяет решить эту проблему.

В IDA Pro реализован удобный C-подобный внутренний язык написания скриптов (IDC), позволяющий реализовать многие полезные действия (например, осуществлять дешифрование зашифрованного участка кода).

Для написания скриптов дешифрования могут, например, понадобиться следующие команды.

**PatchByte([Сегмент, Смещение], Значение)** – заменить данные, находящиеся по адресу [Сегмент, Смещение] на Значение.

**Byte([Сегмент, Смещение])** – получить значение по адресу [Сегмент, Смещение].

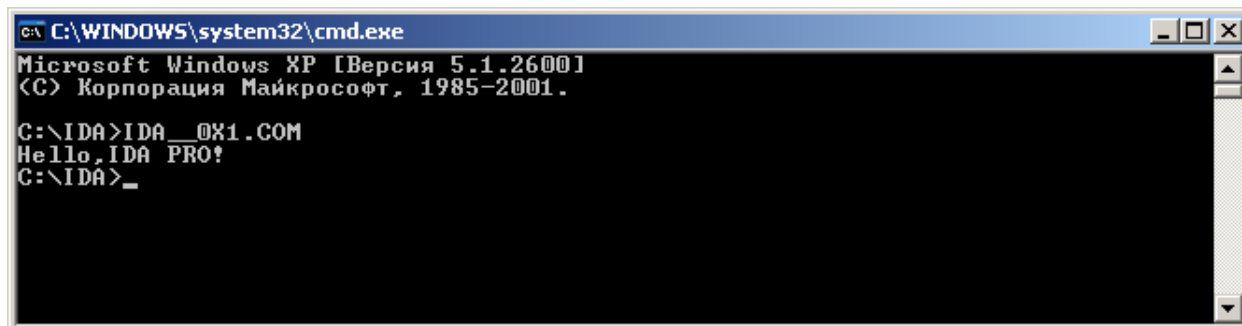
**^** – обозначение функции XOR (побитовое сложение по модулю 2).

**for(Начальное значение; Условие; Шаг)** – цикл, изменяющий значение счетчика цикла от Начального значения на величину Шага до тех пор, пока выполняется Условие.

**auto a** – определение счетчика цикла с именем a.

## Порядок выполнения лабораторной работы:

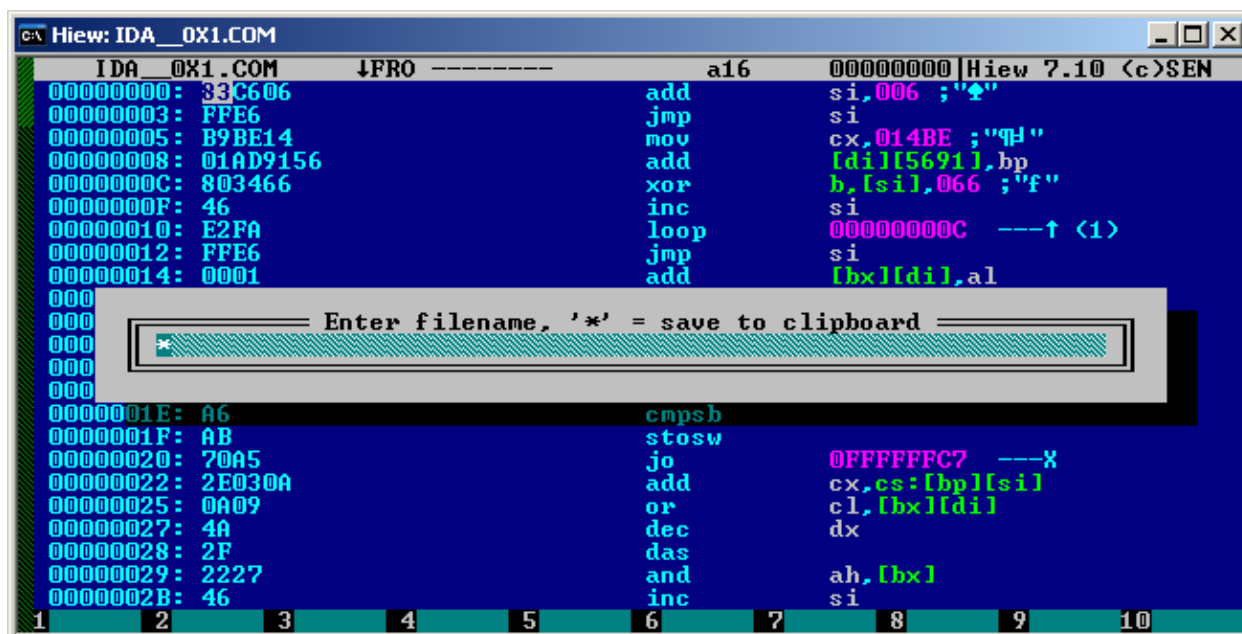
1. В консольном режиме (ПУСК>ВЫПОЛНИТЬ>cmd) запустить программу IDA\_\_0X1.COM и внешне изучить механизм ее работы. Внести описание работы программы в отчет.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

C:\IDA>IDA__0X1.COM
Hello, IDA PRO!
C:\IDA>_
```

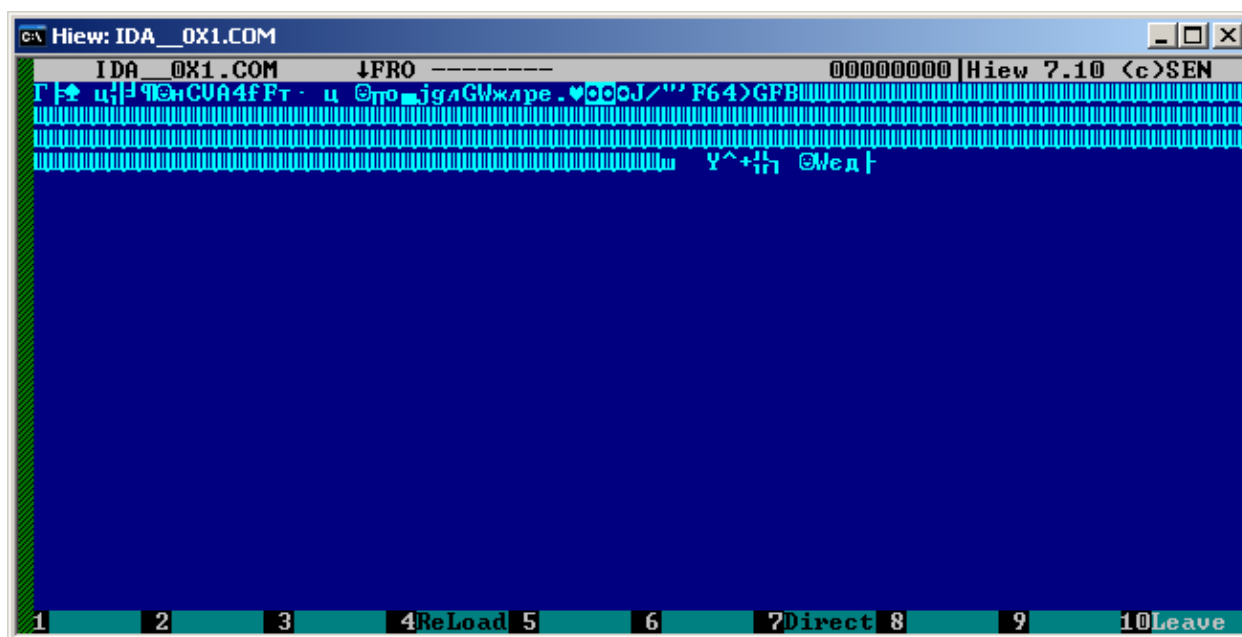
2. Далее проведем предварительное исследование кода программы. Запустите редактор кода Hiew. Загрузите в него программу IDA\_\_0X1.COM. Режимы отображения данных в Hiew переключаются нажатием клавиши **Enter**, либо **F4**
3. Внесите в отчет дизассемблированный код программы. Для этого в режиме дизассемблера (F4>Decode) можете воспользоваться комбинацией клавиш **ALT+P**, для сохранения текущего окна в файл (для этого укажите выходной файл), либо в буфер обмена (для этого укажите вместо имени файла знак «\*»).



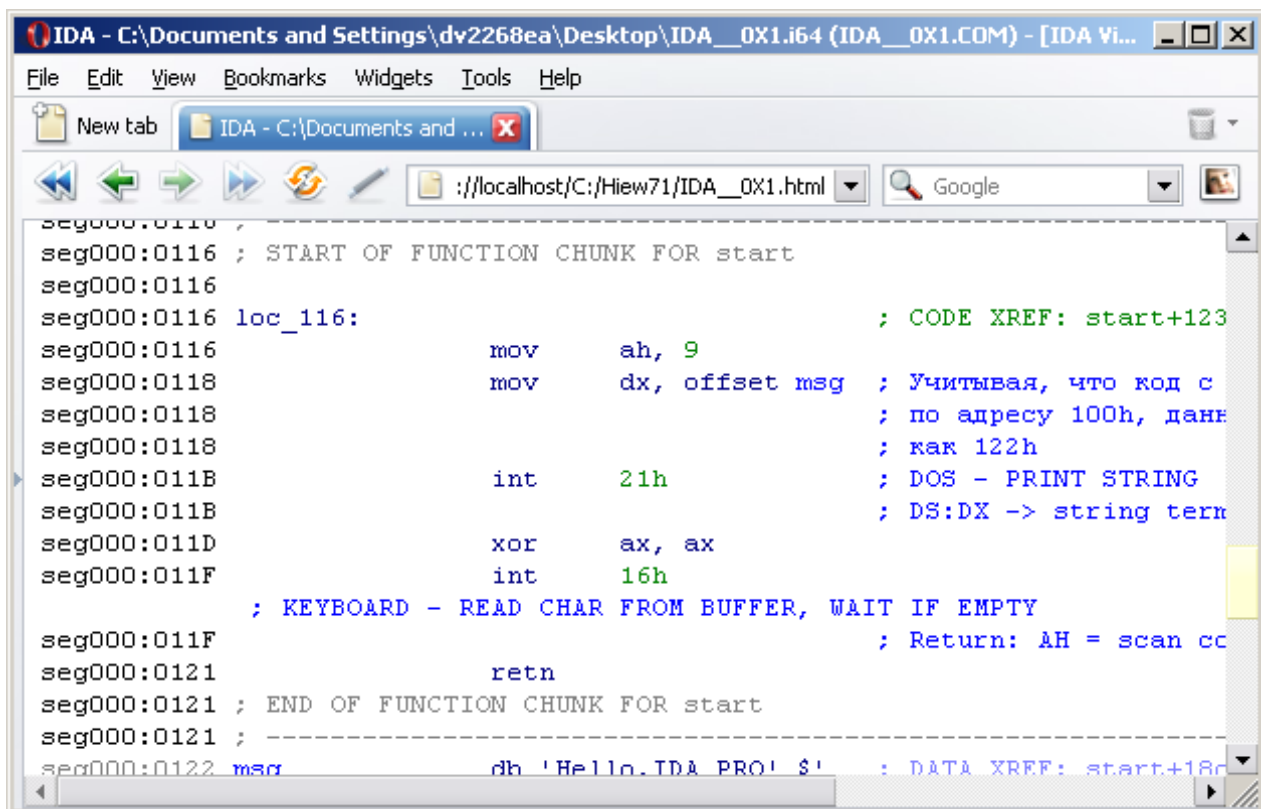
```
Hiew: IDA__0X1.COM
IDA__0X1.COM  ↓FRO -----  a16  00000000 Hiew 7.10 (c)SEN
00000000: 33C606      add     si, 0006 ; "A"
00000003: FFE6        jmp     si
00000005: B9BE14      mov     cx, 014BE ; "qH"
00000008: 01AD9156    add     [di], [5691], bp
0000000C: 803466      xor     b, [si], 066 ; "f"
0000000F: 46          inc     si
00000010: E2FA        loop   0000000C ---↑ (1)
00000012: FFE6        jmp     si
00000014: 0001        add     [bx][di], al
000
000
000
000
000
0000001E: A6          cmpsb
0000001F: AB          stosw
00000020: 70A5        jo     0FFFFFFC7 ---X
00000022: 2E030A      add     cx, cs:[bp][si]
00000025: 0A09        or      cl, [bx][di]
00000027: 4A          dec     dx
00000028: 2F          das
00000029: 2227        and     ah, [bx]
0000002B: 46          inc     si
1      2      3      4      5      6      7      8      9      10
```

Enter filename, '\*' = save to clipboard

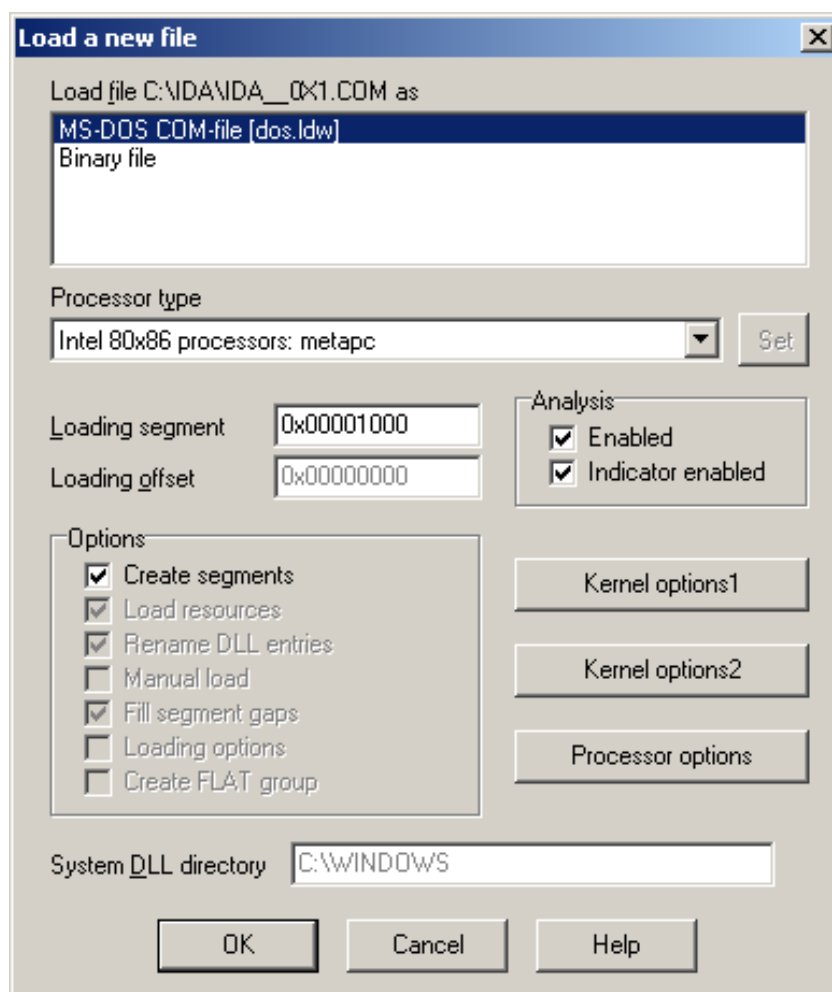
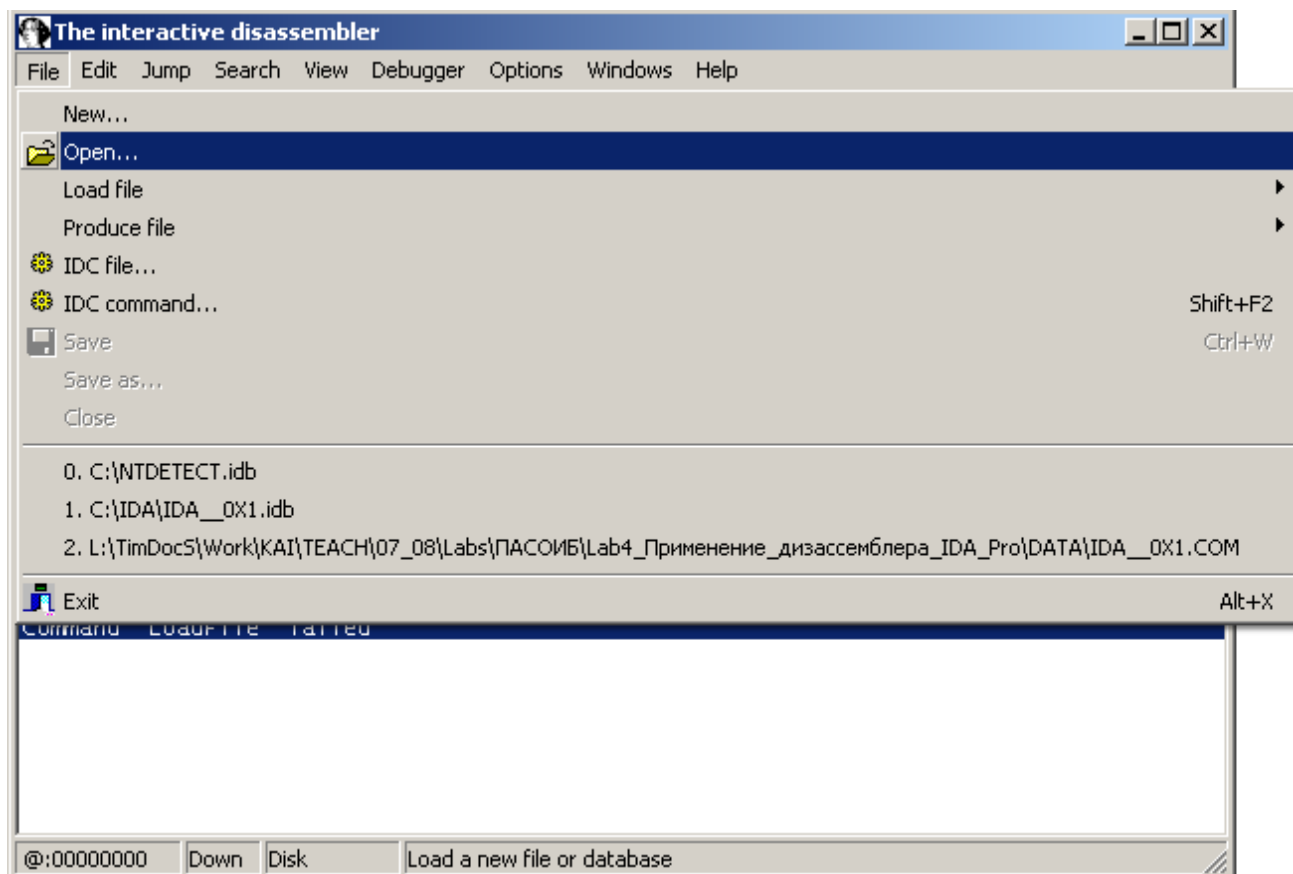
4. Исследовать код программы в режиме **Text** (F4>Text). Удалось ли вам обнаружить текстовую строку, выводимую программой на экран, среди кода? Какой можно сделать вывод? Внесите эту информацию в отчет.



5. Предварительное изучение завершено. Далее необходим более детальный анализ кода программы, с целью обнаружения механизма защиты, скрывающего от исследователя текстовую строку, выводимую программой на экран. Для этого нужно привести код к виду, более удобному для анализа. Откройте и внимательно изучите файл, приложенный к лабораторной работе (каталог `\DATA\`): `IDA__0X1.html`. К этому виду код был приведен при помощи интерактивного дизассемблера IDA Pro.

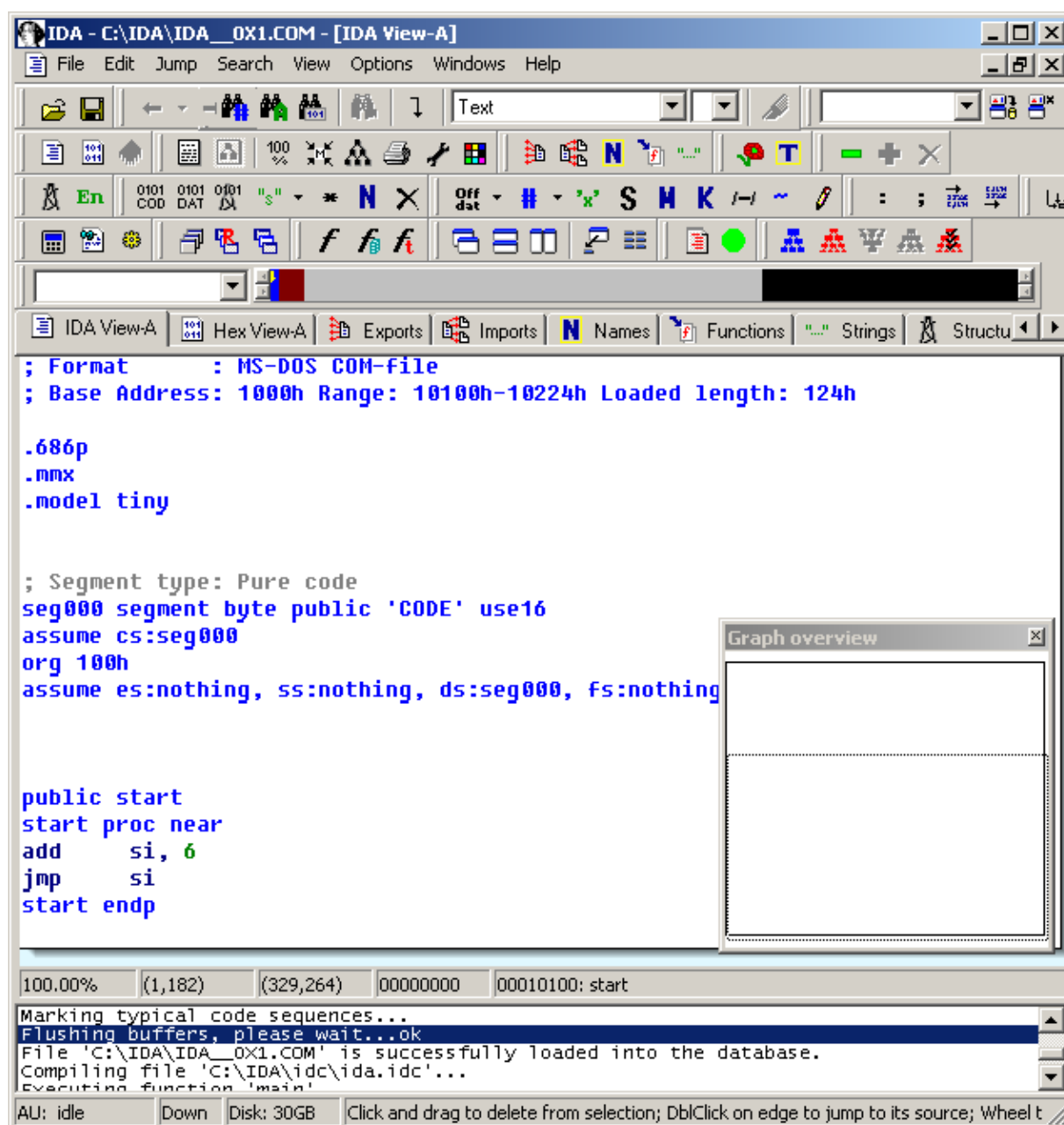


6. Запустить интерактивный дизассемблер IDA Pro в режиме графического интерфейса (файл idag.exe). Загрузить в IDA Pro (FILE>OPEN>Ida\_\_0x1.com) файл IDA\_\_0X1.COM и дизассемблировать его как DOS COM-файл. Для этого, после открытия окна «Open...» выберете режим «MS-DOS COM file». Укажите также загрузочный сегмент (Loading Segment) равный 0x00001000.





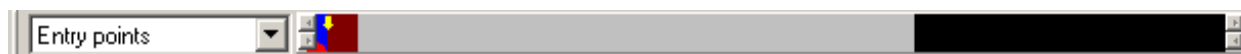
7. После загрузки и предварительного анализа, выполненного IDA Pro автоматически, у вас должен появиться следующий код:



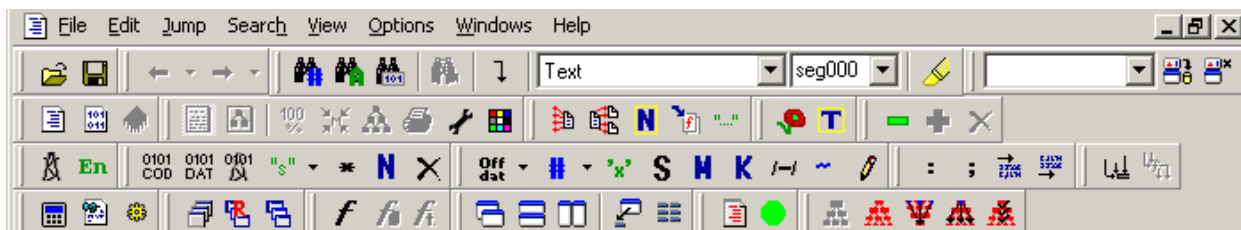
8. Изучите интерфейс IDA. Исследуйте каждую информационную вкладку и возможности, предоставляемые IDA для анализа загруженной программы. Щелкните по каждой вкладке:



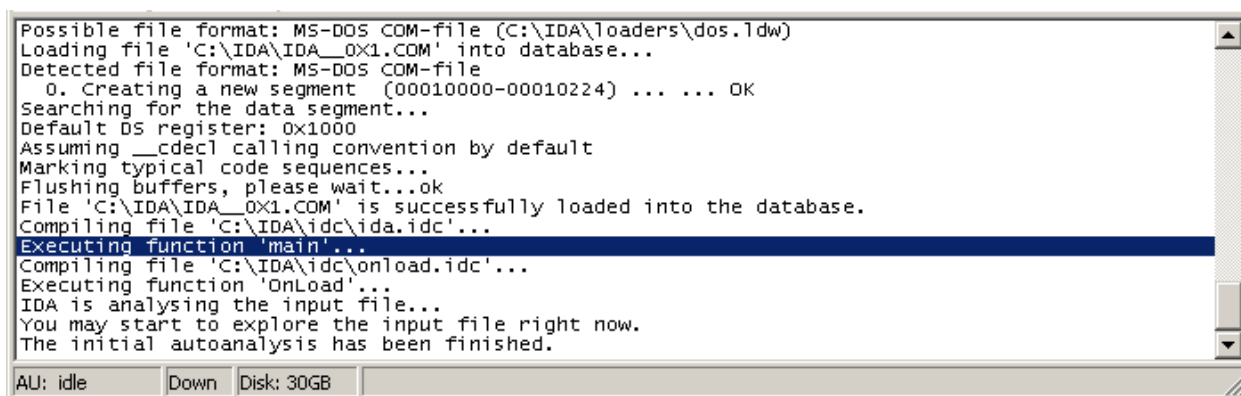
Графическая панель предоставляет возможность быстро перемещаться по коду вашей программы. Синяя область – область верно распознанного и проанализированного кода, коричневая – область неуверенно распознанного кода, серая – область нераспознанного кода.



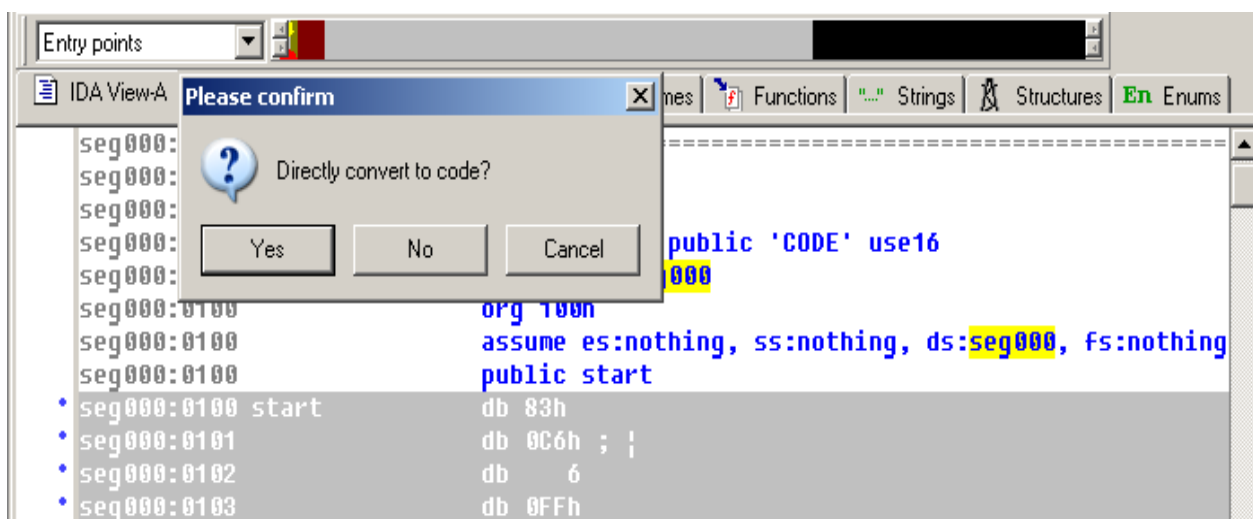
Панели управления позволяют получить быстрый доступ ко всем функциям IDA (подробно об их назначении вы можете прочитать в системе помощи IDA либо приложенных к лабораторной справочниках):



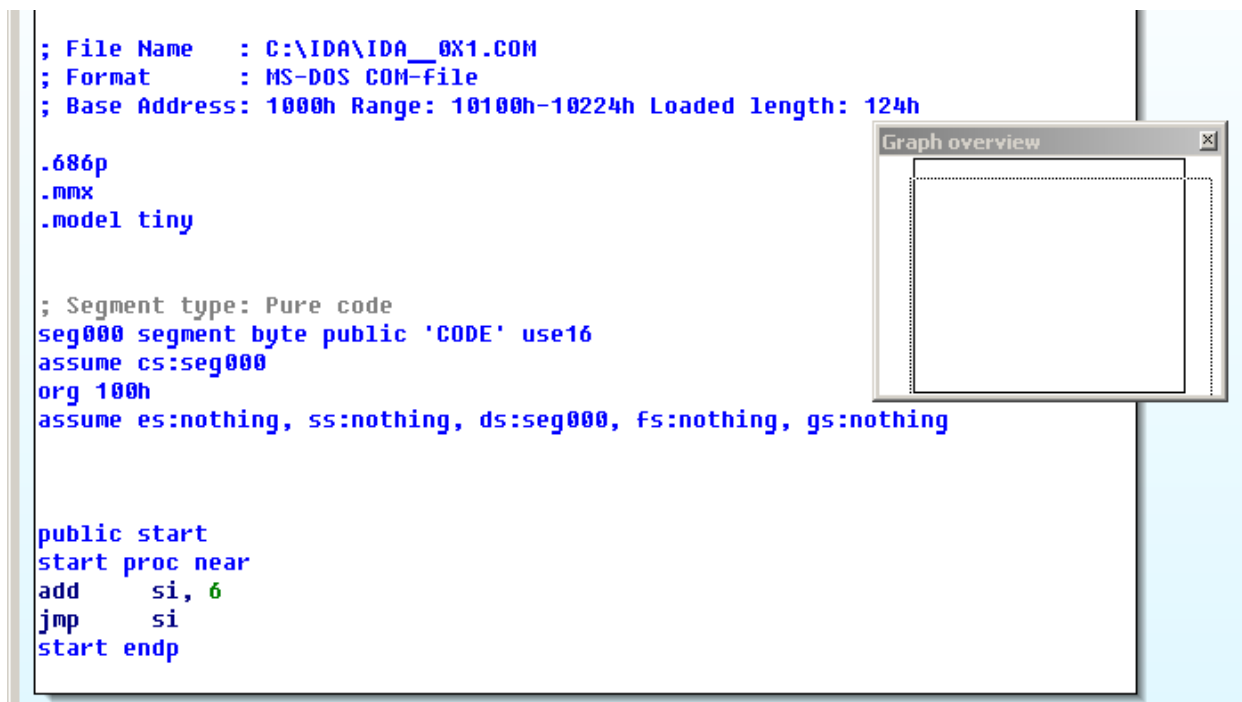
Самая нижняя – строка состояния – отображает информацию о ходе работы по дизассемблированию программы:



9. Если Ida приняла код за данные, не дизассемблировав его, то нужно указать ей, что данные являются командами. Для переключения отображения кода в виде команд либо данных, выделяйте соответствующий код и нажимайте C (команда), либо D (данные). Тоже самое можно проделывать при помощи контекстного меню.



10.Переключитесь на вкладку IDA View-A. Как видим, IDA верно распознала лишь первые команды и остановилась, ожидая подсказок от пользователя. По умолчанию, если вы переместитесь с помощью графической панели на область распознанного кода, IDA построит для вас граф передачи управления алгоритма. Пока в нее входят лишь самые первые команды и первый блок.



11.Переместитесь при помощи графической панели в область неуверенно распознанного кода.



У вас отобразится примерно следующее.

The screenshot shows the IDA Pro disassembler interface. The 'Entry points' window is open at the top. The main window displays assembly code for a subroutine named 'start' located at address 00010100. The code includes instructions such as 'add si, 6', 'jmp si', 'mov cx, 14BEh', 'add [di+5691h], bp', 'xor byte ptr [si], 66h', 'inc si', 'loop loc\_1010C', and 'jmp si'. A cross-reference is shown for 'seg000:0110' pointing to 'loc\_1010C'. The code ends with 'end start'.

```

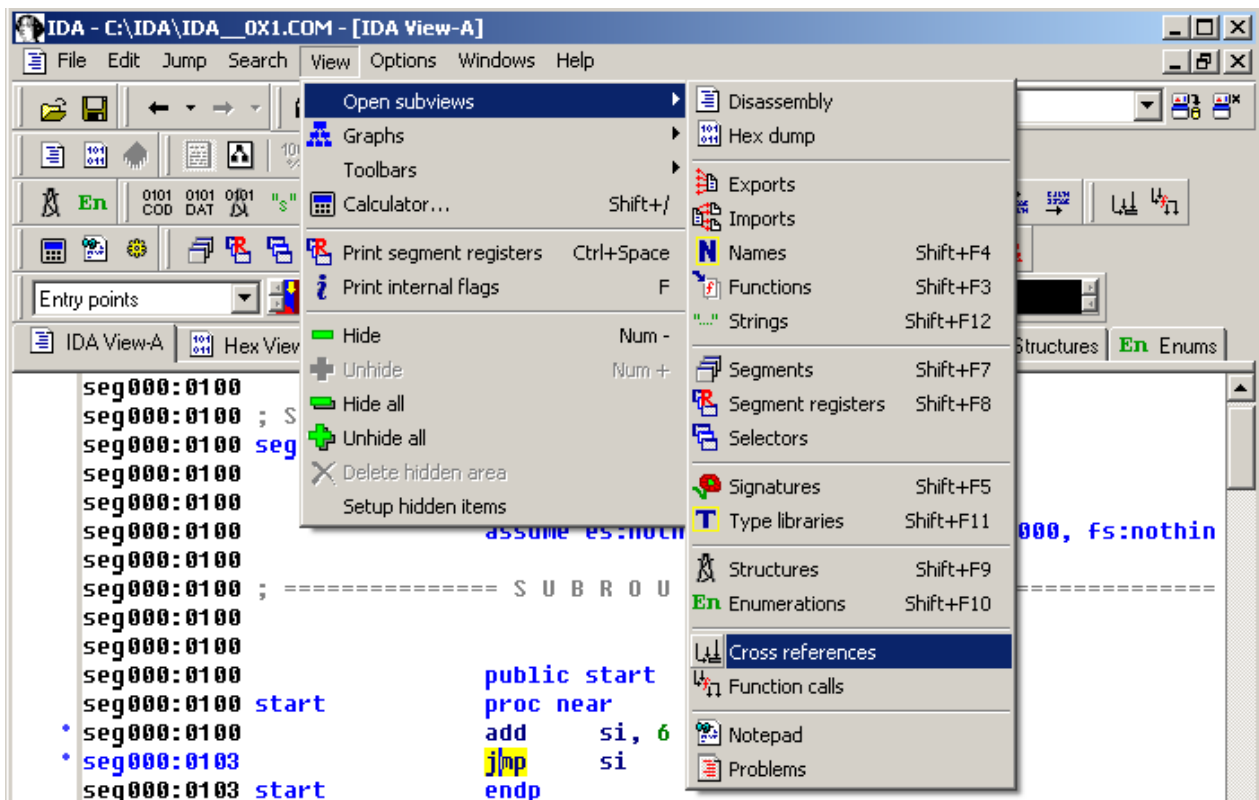
seg000:0100
seg000:0100 ; ===== S U B R O U T I N E =====
seg000:0100
seg000:0100 public start
seg000:0100 start proc near
* seg000:0100 add si, 6
* seg000:0103 jmp si
seg000:0103 start endp
seg000:0103
seg000:0105 ; -----
* seg000:0105 mov cx, 14BEh
* seg000:0108 add [di+5691h], bp
seg000:010C loc_1010C: ; CODE XREF: seg000:0110↓j
* seg000:010C xor byte ptr [si], 66h
* seg000:010F inc si
* seg000:0110 loop loc_1010C
* seg000:0112 jmp si
seg000:0112 ; -----
* seg000:0114 db 0, 1, 0D2h, 6Fh, 0DCBh, 6Ah, 67h, 0ABh, 47h, 57h,
seg000:0114 db 0ABh, 70h, 0A5h, 2Eh, 3, 2 dup(0Ah), 9, 4Ah, 2Fh
seg000:0114 db 27h, 46h, 36h, 34h, 29h, 47h, 46h, 42h, 0E4h dup
seg000:0114 db 0E8h, 2 dup(0), 59h, 5Eh, 2Bh, 0CEh, 0BFh, 0, 1,
seg000:0114 db 0F3h, 0A4h, 0C3h
seg000:0114 seg000 ends
seg000:0114
seg000:0114
seg000:0114
seg000:0114 end start

```

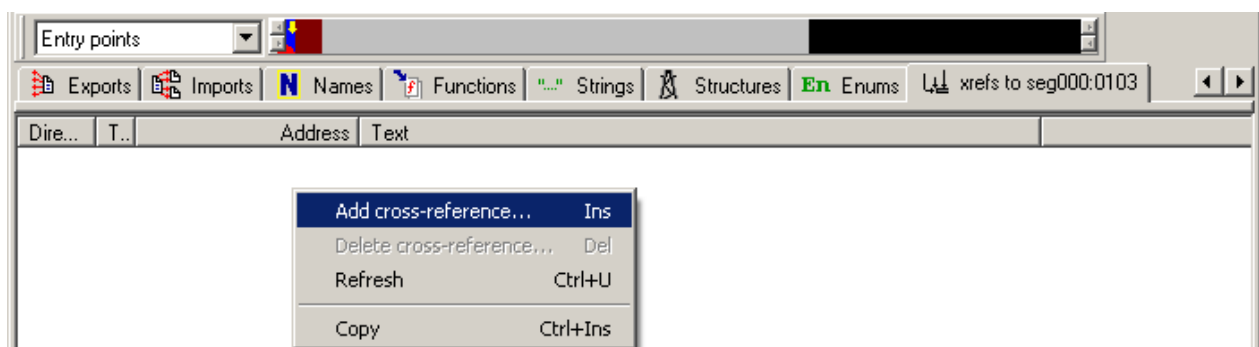
Это и есть весь дизассемблированный код, с которым самостоятельно смогла разобраться IDA. Как видим, многие данные еще не проанализированы.

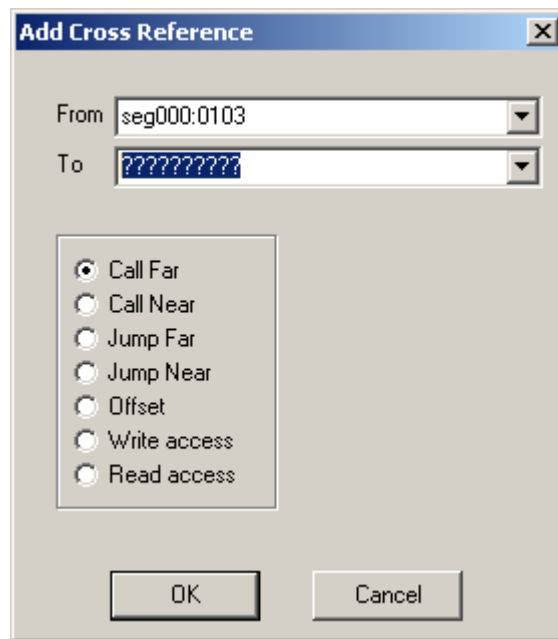
12. Здесь и далее вам придется использовать справочники по языку ассемблера и прерываниям, а также файл, который вы изучили выше, с уже проанализированным кодом. Внесите в отчет листинг команд, находящихся по адресам seg000:0100-seg000:0103. Прокомментируйте их. Для чего они нужны?

13. По какому адресу выполняется переход с адреса seg000:0103? Сообщите об этом IDA. Для этого перейдите к окну перекрестных ссылок (VIEW>OPEN SUBVIEWS>CROSS REFERENCES, либо щелчком на кнопке с соответствующим изображением).



С помощью контекстного меню, либо клавишей **Ins**, добавьте перекрестную ссылку на определенный вами адрес.





14. После выполнения предыдущего пункта IDA должна была автоматически проанализировать дальнейший код.

```

Entry points
IDA View-A  Hex View-A  Exports  Imports  Names  Functions  Strings  Structures  Enu
seg000:0100      assume es:nothing, ss:nothing, ds:seg000, fs:nothi
seg000:0100      ; ===== S U B R O U T I N E =====
seg000:0100
seg000:0100      public start
seg000:0100      proc near
seg000:0100 start
seg000:0100      add     si, 6
seg000:0103      jmp     si
seg000:0103 start
seg000:0103      endp
seg000:0103      ; -----
seg000:0105      db 0B9h
seg000:0106      ; -----
seg000:0106      mov     si, 114h      ; CODE XREF: start+3↑p
seg000:0109      lodsw
seg000:010A      xchg    ax, cx
seg000:010B      push    si
seg000:010C      loc_1010C:
seg000:010C      xor     byte ptr [si], 66h      ; CODE XREF: seg000:0110↓j
seg000:010F      inc     si
seg000:0110      loop    loc_1010C
seg000:0112      jmp     si
seg000:0112      ; -----

```

Если вы не получили аналогичного кода и IDA восприняла код по адресу seg000:0105 как команды,

```

seg000:0105      ; -----
seg000:0105      mov     cx, 148Eh      ; CODE XREF: start+3↑p
seg000:0108      add     [di+5691h], bp

```

то поставьте курсор на адрес seg000:0105 и, нажав кнопку **D**, укажите, что это данные (db).

```

seg000:0103 ; -----
• seg000:0105      db  0B9h
• seg000:0106      db  0BEh ; -           ; CODE XREF: start+31P
• seg000:0107      db  14h
seg000:0108 ; -----

```

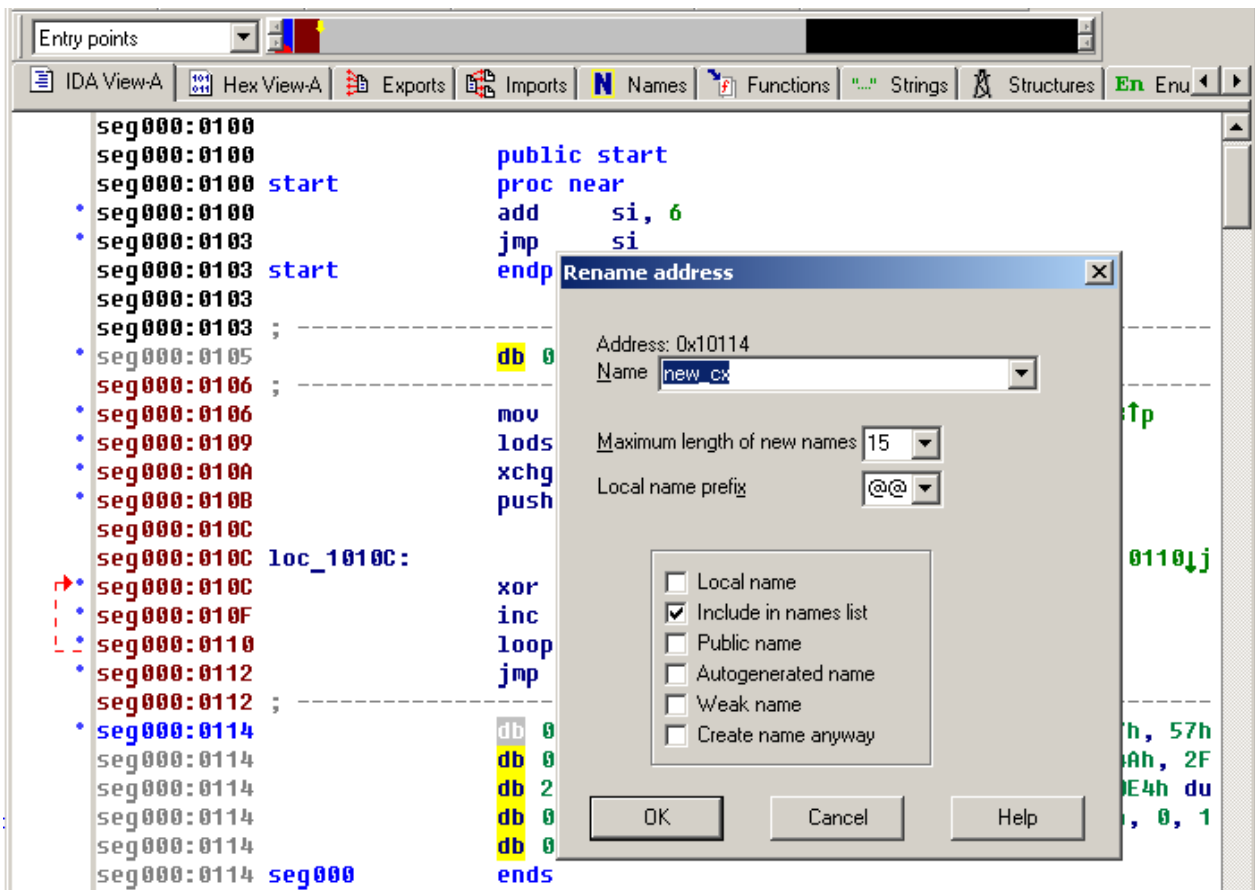
Затем поставьте курсор на адрес seg000:0106 и укажите, что этот код является командой. Для этого нажмите клавишу **C** и вы получите то, что требуется.

```

seg000:0103 ; -----
• seg000:0105      db  0B9h
seg000:0106 ; -----
• seg000:0106      mov     si, 114h
• seg000:0107      lodsw
• seg000:0108      xchg    ax, cx
• seg000:0109      push    si
seg000:010A
seg000:010B
seg000:010C

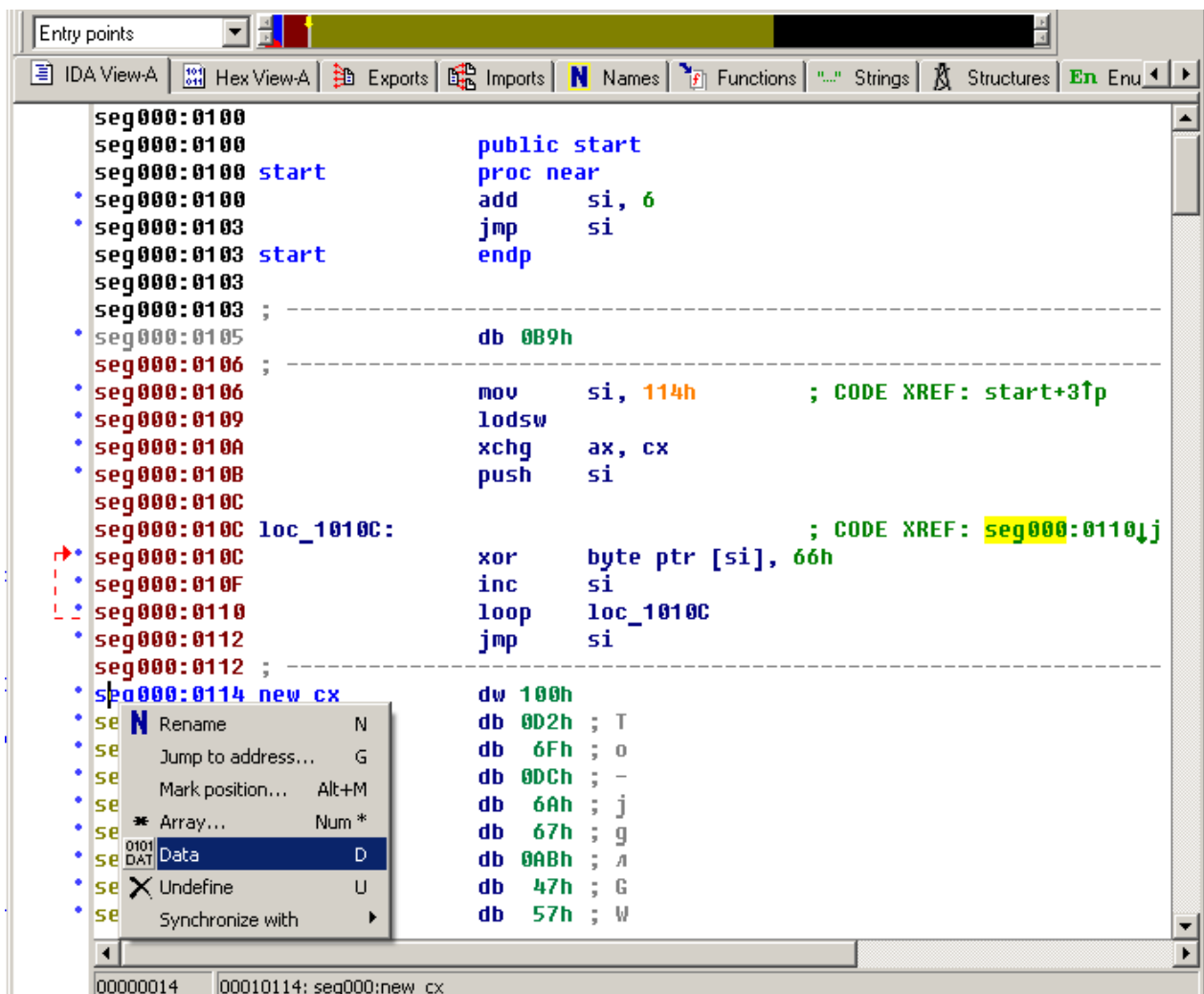
```

15. Произвести анализ кода по адресам seg000:0106-seg000:0112. Каков смысл данного участка кода? В чем его назначение? Что делают данные команды? Внесите эту информацию в отчет.
16. Что делает цикл, расположенный по адресам seg000:010C-seg000:0110? Сколько раз он повторяется? Внесите эту информацию в отчет.
17. Какое значение лежит в SI перед и после выполнения цикла? На что указывает SI?
18. Что представляет собой область кода, расположенного по адресу seg000:0114 и далее?
19. Дайте слову, расположенному по адресу seg000:0114 осмысленное имя (нажав клавишу **N**).

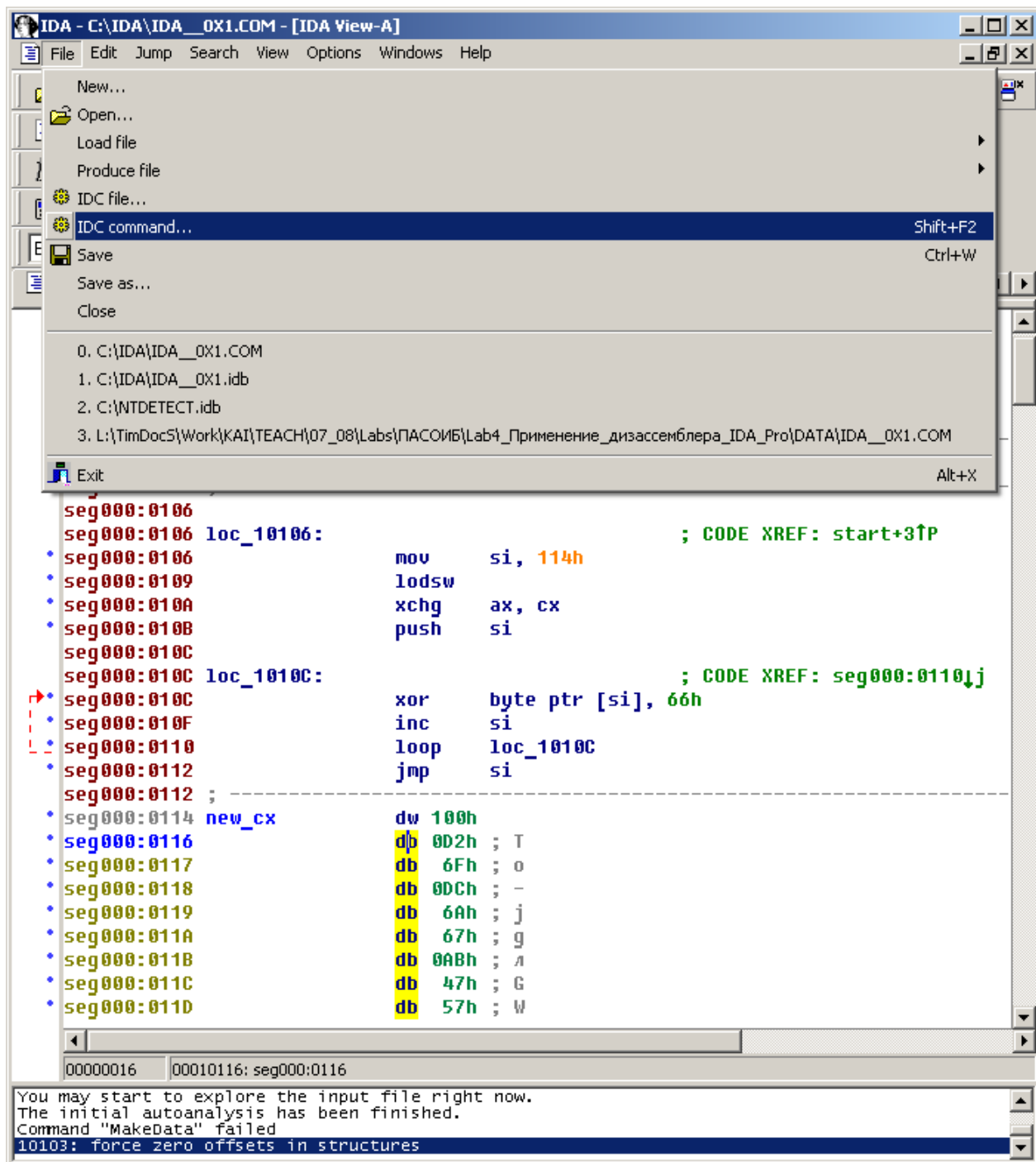


20. Нажимая клавишу **D** (либо выбирая соответствующий пункт контекстного меню), добейтесь преобразования данных по адресу seg000:0114 в двойное слово (dw).





21. Рассчитайте, по какому адресу осуществляется переход с адреса seg000:0112, после выполнения цикла. Внесите этот адрес в отчет. Добавьте перекрестную ссылку, указав дизассемблеру, куда осуществляется переход с адреса seg000:0112. Исследуйте код программы, по которому делается переход. Что делает этот блок команд? Подробно опишите, что делает каждая команда. Какое значение окажется в регистре CX? Какое значение окажется в регистре SI? Куда произойдет переход после команды RETN и почему? Внесите эту информацию в отчет.
22. Итак, мы практически привели код к виду, приведенному в файле *IDA\_0X1.html*. Осталось лишь произвести расшифровку зашифрованного участка кода, выяснить его функции и выявить сообщение «Hello, IDA PRO!». Для расшифровки кода в Ida Pro можно написать скрипт. Воспользуйтесь командой FILE>IDC COMMAND, либо комбинацией клавиш **Shift+F2**, для открытия окна ввода скриптов.

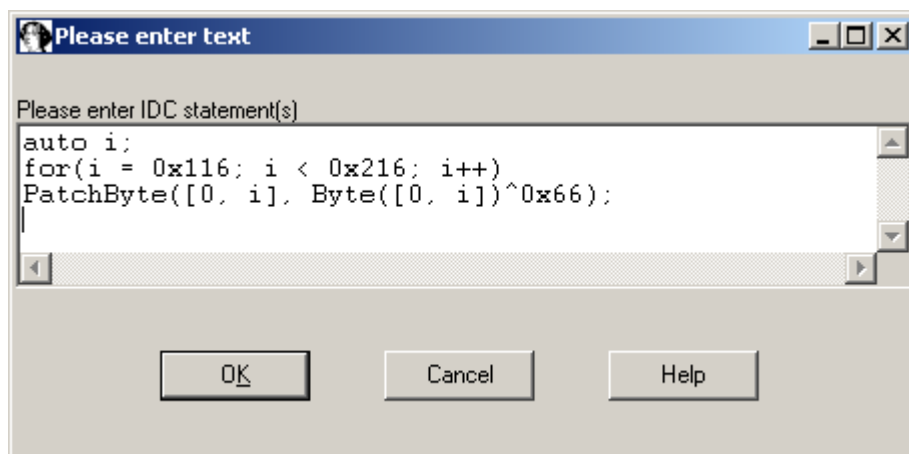


23. Введите скрипт для расшифровки кода программы по адресам seg000:0116-seg000:0215 и нажмите кнопку ОК.

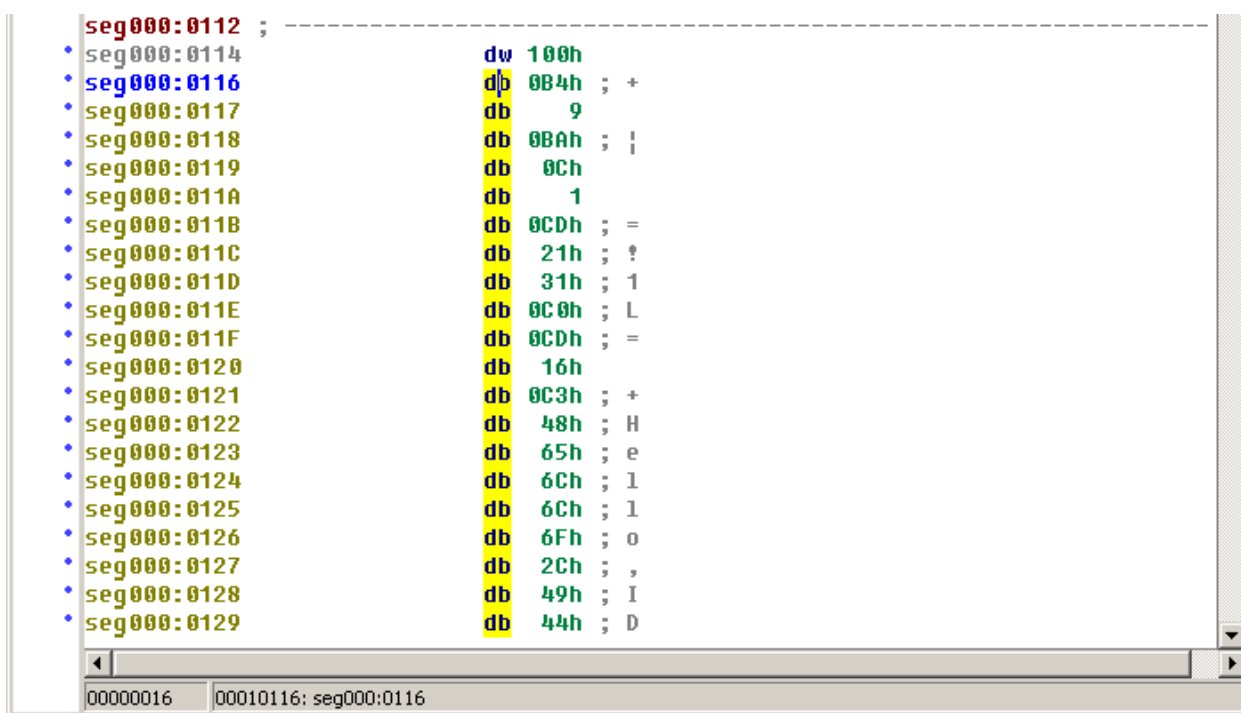
auto i;

for(i=0x116; i<0x216; i++)

PatchByte([0x1000, i], Byte([0x1000, i]^0x66));



24. Произошло ли дешифрование кода? Если да, то Вы должны были получить примерно следующий код:



25. Поясните работу скрипта. Что он делает? На каком ключе происходит дешифрование кода?

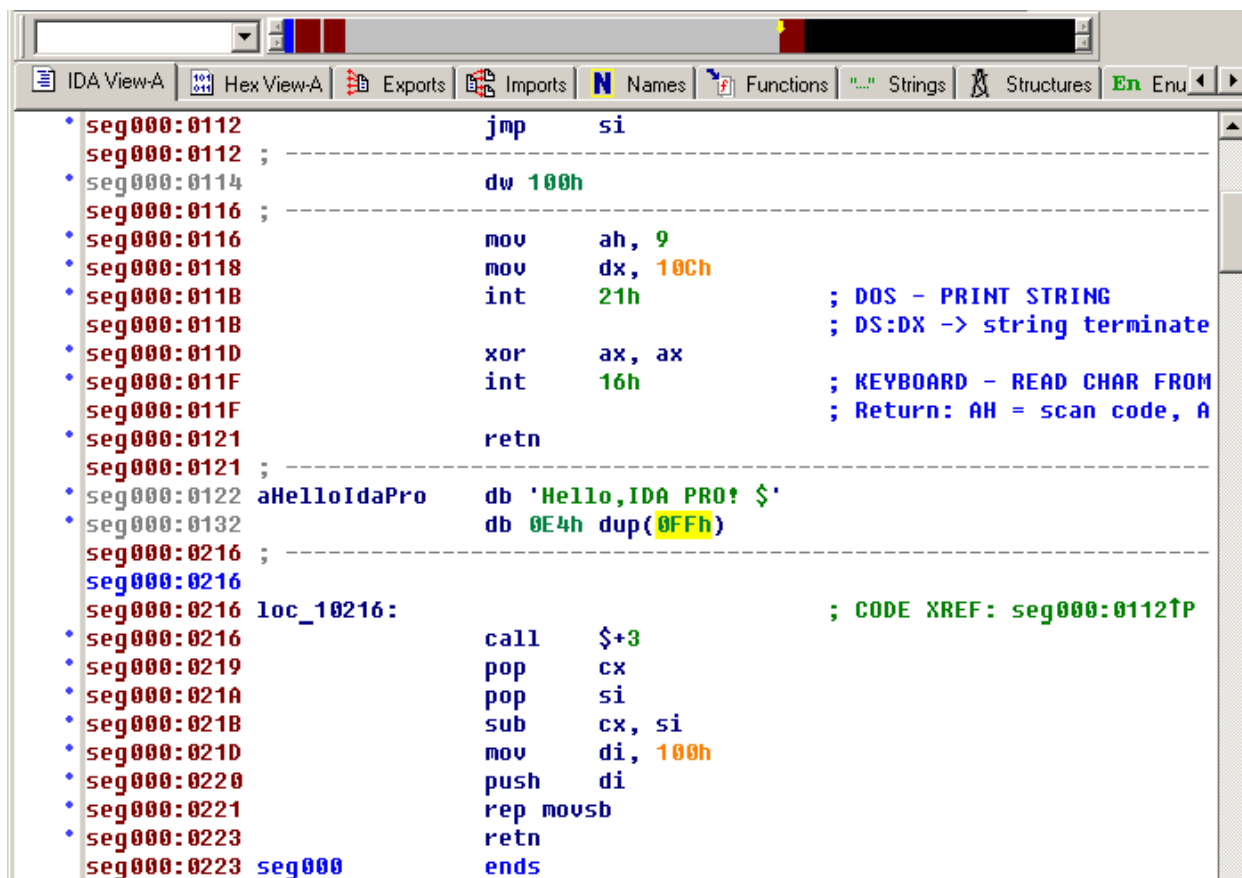
26. Перейдите по адресу seg000:0116 и укажите дизассемблеру, что этот участок является кодом. Для этого нажмите клавишу C.

```

* seg000:0116      mov     ah, 9
* seg000:0118      mov     dx, 10Ch
* seg000:011B      int     21h                ; DOS - PRINT STRING
* seg000:011B      ; DS:DX -> string terminate
* seg000:011D      xor     ax, ax
* seg000:011F      int     16h                ; KEYBOARD - READ CHAR FROM
* seg000:011F      ; Return: AH = scan code, A
* seg000:0121      retn
* seg000:0121      ; -----
* seg000:0122      db      48h ; H
* seg000:0123      db      65h ; e
* seg000:0124      db      6Ch ; l
* seg000:0125      db      6Ch ; l
* seg000:0126      db      6Fh ; o

```

27. Осталось лишь, для более удобного представления данных, представить фразу «Hello,IDA PRO!» одной строкой. Для этого выделите строки с соответствующими символами и нажмите клавишу **A** (ASCII представление данных). Перейдите по адресу seg000:0132 и укажите дизассемблеру, что этот код является массивом данных (для дальнейшего исследования они будут не нужны). Для этого нажмите клавишу **\*** на цифровой клавиатуре и в появившемся окне нажмите кнопку **ОК**. Теперь Вы получили итоговый вид кода программы.



```

* seg000:0112      jmp     si
* seg000:0112      ; -----
* seg000:0114      dw      100h
* seg000:0116      ; -----
* seg000:0116      mov     ah, 9
* seg000:0118      mov     dx, 10Ch
* seg000:011B      int     21h                ; DOS - PRINT STRING
* seg000:011B      ; DS:DX -> string terminate
* seg000:011D      xor     ax, ax
* seg000:011F      int     16h                ; KEYBOARD - READ CHAR FROM
* seg000:011F      ; Return: AH = scan code, A
* seg000:0121      retn
* seg000:0121      ; -----
* seg000:0122      aHelloIdaPro  db  'Hello,IDA PRO!'
* seg000:0132      db      0E4h dup(0FFh)
* seg000:0216      ; -----
* seg000:0216      loc_10216:                ; CODE XREF: seg000:0112↑P
* seg000:0216      call    $+3
* seg000:0219      pop     cx
* seg000:021A      pop     si
* seg000:021B      sub     cx, si
* seg000:021D      mov     di, 100h
* seg000:0220      push    di
* seg000:0221      rep     movsb
* seg000:0223      retn
* seg000:0223      seg000      ends

```

28. Перейти на расшифрованный участок кода и проанализировать его. Внести листинг расшифрованного кода с комментариями каждой ко-

манды в отчет. Какие действия он выполняет?

29. По каким адресам располагается фраза «Hello, IDA PRO!»? Внесите эти адреса в отчет.

30. Обобщите все данные, полученные в результате анализа программы. Внесите в отчет пошаговое описание алгоритма работы программы, что и как она делает, из каких блоков состоит. Опишите механизм защиты.

### **Контрольные вопросы:**

1. Что понимают под обратным проектированием?
2. Для чего служат отладчики?
3. Для чего служат мониторы событий?
4. Для чего служат дизассемблеры?
5. Для чего служат редакторы кода?
6. Какова классификация средств обратного проектирования?
7. Опишите приемы защиты от отладки и дизассемблирования ПО.
8. Почему шифрование кода программы считается наиболее предпочтительным способом защиты от отладчиков и дизассемблеров?
10. Что такое ассемблирование и дизассемблирование?
11. Какие выделяют группы дизассемблеров по типу взаимодействия с пользователем?
12. Охарактеризуйте возможности интерактивного дизассемблера IDA Pro.

#### **4.4. Лабораторная работа № 4**

**Наименование лабораторной работы** – Применение редактора кода Hiew совместно с отладчиком Turbo Debugger.

**Время на выполнение** – 4 часа.

**Цель** – познакомиться на практике с совместным использованием методов статического и динамического исследования эффективности механизмов защиты.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками статического и динамического исследования эффективности механизмов защиты.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, редактор кода Hiew, отладчик Turbo Debugger, программы для исследования: Act-1.com, Crackme.com.

##### **Краткие теоретические сведения:**

###### Редактор кода HIEW

Редактор кода Hiew представляет собой простой, но достаточно мощный инструмент, предназначенный для статического анализа и редактирования программ непосредственно в их исполняемом коде.

Hiew совмещает как функции hex-редактора, так и функции дизассемблера. Он содержит следующие компоненты:

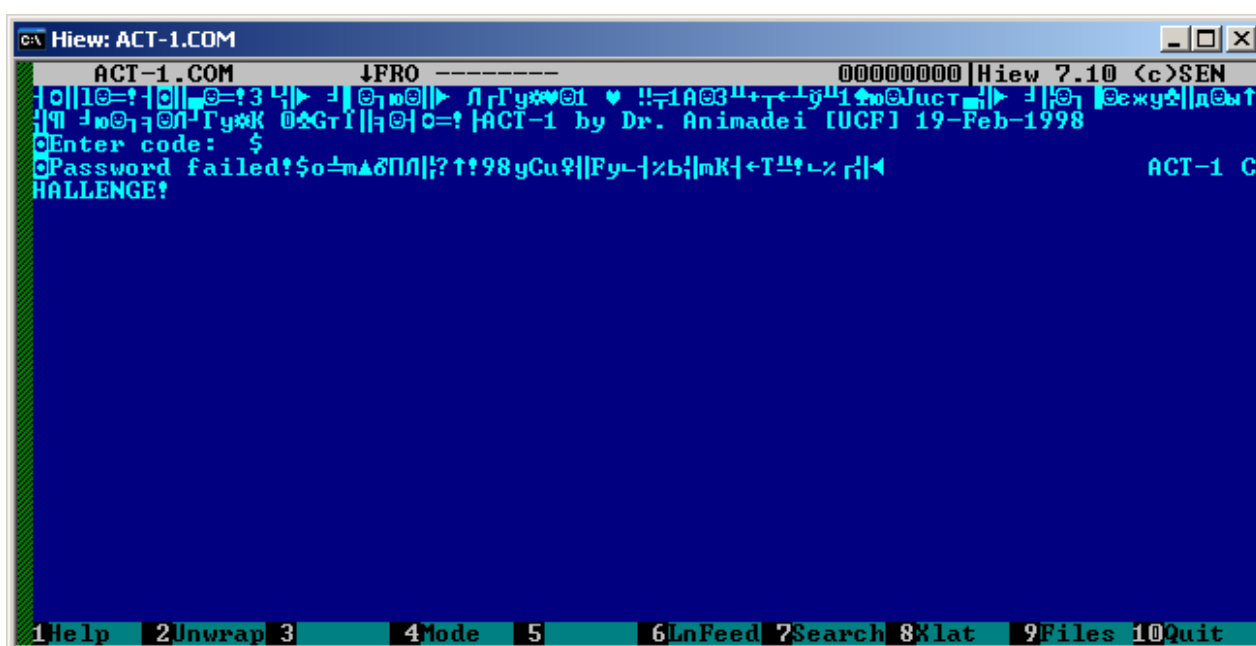
- шестнадцатеричный редактор файлов неограниченной длины;
- уникальное средство поиска ассемблерных команд по маске;
- встроенный ассемблер;
- встроенный дизассемблер;
- интерпретируемая крипт-система;
- система интерактивной помощи.

При открытии любого файла, Hiew разделяет экран на три части (сверху вниз). Верхняя информационная строка редактора отображает следующие дан-

ные (слева направо): название файла, режим поиска по тексту (вверх или вниз), смещение текущего байта во всем коде, версию Hiew. Ниже идет окно редактора, в котором отображается текст файла. Нижняя строка редактора указывает на то, какие функциональные клавиши или их комбинации необходимо использовать для получения доступа к встроенным функциям Hiew.

При редактировании выбранного файла Hiew может работать в трех режимах, смена между которыми осуществляется по нажатию клавиши Enter.

Текстовый режим – редактируемый файл отображается в виде последовательности ASCII-кодов.



Шестнадцатеричный режим – содержимое файла отображается в комбинированном шестнадцатеричном и текстовом виде. Окно редактора поделено на три колонки (слева направо): смещение, в каждой строке отображаются 16 байт кода и соответствующие этим байтам ASCII-символы.

```

C:\ Hiew: ACT-1.COM
ACT-1.COM  ↓FRO ----- 00000000 Hiew 7.10 <c>SEN
00000000: B4 09 BA 6C-01 CD 21 B4-0A BA DC 01-CD 21 33 C0 00 00 00 00
00000010: B9 10 00 BE-DD 01 BF EE-01 BA 10 00-8B DA 83 E3 00 00 00 00
00000020: 0F 03 01 31-00 03 00 13-D1 31 41 01-33 D0 2B C2 00 00 00 00
00000030: 1B C1 F7 D0-31 06 EE 01-4A 75 E1 E2-DC B9 10 00 00 00 00 00
00000040: BE CC 01 BF-DE 01 F3 A6-E3 05 BA A4-01 EB 18 B9 00 00 00 00
00000050: 14 00 BE EE-01 BF B8 01-8B D9 83 E3-0F 8A 00 30 00 00 00 00
00000060: 05 47 E2 F4-BA B8 01 B4-09 CD 21 C3-41 43 54 2D 00 00 00 00
00000070: 31 20 62 79-20 44 72 2E-20 41 6E 69-6D 61 64 65 00 00 00 00
00000080: 69 20 5B 55-43 46 5D 20-31 39 2D 46-65 62 2D 31 00 00 00 00
00000090: 39 39 38 0D-0A 0A 45 6E-74 65 72 20-63 6F 64 65 00 00 00 00
000000A0: 3A 20 20 24-0D 0A 0A 50-61 73 73 77-6F 72 64 20 00 00 00 00
000000B0: 66 61 69 6C-65 64 21 24-6F CF 6D 1E-0B 8F 8B CC 00 00 00 00
000000C0: 3F 18 21 39-38 E3 43 75-0C B6 46 79-1C B4 25 9C 00 00 00 00
000000D0: B9 6D 8A B4-1B 54 CA 21-1C 25 DA B9-11 00 00 00 00 00 00
000000E0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 41 43 00 00 00 00
000000F0: 54 2D 31 20-43 48 41 4C-4C 45 4E 47-45 21 00 00 00 00 00 00
1Help 2PutBlk 3Edit 4Mode 5Goto 6DatRef 7Search 8Header 9Files 10Quit

```

Режим ассемблера – в этом режиме редактор Hiew пытается дизассемблировать код программы. Окно редактора в этом режиме поделено на четыре колонки (слева направо): смещение команды, шестнадцатеричный код команды, мнемоника команды, операнды. Также справа, после символа «;» (точка с запятой) отображаются комментарии к коду, вставленные Hiew.

```

C:\ Hiew: ACT-1.COM
ACT-1.COM  ↓FRO ----- a16 00000000 Hiew 7.10 <c>SEN
00000000: B409          mov     ah,009 ;"0"
00000002: BA6C01        mov     dx,0016C ;"01"
00000005: CD21          int     021
00000007: B40A          mov     ah,00A ;"0"
00000009: BADC01        mov     dx,001DC ;"0_"
0000000C: CD21          int     021
0000000E: 33C0          xor     ax,ax
00000010: B91000        mov     cx,00010 ;" 1"
00000013: BEDD01        mov     si,001DD ;"0|"
00000016: BFEE01        mov     di,001EE ;"0x"
00000019: BA1000        mov     dx,00010 ;" 1"
0000001C: 8BDA          mov     bx,dx
0000001E: 83E30F        and     bx,00F ;"*"
00000021: 0301          add     ax,[bx][di]
00000023: 3100          xor     [bx][si],ax
00000025: 0300          add     ax,[bx][si]
00000027: 13D1          adc     dx,cx
00000029: 314101        xor     [bx][di][01],ax
0000002C: 33D0          xor     dx,ax
0000002E: 2BC2          sub     ax,dx
00000030: 1BC1          sbb     ax,cx
00000032: F7D0          not     ax
00000034: 310EE01       xor     [01EE],ax
1Help 2PutBlk 3Edit 4Mode 5Goto 6Refer 7Search 8Header 9Files 10Quit

```

Полный список команд для каждого режима работы Hiew можно получить, вызвав экран помощи путем нажатия клавиши F1.

### Отладчик Turbo Debugger

Отладчик Turbo Debugger представляет собой средство динамического



исследования программ. Как и любой отладчик, Turbo Debugger позволяет изменять код, содержимое памяти и регистров во время исполнения программы.

Некоторые возможности Turbo Debugger:

- вычисление любых выражений языка Си, C++, Паскаль и Ассемблера;
- настраиваемое размещение информации на экране;
- доступ к ассемблеру и процессору по мере необходимости;
- мощные средства использования точек останова и протокола регистрации;
- запись нажатий клавиш (макрокоманды);
- средства обратной трассировки отлаживаемой программы;
- полная поддержка языка C++ семейства компиляторов Borland C++;
- возможности отладки резидентных в памяти программ и драйверов устройств;
- возможности отладки прикладных программ Microsoft Windows;
- интерактивная, контекстно-зависимая система подсказки.

При открытии любого исполняемого файла Turbo Debugger разделяет экран на три части (сверху вниз). Верхняя строка позволяет получить доступ к меню программы по нажатию клавиши **F10**. Ниже располагается окно редактора. Самая нижняя строка редактора – строка состояния, в которой указываются, в зависимости от контекста, какие функциональные клавиши **F1-F10** надо нажать, для получения доступа к встроенным функциям Turbo Debugger, либо пояснения для текущего режима работы.

Окно редактора состоит из следующих частей.

Адрес	Команда	Операнды	Регистры
cs:0100	mov	ah,09	ax=0000, bx=0000, cx=0000, dx=0000, si=0000, di=0000, bp=0000, sp=FFFE, ds=52D1, es=52D1, ss=52D1, cs=52D1, ip=0100
cs:0102	mov	dx,016C	
cs:0105	int	21	
cs:0107	mov	ah,0A	
cs:0109	mov	dx,01DC	
cs:010C	int	21	
cs:010E	xor	ax,ax	
cs:0110	mov	cx,0010	
cs:0113	mov	si,01DD	
cs:0116	mov	di,01EE	
cs:0119	mov	dx,0010	
cs:011C	mov	bx,dx	
cs:011E	and	bx,000F	
cs:0121	add	ax,[bx+di]	
cs:0123	xor	[bx+si],ax	

**Окно кода** – состоит из четырех столбцов: адрес команды, код команды, мнемоника команды, операнды.

**Окно содержимого основных регистров** – регистры общего назначения (AX, BX, CX, DX, SI, DI, BP, SP), сегментные регистры (DS, ES, SS, CS), указатель команд (IP), регистр флагов (FLAGS – его биты вынесены в отдельный столбец).

**Окно дампа памяти** – отображает содержимое памяти в шестнадцатеричном виде и ASCII-коде.

**Окно состояния стека** – отображает содержимое стека.

Разработчиками ПО Turbo Debugger используется для решения двух труднейших проблем процесса отладки: поиска места нахождения ошибки и ее причины. Turbo Debugger помогает преодолеть эти трудности при помощи трассировки, пошагового выполнения, просмотра, изменения и прослеживания.

**Трассировка.** Программа выполняется по одному оператору.

**Обратная трассировка.** Имеется возможность выполнить код в обратном порядке.

**Пошаговое выполнение.** Имеется возможность выполнять программу по одному оператору, но пропуская вызовы процедур и функций.

**Просмотр.** Turbo Debugger может создать специальное окно для показа самой различной информации – переменных, их значений, точек останова, содержимого стека, файлов регистрации, данных, файлов исходных текстов, со-

держимого памяти, регистров, информацию о состоянии процессора и т.п.

**Проверка.** Turbo Debugger может получать содержимое сложных структур данных из отлаживаемой программы.

**Изменение.** Можно изменить содержимое переменной (как локальной, так и глобальной) на новое значение.

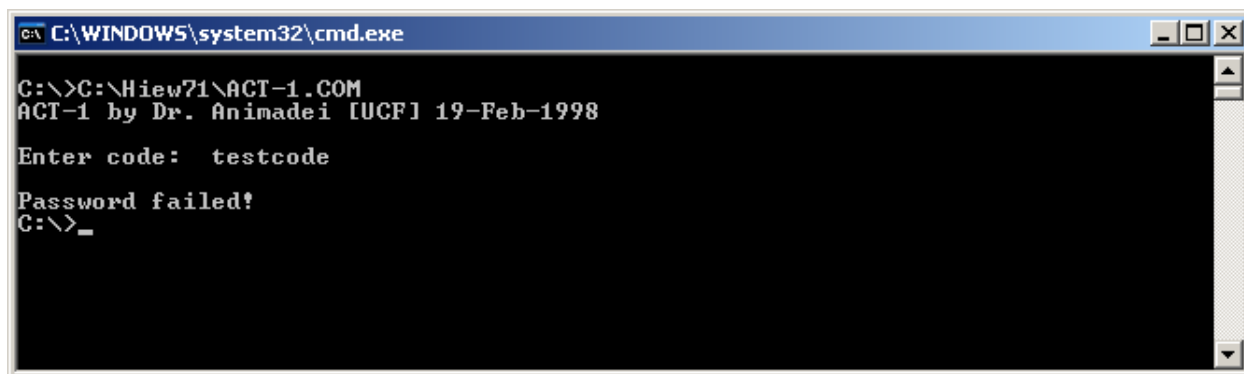
**Прослеживание.** Можно выделить некоторые программные переменные и проследивать изменение их значений в процессе исполнения программы.

Злоумышленник также может использовать все перечисленные возможности Turbo Debugger, но уже с целью динамического поиска и исследования защитных механизмов программы, а также их обхода.

### **Порядок выполнения лабораторной работы:**

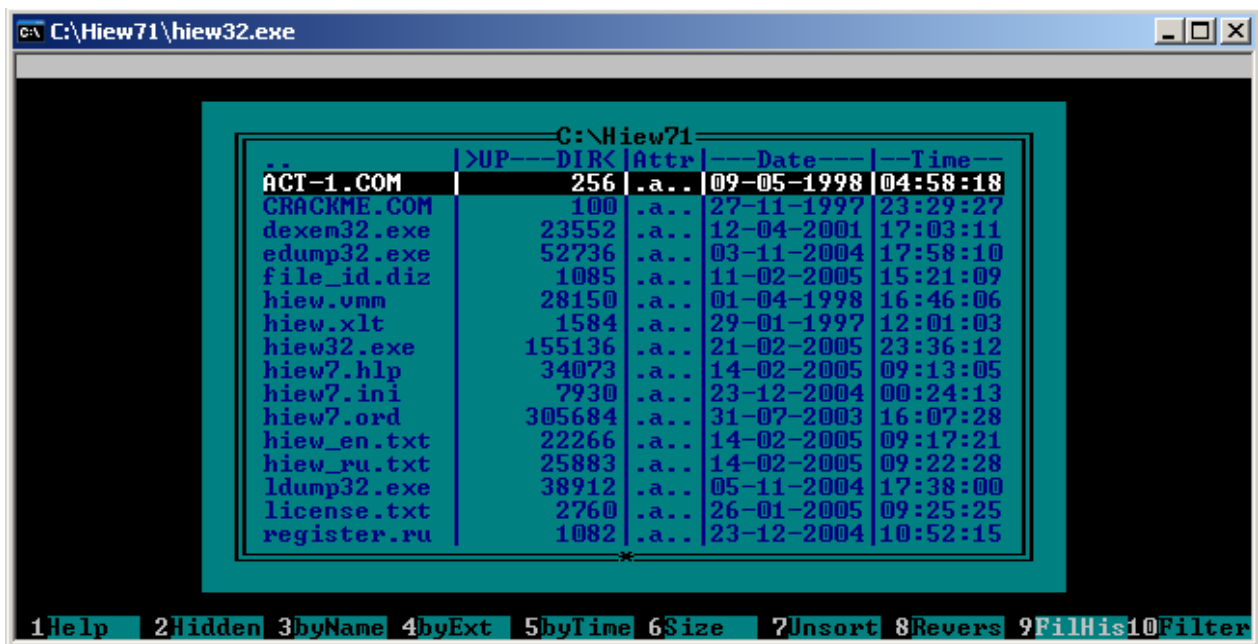
#### Исследование программы Act-1.com

1. В консольном режиме (ПУСК>ВЫПОЛНИТЬ>cmd) запустить программу АСТ-1.COM и внешне изучить механизм ее работы. Внести описание работы программы в отчет.

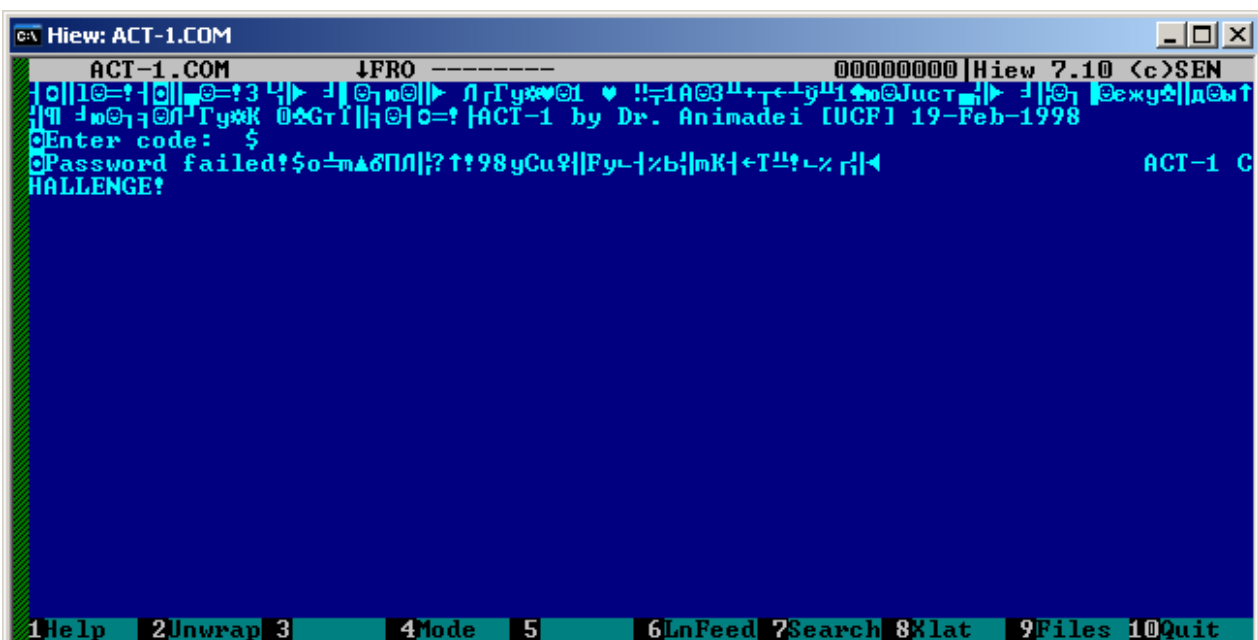


```
C:\WINDOWS\system32\cmd.exe
C:\>C:\Нiew71\ACT-1.COM
ACT-1 by Dr. Animadei [UCF] 19-Feb-1998
Enter code: testcode
Password failed!
C:\>_
```

2. Далее исследуем механизм проверки пароля. Запустите редактор кода Нiew. Загрузите в него программу АСТ-1.COM.



- Внесите в отчет дизассемблированный код программы. Для этого в режиме дизассемблера (F4>Decode) можете воспользоваться комбинацией клавиш **ALT+P**, для сохранения текущего окна в файл (для этого укажите выходной файл), либо в буфер обмена (для этого укажите вместо имени файла знак «\*»).
- Исследовать код программы в режиме **Text** (F4>Text). Какие осмысленные текстовые данные содержатся в программе? Внесите их в отчет. Режимы отображения данных переключаются нажатием клавиши **Enter**, либо **F4**.



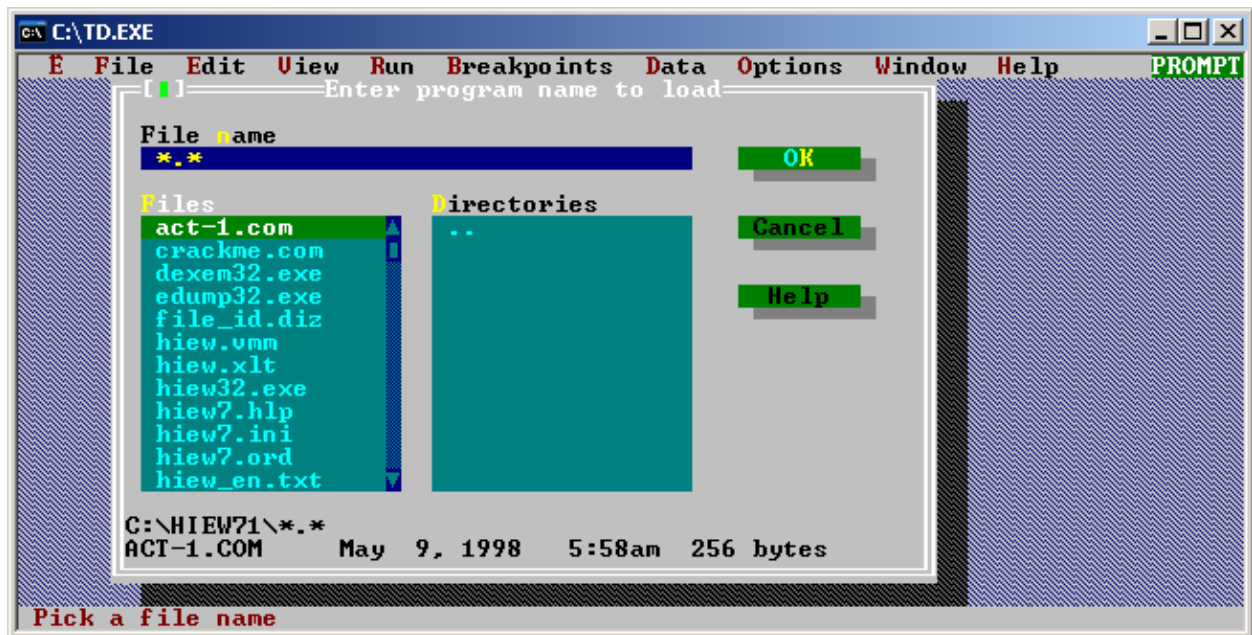
5. Переключиться в режим **Hex** – для просмотра шестнадцатеричного кода (F4>Hex) и внести в отчет примерный адрес начала области данных (где примерно начинаются текстовые данные).

6. Внесите в отчет адреса начала и окончания обнаруженных текстовых строк.
7. Переключиться в режим **Decode** – для дизассемблирования программы (F4>Decode).

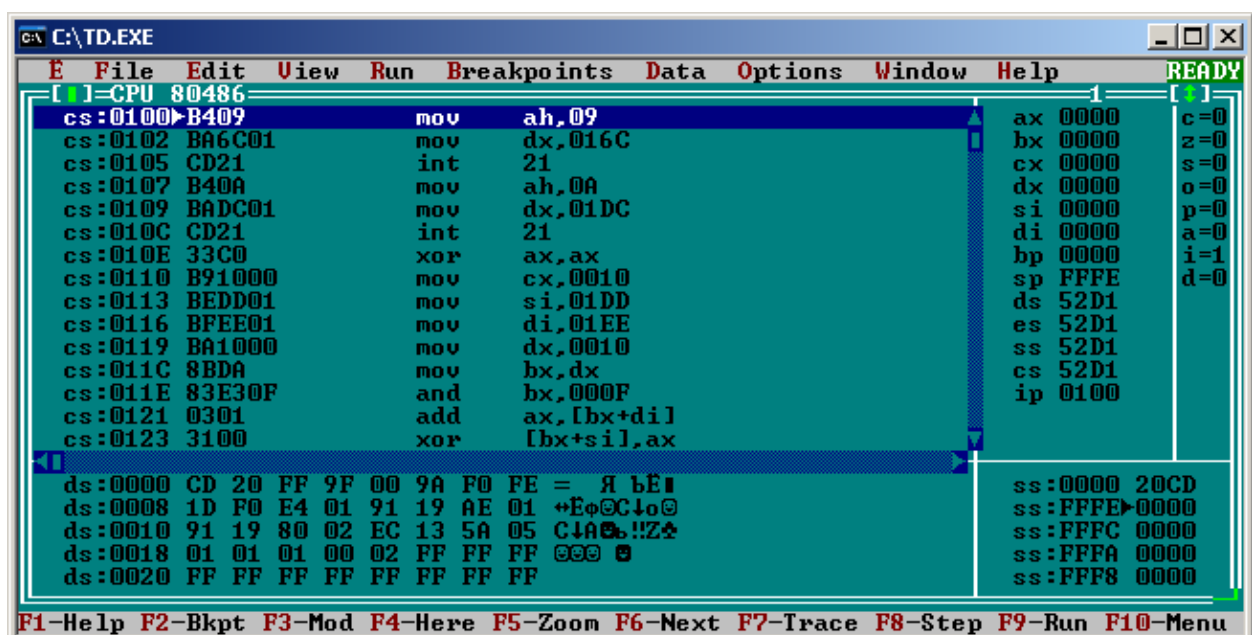
8. Руководствуясь дизассемблированным текстом программы внести в отчет точный адрес окончания работы программы (способ, который

чаще всего применяется для COM-файлов – использование команды RETN).

9. Опираясь на полученную выше информацию, определите и внесите в отчет приблизительный размер области кода программы и приблизительный размер области данных?
10. Запустите отладчик Turbo Debugger. Загрузите в него программу ACT-1.COM для отладки (FILE>OPEN>ACT-1.COM).



11. Сравните данные, полученные отладчиком Turbo Debugger и редактором Hiew.



Address	Instruction	Hex	ASCII
00000000	mov	ah, 009	"0"
00000002	mov	dx, 0016C	"01"
00000005	int	021	
00000007	mov	ah, 00A	"0"
00000009	mov	dx, 001DC	"0"
0000000C	int	021	
0000000E	xor	ax, ax	
00000010	mov	cx, 00010	">"
00000013	mov	si, 001DD	"0"
00000016	mov	di, 001EE	"0"
00000019	mov	dx, 00010	">"
0000001C	mov	bx, dx	
0000001E	and	bx, 00F	"&"
00000021	add	ax, [bx][di]	
00000023	xor	[bx][si], ax	
00000025	add	ax, [bx][si]	
00000027	adc	dx, cx	
00000029	xor	[bx][di][01], ax	
0000002C	xor	dx, ax	
0000002E	sub	ax, dx	
00000030	sbb	ax, cx	
00000032	not	ax	
00000034	xor	[01EE], ax	

12. Подробно объясните назначение группы команд по адресам: 00000000-00000006. Внесите команды и объяснение в отчет.

13. Какое сообщение выводит данная группа команд? Внесите его в отчет.

14. Убедитесь в правильности вашего объяснения при помощи Turbo Debugger. Для этого выполните трассировку указанных команд. Убедитесь также в правильности изменения регистров процессора во время исполнения команд.

Приведем некоторые команды Turbo Debugger:

**F2** – установить точку останова в указанную курсором позицию;

**F7** – трассировка всей программы по шагам, со входом во все подпрограммы;

**F8** – трассировка всей программы по шагам, без входа в подпрограммы;

**F9** – запуск программы;

**Пробел** – изменение содержимого под курсором;

**ALT+F5** – наблюдение за выполнением программы;

**CTRL+F2** – перезапуск программы.

15. Исследовать группу команд по адресам 00000007-0000000D? Внесите команды и объяснение их действий в отчет.

16. Убедитесь в правильности вашего объяснения при помощи отладчика. Для этого выполните трассировку указанных команд.



17. Какой длины сообщение можно вводить в качестве запрашиваемого пароля и почему? Для объяснения воспользуйтесь справочником по прерываниям. Внесите ее в отчет.
18. Используя найденный ранее адрес начала строки о неправильном вводе пароля найдите в тексте программы группу команд, которая выводит данное сообщение на экран. Внесите команды и их адреса в отчет. Учтите, что команды могут располагаться в тексте программы не обязательно по порядку.
19. Изменить команду короткого безусловного перехода JMPs на вывод сообщения о неправильном вводе пароля (находящуюся по адресу 0000004D), двумя командами NOP, т.к. команда JMPs двухбайтовая. Для этого установите курсор по адресу команды JMPs и нажмите клавишу **F3** для входа в редактор, затем клавишу **F2** для вызова встроенного ассемблера. В открывшемся окне сотрите текущую команду, наберите команду NOP, нажмите **Enter** для перехода на следующую строку и наберите еще раз NOP. NOP – это пустая команда, не производящая никаких действий. Нажмите **Esc** для выхода из ассемблера и **F9** для сохранения изменений.

```

Hiew: ACT-1.COM
ACT-1.COM  ↓FWO EDITMODE  a16  0000004D | Hiew 7.10 <c>SEN
00000048: E305                jcxz  0000004F
0000004A: BAA401             mov   dx,001A4 ;"Cd"
0000004D: EB18              jmps  00000067
0000004F: B91400             mov   cx,00014 ;"q"
00000052: BEE001             mov   si,001EE ;"o"
00000055: BFB801             mov   di,001B8 ;"q"
00000058: 8BD9              mov   bx,cx
0000005A: 83E30F             and   bx,00F ;"*"
0000005D: 8A00              mov   al,[bx][si]
00
00
00
00
00
00000069: CD21              int   021
0000006B: C3                retn
0000006C: 41                inc   cx
0000006D: 43                inc   bx
0000006E: 54                push  sp
0000006F: 2D3120            sub   ax,02031 ;"1"
00000072: 627920            bound di,[bx][di][20]
00000075: 44                inc   sp
00000076: 722E              jb    000000A6
1      2      3      4      5      6      7      8      9      10
  
```

20. Проанализируйте получившийся код, закройте Hiew и запустите про-



грамму. Внесите в отчет реакцию программы.

21. Проанализируйте программу и внесите в отчет объяснение реакции программы на удаление команды JMPS.
22. Восстановите измененную программу при помощи Hiew, либо из заранее сохраненной копии.
23. Перейти к анализу блока команд, начинающих генерацию пароля – с адреса 0000000E. Внесите в отчет группу команд генерации пароля и их адреса. Объясните их назначение.
24. Запустить отладчик. Установить точку прерывания (F2) по адресу cs:011E и запустить программу в отладчике (F9). Для выполнения команды по адресу cs:011E нажмите F7.

The screenshot shows the HIEW debugger window for the file C:\TD.EXE. The CPU is 80486. The assembly window displays the following code:

Address	Disassembly	Comment
cs:0100	B409	mov ah,09
cs:0102	BA6C01	mov dx,016C
cs:0105	CD21	int 21
cs:0107	B40A	mov ah,0A
cs:0109	BADC01	mov dx,01DC
cs:010C	CD21	int 21
cs:010E	33C0	xor ax,ax
cs:0110	B91000	mov cx,0010
cs:0113	BEDD01	mov si,01DD
cs:0116	BFEE01	mov di,01EE
cs:0119	BA1000	mov dx,0010
cs:011C	8BDA	mov bx,dx
cs:011E	83E30F	and bx,000F
cs:0121	0301	add ax,[bx+di]
cs:0123	3100	xor [bx+si],ax

The registers window on the right shows the following values:

Register	Value
ax	0971
bx	000F
cx	000F
dx	01A4
si	01CD
di	01DF
bp	0100
sp	FFFE
ds	52D1
es	52D1
ss	52D1
cs	52D1
ip	0100

The status bar at the bottom shows the following keyboard shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.

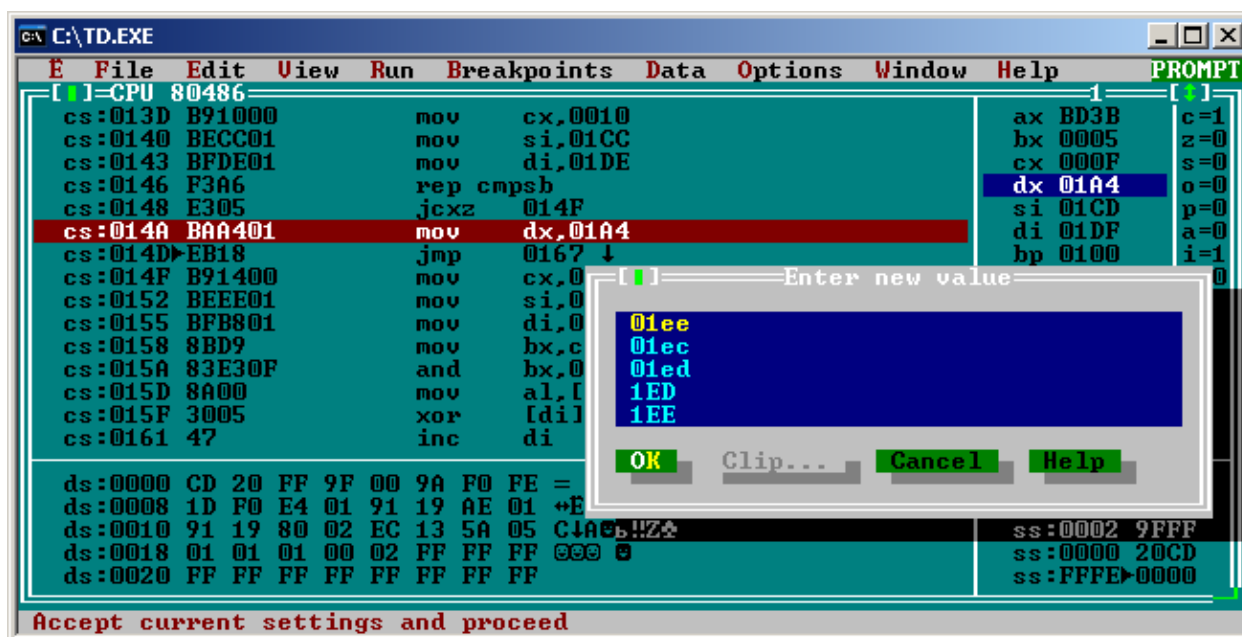
25. Какое значение заносится в регистр BX? Внести его в отчет.
26. В редакторе Hiew определить, с какими двумя блоками данных оперирует блок команд по адресам 00000019-00000003B? Для упрощения исследования используйте трассировку данного блока в отладчике. Что представляет собой первый блок? Что представляет собой второй блок? Внесите эту информацию в отчет.
27. Для чего служит строка «ACT-1 CHALLENGE!»? Внесите эту информацию в отчет.
28. Внесите в отчет назначение блока команд по адресам 0000003D-

00000048. Для упрощения исследования, используйте трассировку данного блока в отладчике.

29. Внесите в отчет назначение блока команд по адресам 0000004F-00000062.

30. Какое значение должна принять строка введенного буфера после всех изменений, чтобы произошел переход на блок команд по адресам 0000004F-00000062.

31. Какие вы можете предложить способы жесткого взлома механизма защиты данной программы? Внести в отчет, какие команды на какие, и по каким адресам, нужно для этого изменить. Объясните суть этих изменений. Для упрощения исследования, установите в отладчике точки останова на исследуемые вами команды, и после выполнения останова по нажатию клавиш **Tab** вы можете перемещаться по окнам отладчика и изменять команды, либо значения регистров, под курсором по нажатию клавиши **Пробел**. Например,



32. Также можете по-прежнему модифицировать код программы в редакторе.

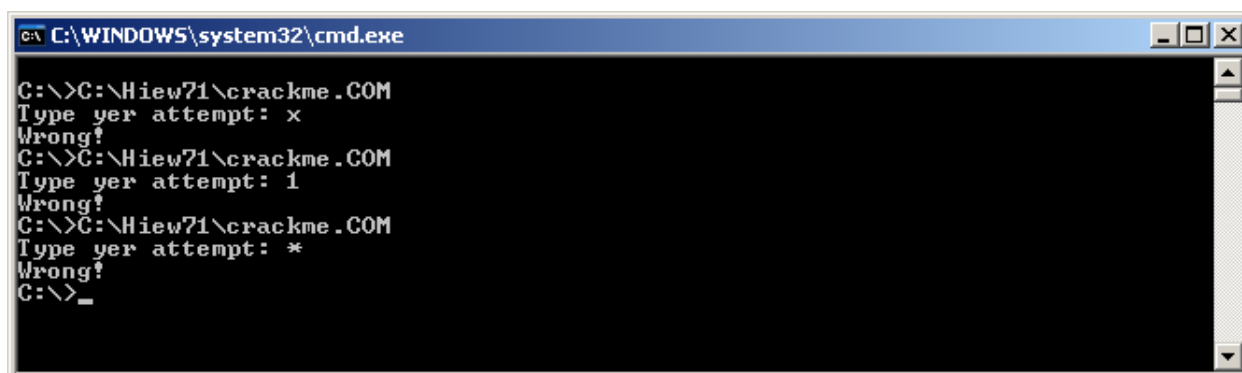
33. Удастся ли при жестком взломе вывести осмысленный итоговый текст? Почему? Внесите ответ и объяснение в отчет.

34. Обобщите все данные, полученные в результате анализа программы, и

внесите в отчет описание алгоритма работы программы, что и как она делает, из каких блоков состоит. В чем заключается суть механизма защиты? Каковы используются механизмы генерации и проверки ключевой информации? Что в итоге должна сделать программа при условии правильности введенной пользователем ключевой информации.

#### Исследование программы Crackme.com

35. В консольном режиме (ПУСК>ВЫПОЛНИТЬ>cmd) запустить файл CRACKME.COM и внешне изучить механизм ее работы. Внести описание работы программы в отчет.



```
C:\WINDOWS\system32\cmd.exe
C:\>C:\Hiew71\crackme.COM
Type ver attempt: x
Wrong!
C:\>C:\Hiew71\crackme.COM
Type ver attempt: 1
Wrong!
C:\>C:\Hiew71\crackme.COM
Type ver attempt: *
Wrong!
C:\>_
```

36. Используя навыки работы с отладчиком и редактором кода, полученные в результате работы над предыдущей программой, исследуем механизм защиты CRACKME.COM. Для этого придется точно также загрузить в HIEW файл CRACKME.COM, исследовать его в различных режимах и воспользоваться справочниками по прерываниям и командам ассемблера.

37. Внесите в отчет дизассемблированный код программы.

38. Какие осмысленные текстовые данные содержатся в программе? Внесите их в отчет.

39. Внесите в отчет примерный адрес начала области данных.

40. Внесите в отчет адреса начала и окончания обнаруженных текстовых строк.

41. Внесите в отчет адрес окончания работы программы.

42. Внесите в отчет приблизительный размер области кода программы.

43. Внесите в отчет приблизительный размер области данных.

44. Что выполняет процедура по адресу 00000036? Укажите в отчете все команды, относящиеся к этой процедуре.
45. Что выполняет группа команд по адресам 00000006-0000000D? Внесите команды в отчет, с пояснением их действий.
46. По какому адресу заносятся данные, вводимые пользователем? Внесите адрес в отчет.
47. Внесите в отчет длину вводимых данных. Объясните, почему она такая.
48. Выделите группу команд, которая выводит сообщение о ложном пароле. Внесите эти команды в отчет.
49. Выделите группу команд, которая выводит сообщение о правильном пароле. Внесите эти команды в отчет.
50. Внесите в код программы исправления таким образом, чтобы всегда выводилось сообщение о правильном пароле. Для этого можно пойти различными путями. Например, можно проделать следующее (не забывайте после любого изменения кода программы восстанавливать ее из сохраненной копии).
1. В отладчике установить точку прерывания на команду сравнения CMP, находящуюся по адресу cs:0127 и после ее выполнения принудительно установить флаг Z равный 1.

The screenshot shows a debugger window titled 'C:\TD.EXE'. The assembly window displays the following instructions:

```

cs:0106 BA6001 mov dx,0160
cs:0109 B8000A mov ax,0A00
cs:010C CD21 int 21
cs:010E BA4E01 mov dx,014E
cs:0111 66B84E455453 mov eax,5354454E
cs:0117 660FB60E6201 movzx ecx,byte ptr [0162]
cs:011D 66C1C006 rol eax,06
cs:0121 32E0 xor ah,al
cs:0123 02C1 add al,cl
cs:0125 E2F6 loop 011D
cs:0127 663DE82227D6 cmp eax,D62722E8
cs:012D 7403 je 0132
cs:012F BA5701 mov dx,0157
cs:0132 E80100 call 0136
cs:0135 C3 ret

```

The registers window on the right shows the following values:

```

eax F6FB22E8 c=0
ebx 000052D1 z=1
ecx 00000027 s=0
edx 0000014E o=0
esi 0000FFFF p=1
edi 0000173B a=0
ebp 00000100 i=1
esp 0000FFFE d=0
ds 52D1
es 52D1
fs 0000
gs 0000
ss 52D1
cs 52D1
ip 012D

```

The status bar at the bottom shows the following shortcuts:

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

2. В редакторе кода изменить команду сравнения

СМР EAX, D62722E8, находящуюся по адресу 00000027 на команду СМР EAX, EAX, а появившиеся лишние три байта заменить командой NOP.

The screenshot shows a Windows XP desktop with a single application window titled "C:\ Hiaw: CRACKME.COM". The window is an assembly editor, displaying assembly code for a program named "CRACKME.COM". The code is in 16-bit mode, as indicated by the "16-bit opcode" label. The code includes instructions like "add al,cl", "loop", "cmp eax,eax", "nop", "je", "mov dx,00157", and "call". A status bar at the bottom shows a progress bar with 10 segments, with the first segment highlighted in green.

Address	Disassembly	Comment
00000023: 02C1	add	al,cl
00000025: E2F6	loop	
00000027: 663BC0	cmp	eax,eax
0000002A: 90	nop	
0000002B: 90	nop	
0000002C: 90	nop	
0000002D: 7403	je	
0000002F: BA5701	mov	dx,00157 ; "OW"
00000032: E80100	call	
0000003C: 7970	jns	
0000003E: 65207965	and	gs:[bx][di][65],bh
00000042: 7220	jb	
00000044: 61	popa	
00000045: 7474	je	
00000047: 65	gs:	
00000048: 6D	insw	
00000049: 7074	jo	
0000004B: 3A20	cmp	ah,[bx][si]

3. В редакторе кода изменить команду условного перехода JE 000000032, находящуюся по адресу 0000002D, на команду безусловного перехода JMP 000000032.

```
C:\ Hiew: CRACKME.COM
CRACKME.COM  ↓FPO EDITMODE  a16  0000002D | Hiew 7.10 <c>SEN
00000000: BA3B01          mov     dx,0013B ;"@";
00000003: E83000          call    00000036
00000006: BA6001          mov     dx,00160 ;"@";
00000009: B8000A          mov     ax,00A00 ;"@";
0000000C: CD21          int     021

Assembler
jmps 00000032
16-bit opcode

00000023: 02C1          add     al,cl
00000025: E2F6          loop    0000001D
00000027: 663DE8227D6   cmp     eax,0D62722E8 ;"n" "m"
0000002D: EB03          jmps    00000032
0000002F: BA5701          mov     dx,00157 ;"@";
00000032: E80100          call    00000036
00000035: C3          retn
00000036: B409          mov     ah,009 ;"o"
00000038: CD21          int     021
0000003A: C3          retn
0000003B: 54          push    sp
0000003C: 7970          jns     000000AE
0000003E: 65207965      and     gs:[bx][di][65],bh
```

4. Реализуйте все варианты отключения защиты и внесите их результаты в отчет.

51. Восстановите исходный код программы. Далее исследуйте механизм проверки пароля. Внесите в отчет группу команд, относящихся к про-

цедуре проверке пароля, и их адреса.

52. Внесите в отчет назначение регистра ЕСХ в процедуре проверки пароля. Сколько раз выполняется цикл процедуры проверки пароля?

53. Каково начальное значение регистра ЕАХ? Внесите его в отчет. Для упрощения поиска ответа на этот и последующие вопросы можете использовать отладчик, установив точки прерывания на первую и последнюю команды процедуры проверки пароля.

54. Какое значение должен принять регистр ЕАХ после окончания цикла для того, чтобы пароль оказался верным?

55. Внести в отчет назначение пароля в алгоритме проверки.

56. Напишите алгоритм атаки на пароль.

Обобщите все данные, полученные в результате анализа программы, и внесите в отчет описание алгоритма работы программы, что и как она делает, из каких блоков состоит. Каков механизм проверки ключевой информации? Как можно узнать верный пароль? Каковы возможные варианты взлома данной программы и т.д.

### **Контрольные вопросы:**

1. Опишите приемы, используемые злоумышленником при отладке и дизассемблировании ПО.

2. Почему шифрование кода программы считается наиболее предпочтительным для защиты от отладчиков и дизассемблеров?

3. Какие способы шифрования кода программы выделяют?

4. Опишите возможности редактора кода Hiew.

5. Опишите возможности отладчика Turbo Debugger

## **4.5. Лабораторная работа № 5**

**Наименование лабораторной работы** – Защита конфиденциальной информации с помощью программно-аппаратного комплекса Secret Disk.

**Время на выполнение** – 4 часа.

**Цель** – изучение способов программно-аппаратной защиты конфиденциальной информации. Знакомство с программно-аппаратным комплексом Secret Disk.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками применения программно-аппаратных комплексов защиты конфиденциальной информации.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, ПАК Secret Disk.

### **Краткие теоретические сведения:**

Система криптографической защиты информации Secret Disk предназначена для создания и обслуживания секретных дисков.

Секретный диск представляет собой логический диск, информация на котором хранится в зашифрованном виде. Он становится доступным пользователю только после того, как к ПК будет подключен электронный ключ (смарт-карту, e-Token) и введен пароль. Такие диски не являются реальными физическими устройствами, они виртуальны, их существование – результат работы VxD-драйвера системы Secret Disk. Secret Disk создает для секретного диска специальный файл секретного диска и с помощью своего драйвера подключает его на правах физического устройства.

Активация электронного идентификатора перед его использованием в Secret Disk заключается в построении случайной последовательности чисел и записи ее в электронный идентификатор. Можно сохранить его в файл и восстановить из файла. Из файла ключ можно считать только в тот электронный идентификатор, из которого он был сохранен.

У каждого секретного диска имеется рабочий ключ, с помощью которого осуществляется шифрование данных. Данный ключ представляет собой случайную последовательность символов. Рабочий ключ хранится в заголовке файла секретного диска и тоже зашифрован. Для его шифрования система использует комбинацию пароля доступа к диску и личного ключа пользователя, записанного в электронном идентификаторе в момент его активизации.

Secret Disk поддерживает следующие аппаратные устройства идентификации пользователя.

1. Электронные ключи HASP.
2. Электронные идентификаторы e-Token.
3. Смарт-карты.

При записи на секретный диск данные автоматически шифруются, а при чтении – расшифровываются. Для идентификации пользователя необходимо выполнить следующие действия.

1. Ввести пароль доступа к секретному диску. Пароль указывается при создании секретного диска.
2. Присоединить к компьютеру свой личный идентификатор.

Для шифрования информации в Secret Disk могут быть использованы следующие алгоритмы.

1. Собственный алгоритм преобразования данных системы Secret Disk с длиной ключа 128 бит.
2. Криптографический алгоритм ГОСТ 28147-89 с длиной ключа 256 бит.
3. Алгоритм RC4.

Данную систему можно использовать также для архивирования данных.

В системе Secret Disk предусмотрена функция защиты от того, что злоумышленник попытается силой заставить Вас отдать электронный идентификатор и назвать пароль. На данный случай в системе Secret Disk предусмотрен режим работы под принуждением. Пользователь называет злоумышленнику «пароль входа под принуждением». При вводе этого пароля система некоторое время «делает вид», что введен правильный пароль, а затем стирает данные в



электронном идентификаторе и симулирует какой-либо сбой: зависание компьютера или ошибку чтения из файла.

Реализовано ведение журнала обращений к секретному диску – контроль и протоколирование работы пользователей с секретным диском.

В системе Secret Disk реализована функция «Красная кнопка» – для мгновенного отключения всех секретных дисков и стирания памяти электронного идентификатора в случае экстренной необходимости.

Реализована функция блокирования ОС – запуск скринсейвера (с блокировкой мыши и клавиатуры), выйти из которого можно только подключив электронный идентификатор и/или введя пароль.

Действия в случае утери электронного идентификатора или пароля:

1. Был забыт пароль к диску – следует используя записанный на аварийной дискете рабочий ключ диска заново задать для этого диска пароль доступа и личный ключ.

2. Был потерян электронный идентификатор

2.1. Приобрести электронный идентификатор.

2.2. Подключить его к компьютеру.

2.3. Активировать новый электронный идентификатор.

2.4. Используя записанный на аварийной дискете рабочий ключ диска, заново задать для диска пароль доступа и личный ключ.

3. Был затерт личный ключ в идентификаторе (в результате нажатия красной кнопки), но есть аварийная дискета с сохраненным личным ключом – следует с аварийной дискеты загрузить в электронный идентификатор личный ключ.

4. Был затерт записанный в электронном идентификаторе личный ключ. Однако нет сохраненного файла с личным ключом.

4.1. Заново активизировать электронный идентификатор.

4.2. Используя записанный на аварийной дискете рабочий ключ диска, заново задать для этого диска пароль доступа и личный ключ.

## **Порядок выполнения лабораторной работы:**

### **1. Организация секретного диска**

1.1. Загрузить администратор секретных дисков.

1.2. Создать секретный диск, который автоматически будет подключаться при запуске системы. Описать данный диск как «ДИСК1». Выделить объем данному секретному диску – 4Мб, который может расширяться до размера 16 Мб.

1.3. Задать пароль для доступа к секретному диску. В качестве электронного идентификатора пользователя выбрать электронный ключ HASP. В качестве алгоритма шифрования выбрать алгоритм RC4.

1.4. Активизировать выданный Вам электронный идентификатор, в результате чего в него запишется случайная последовательность символов. Сгенерированный личный ключ сохранить на дискете (для возможности отката).

1.5. Задать пароль для входа в систему под принуждением (который можно сказать злоумышленнику в экстренной ситуации).

1.6. Сгенерировать рабочий ключ диска и сохранить его на дискете (для возможности отката).

1.7. Подключить к системе секретный диск.

1.8. Открыть в Word текстовый документ, внести в него информацию и сохранить на секретном диске.

1.9. Попытаться в проводнике проводить операции над файлами на секретном диске (копирование – удаление и т.д.).

1.10. Открыть в Word созданный файл с секретного диска.

1.11. Отключить секретный диск.

1.12. Попытаться обратиться к открытому файлу в Word. Проследить реакцию системы.

1.13. Попытаться подключить секретный диск с отключенным идентификатором.

## 2. Работа с секретным диском

2.1. Просмотреть параметры созданного секретного диска (диск должен быть отключен). Исследовать информацию, выдаваемую по данному диску. Какую информацию хранит о данном диске система? Какие дополнительные параметры появились в данном окне по сравнению с информацией о стандартных дисках?

2.2. Какого рода действия предоставляются пользователю в меню «Параметры»?

2.3. Настроить параметры резервного копирования диска. Расположение – на диске С, резервное копирование – перед подключением диска.

2.4. Реализовать ведение журнала безопасности для данного диска. Тип ведения журнала – «При заполнении журнала новая запись записывается на место самой ранней записи»

2.5. Просмотреть пользователей созданного секретного диска и доступные для администратора операции управления пользователями.

2.6. Исследовать допустимые настройки в меню «Диск->Параметры системы».

2.7. Задать для системы тип хранителя экрана – картинка. Задать комбинации клавиш для экстренной блокировки компьютера как “CTRL+SHIFT+1” и красной кнопки ”CTRL+SHIFT+2”.

2.8. Определить время блокировки компьютера после отключения идентификатора, равное 5 сек.

2.9. Для функции «Действия при принуждении» задать режим «Синий экран» и «Уничтожение содержимого электронного идентификатора».

2.10. Подключить секретный диск. Отключить идентификатор от компьютера и выждать 5 сек, после чего компьютер будет заблокирован вплоть до подключения идентификатора пользователя.

2.11. Заблокировать компьютер путем нажатия комбинации клавиш CTRL+SHIFT+1.

2.12. Отключить секретный диск.

### 3. Работа в экстренных ситуациях

3.1. Подключить секретный диск, после чего воспользоваться комбинацией клавиш «Красная кнопка», что приведет к моментальному отключению секретного диска и уничтожению записей электронных идентификаторов.

3.2. Попытаться подключиться к секретному диску после срабатывания «Красной кнопки». Объяснить причину неудачи.

3.3. Восстановить потерянную информацию первым способом – с аварийной дискеты, где хранится личный ключ пользователя. Подключиться к секретному диску.

3.4. Сохранить рабочий ключ шифрования диска.

3.5. Нажать комбинацию клавиш «Красная кнопка».

3.6. Восстановить потерянную информацию вторым способом 1. Активизировать новый электронный идентификатор 2. Используя аварийную копию рабочего ключа диска, заново задать для диска пароль доступа и личный ключ. Для этого вызвать меню параметров секретного диска, далее изменение пароля и электронного идентификатора, в качестве файла указать аварийную копию рабочего ключа диска.

3.7. Заново задать пароль для входа под принуждением.

### 4. Работа с секретным диском под принуждением

4.1. Подключить секретный диск в режиме входа под принуждением. Проследить за реакцией системы.

4.2. Попытаться подключить секретный диск после перезагрузки системы. Объяснить причину неудачи.

4.3. Восстановить информацию на диске любым из выше представленных способов.

### 5. Работа с архивами.

5.1. Скопировать на секретный диск несколько файлов.

5.2. Нажать кнопку «Сохранить данные». В качестве ключа к архиву указать электронный идентификатор. Алгоритм шифрования – собственный с длиной ключа 128 бит, опцию сжатия.

5.3. Добавить с секретного диска несколько файлов для архивации и заархивировать их.

5.4. Попытаться разархивировать данные с электронным идентификатором и без него.

#### 6. Работа с журналом

6.1. Вызвать журнал безопасности секретного диска.

6.2. Исследовать содержание журнала.

#### **Контрольные вопросы:**

1. В чем состоит преимущество защиты конфиденциальной информации с помощью программно-аппаратном комплекса Secret Disk по сравнению с чисто программными способами реализации защищенных секретных дисков?

2. Какие аппаратные устройства могут быть использованы для идентификации пользователя в комплексе Secret Disk?

3. Охарактеризуйте работу пользователя в режиме входа под принуждением, функции красной кнопки и блокировки ПК. Приведите примеры реальных ситуаций, когда Вы будете использовать ту или иную функцию или режим.

## **4.6. Лабораторная работа № 6**

**Наименование лабораторной работы** – Применение сканеров безопасности.

**Время на выполнение** – 4 часа.

**Цель** – познакомиться на практике с решением задач выявления уязвимостей сетевого узла при помощи сканеров безопасности. Определить способы устранения обнаруженных уязвимостей.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками применения сканеров безопасности для оценки эффективности защиты.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, сканер портов nmap, сканер Nessus.

### **Краткие теоретические сведения:**

Одной из важнейших проблем при анализе защищенности сетевого узла является проблема поиска уязвимостей в системе защиты. Под понятием уязвимость понимают слабость в системе защиты, которая дает возможность реализовать ту или иную угрозу. Под угрозой понимают совокупность условий и факторов, которые могут стать причиной нарушения целостности, доступности и конфиденциальности информации, хранящейся, обрабатываемой и проходящей через сетевую компьютерную систему. Уязвимости могут являться как следствием ошибочного администрирования компьютерной системы, так и следствием ошибок, допущенных при реализации механизмов безопасности разработчиком ПО.

Для облегчения работы специалистов существуют программы, позволяющие сократить суммарно потраченное время на поиск уязвимостей, за счет автоматизации операций по оценке защищенности систем. Такие программы называют сканерами безопасности. Сканеры выявляют слабые места в безопас-

ности на удаленном либо локальном ПК. Некоторые из них способны выдавать рекомендации по устранению обнаруженных уязвимостей.

### Принципы работы сканера безопасности

Основной модуль сканера безопасности подсоединяется по сети к удаленному компьютеру. В зависимости от активных сервисов формируются проверки и тесты. Пассивное сканирование представляет собой механизм пассивного анализа, с помощью которого сканер пытается определить наличие уязвимости без фактического подтверждения ее наличия – по косвенным признакам. Найденные при сканировании каждого порта заголовки сравниваются с таблицей правил определения сетевых устройств, операционных систем и возможных уязвимостей. На основе проведенного сравнения делается вывод о наличии или отсутствии потенциальной уязвимости.

При любой оценке безопасности большая сложность заключается в выяснении списка программного обеспечения, установленного в сети. Наличие точного перечня портов и использующих их служб может быть одним из важнейших условий полной идентификации всех уязвимых мест. Для сканирования всех 131070 портов (от 1 до 65535 для TCP и UDP) на всех узлах может понадобиться много дней и даже недель. Поэтому лучше обратиться к более кратким спискам портов и служб, чтобы определить в первую очередь наличие самых опасных уязвимых мест.

Сканеры безопасности – обоюдоострое оружие. Ими может воспользоваться как администратор компьютерной системы для выявления незащищенных мест, так и злоумышленник.

### Сканер портов nmap

NMap – the Network Mapper. Консольная программа NMap предназначен для сканирования сетей с любым количеством объектов, определения состояния объектов сканируемой сети, а также портов и соответствующих им служб. Для этого NMap использует много различных методов сканирования, таких, как UDP, TCP connect(), TCP SYN (полуоткрытое), FTP proxy (прорыв через ftp),

Reverse-ident, ICMP (ping), FIN, ACK, Xmas tree, SYN и NULL-сканирование (для определения действий соответствующих служб см. «Справку Windows»).

NMap также поддерживает большой набор дополнительных возможностей: определение ОС удаленного хоста с использованием отпечатков стека TCP/IP, «невидимое» сканирование, динамическое вычисление времени задержки и повтор передачи пакетов, параллельное сканирование, определение неактивных хостов методом параллельного ping-опроса, сканирование с использованием ложных хостов, определение наличия пакетных фильтров, прямое RPC-сканирование, сканирование с использованием IP-фрагментации а также произвольное указание IP-адресов и номеров портов сканируемых сетей.

Результатом работы NMap является список отсканированных портов удаленной машины с указанием номера и состояния порта, типа используемого протокола, а также названия службы, закрепленной за этим портом.

Порт характеризуется тремя возможными состояниями: «открыт», «фильтруемый» и «нефильтруемый». Состояние «открыт» означает, что удаленная машина прослушивает данный порт. Состояние «фильтруемый» означает, что межсетевой экран, пакетный фильтр или другое устройство блокирует доступ к этому порту и NMap не смог определить его состояние. «Нефильтруемый» означает, что по результатам сканирования NMap воспринял данный порт как закрытый, при этом средства защиты не помешали NMap определить его состояние. Это состояние NMap определяет в любом случае.

#### Команда использования NMap (в консольном режиме)

nmap [Метод(ы) сканирования] [Опции] <Хост или сеть #1,[#N]>

#### Основные методы сканирования

-sT (scan TCP) – использовать метод TCP connect(). Наиболее общий метод сканирования TCP-портов. Функция connect(), присутствующая в любой ОС, позволяет создать соединение с любым портом удаленной машины. Если указанный в качестве аргумента порт открыт и прослушивается сканируемой машиной, то результат выполнения connect() будет успешным (т.е. соединение будет установлено), в противном случае указанный порт является закрытым,



либо доступ к нему заблокирован средствами защиты. Для того, чтобы использовать данный метод, пользователь может не иметь никаких привилегий на сканирующем хосте. Данный метод сканирования легко обнаруживается целевым (т.е. сканируемым) хостом, поскольку его log-файл будет содержать запро-токолированные многочисленные попытки соединения и ошибки выполнения данной операции. Службы, обрабатывающие подключения, немедленно блокируют доступ адресу, вызвавшему эти ошибки.

-sS (scan SYN) – использовать метод TCP SYN. Этот метод часто называют полуоткрытым сканированием, поскольку при этом полное TCP-соединение с портом сканируемой машины не устанавливается. NMap посылает SYN-пакет, намереваясь открыть настоящее соединение, и ожидает ответ. Наличие флагов SYN|ACK в ответе указывает на то, что порт удаленной машины открыт и прослушивается. Флаг RST в ответе означает обратное. Если NMap принял пакет SYN|ACK, то в ответ он немедленно отправляет RST-пакет для сброса еще не установленного соединения.

-sF,-sX,-sN (scan FIN, scan Xmas, scan NULL) – «невидимое» FIN, Xmas Tree и NULL-сканирование. Эти методы используются в случае, если SYN-сканирование по каким-либо причинам оказалось неработоспособным (некоторые межсетевые экраны и пакетные фильтры ожидают и блокируют поддельные SYN-пакеты на защищенные ими порты).

-sP (scan Ping) – ping-сканирование. Иногда вам необходимо лишь узнать адреса активных хостов в сканируемой сети. NMap делает это, послав ICMP-сообщение «запрос эха» на каждый указанный IP-адрес. Хост, отправивший ответ на эхо, является активным.

-sV (scan Version) – включение режима определения версий служб, за которыми закреплены сканируемые порты. После окончания сканирования будет получен список открытых TCP и/или UDP-портов. Без этой опции в списке напротив каждого порта будет указана служба, которая обычно использует данный порт (эта информация берется из базы данных общеизвестных портов, файл nmap-services).

-sU (scan UDP) – сканировать UDP-порты. Этот метод используется для определения того, какие UDP-порты на сканируемом хосте являются открытыми. На каждый порт сканируемой машины отправляется UDP-пакет без данных. Если в ответ было получено ICMP-сообщение «порт недоступен», то это означает, что порт закрыт. В противном случае предполагается, что сканируемый порт открыт. Надо помнить, что сканирование UDP-портов проходит очень медленно, поскольку практически все ОС следуют рекомендации RFC 1812 (раздел 4.3.2.8) по ограничению скорости генерирования ICMP-сообщений «порт недоступен». Например, ядро Linux ограничивает генерирование таких сообщений до 80 за 4 секунды с простоем 0,25 секунды, если это ограничение было превышено. Microsoft не использует в своих ОС никаких ограничений, поэтому можно достаточно быстро просканировать все 65535 UDP-портов хоста, работающего под управлением ОС Windows.

-sO (scan Open protocol) – сканирование протоколов IP. Данный метод используется для определения IP-протоколов, поддерживаемых сканируемым хостом.

-sI <zombie\_хост[:порт]> (scan Idle) – позволяет произвести «абсолютно невидимое» сканирование портов. Атакующий может просканировать цель, не посылая при этом пакетов от своего IP-адреса. Вместо этого используется метод IdleScan, позволяющий просканировать жертву через так называемый хост-«зомби». Кроме абсолютной невидимости, этот тип сканирования позволяет определить политику доверия между машинами на уровне протокола IP.

-sA (scan ACK) – использовать ACK-сканирование. Этот дополнительный метод используется для определения набора правил межсетевого экрана. В частности, он помогает определить, защищен ли сканируемый хост таким экраном или просто пакетным фильтром, блокирующим входящие SYN-пакеты.

-sW (scan Window) – использовать метод TCP Window. Этот метод похож на ACK-сканирование, за исключением того, что иногда с его помощью можно определять открытые порты точно так же, как и фильтруемые/нефильтруемые.

-sR (scan RPC) – использовать RPC-сканирование. Этот метод используется совместно с другими методами сканирования и позволяет определить программу, которая обслуживает RPC-порт, и номер ее версии.

-sL (scan List) – получить список сканируемых адресов. Эта опция позволяет вам получить список адресов хостов, которые будут просканированы NMap до начала процесса сканирования. Опция может использоваться в случае, когда вам необходимо определить имена большого количества хостов по их адресам и т.д.

#### Дополнительные опции

Эти опции не обязательные (т.е. нормальная работа NMap возможна и без их указания).

-h (show help) – печатает справку по использованию Nmap с указанием опций и краткого их описания, не запуская саму программу.

-P0 (Ping 0) – не производить ping-опрос хостов перед их непосредственным сканированием.

-PT (Ping TCP) – использовать TCP-ping. Вместо отправки запроса ICMP-эха, Nmap отправляет TCP ACK-пакет на сканируемый хост и ожидает ответ. Если хост активен, то в ответ должен прийти RST-пакет.

-PS (Ping SYN) – опция, также используемая для ping-опроса. При этом вместо ACK-пакета TCP-ping используется SYN-пакет. Активные хосты посылают в ответ RST-пакеты (реже SYN|ACK).

-PU [portlist] (Ping UDP) – использовать UDP-ping. NMap отправляет UDP-пакеты на указанный хост и ожидает в ответ ICMP «port unreachable» (или ответы от открытых портов UDP) если хост активен.

-PE (Ping ICMP) – эта опция используется в качестве ping-запроса нормальный ping-пакет (запрос ICMP-эха). Опция применяется для поиска активных хостов, а также адресов сетей с возможностью широковещания. Такие сети пересылают прибывший ICMP-пакет всем своим объектам. Как правило, такие сети представляют собой «живую мишень» для злоумышленника.

-PP – использует пакет ICMP «timestamp request (code 13)» для определения активных хостов.

-PB (Ping Both) – режим ping-опроса по-умолчанию. Использует одновременно запросы типа ACK и ICMP.

-O (Operating system detection) – эта опция позволяет определить операционную систему сканируемого хоста с помощью метода отпечатков стека TCP/IP. Другими словами, NMap активизирует мощный алгоритм, функционирующий на основе анализа свойств сетевого программного обеспечения установленной на нем ОС. В результате сканирования получается формализованный «отпечаток», состоящий из стандартных тестовых запросов и ответов хоста на них. Затем полученный отпечаток сравнивается с имеющейся базой стандартных ответов известных ОС, хранящейся в файле nmap-os-fingerprinting, и на основании этого принимается решение о типе и версии ОС сканируемого хоста. Этот метод требует наличия хотя бы одного закрытого и одного открытого порта на целевом хосте.

-A – Эта опция включает режим additional advanced aggressive, и разрешает опции -O, -sV, -T4, -v.

-I (Ident scan) – использовать reverse-ident сканирование. Протокол ident позволяет вскрыть имя пользователя (username) процесса, использующего TCP, даже если этот процесс не инициализировал соединение. Так, например, вы можете подключиться к порту http и затем использовать identd для поиска на сервере пользователя root. Это может быть сделано только при установлении «полного» TCP-соединения с портом сканируемой машины (т.е. необходимо использовать опцию -sT). NMap опрашивает identd сканируемого хоста параллельно с каждым открытым портом. Естественно, этот метод не будет работать, если на целевом хосте не запущен identd.

-f (use fragmentation) – эта опция используется совместно с SYN, FIN, Xmas или NULL-сканированием и указывает на необходимость использования IP-фрагментации с малым размером фрагментов. Это значительно усложняет

фильтрацию пакетов, работу систем обнаружения и других подобных средств защиты, позволяет NMap скрыть свои действия.

-v (verbose output) – использовать режим «подробного отчета». Эту опцию рекомендуется использовать в любых случаях, поскольку при этом NMap подробно сообщает о ходе выполнения текущей операции.

-iR (input Random) – если вы укажете эту опцию, NMap будет сканировать случайно выбранные им хосты, адреса которых получены с помощью генератора случайных величин. Этот процесс будет длиться, пока вы его не остановите. Функция может пригодиться для статистического исследования Internet.

-p <диапазон(ы)\_портов> (ports) – эта опция указывает NMap, какие порты необходимо просканировать. Например, -p 23 означает сканирование 23 порта на целевой машине. По-умолчанию Nmap сканирует все порты в диапазоне 1-1024, поскольку все они перечислены в файле services.

-F (Fast scan) – быстрое сканирование.

--packet\_trace - показывать все принимаемые и передаваемые пакеты в формате TCPDump.

#### Задание цели

Для сканирования подсети IP-адресов, необходимо указать параметр

/<mask>

– маска, после имени или IP-адреса сканируемого хоста.

Маска может принимать следующие значения:

/0 – сканировать весь Интернет;

/16 – сканировать адреса класса В;

/24 – сканировать адреса класса С;

/32 – сканировать только указанный хост.

NMap позволяет также гибко указать целевые IP-адреса, используя списки и диапазоны для каждого их элемента. Например, необходимо просканировать подсеть класса В с адресом 128.210.\*.\*. Задать эту сеть можно любым из следующих способов:

128.210.\*.\*

128.210.0-255.0-255

128.210.1-50,51-255.1,2,3,4,5-255

128.210.0.0/16

Все эти строки эквивалентны.

Если необходимо просканировать, например, все IP-адреса, оканчивающиеся на 5.6 либо 5.7, то можно указать в качестве целевого IP-адреса строку:

\*.\*.5.6-7

### **Порядок выполнения лабораторной работы:**

1. Выяснить IP-адрес вашего компьютера, маску подсети и основной шлюз. Данную информацию можно получить, посмотрев свойства протокола TCP/IP («Сетевое окружение» – «Свойства»). Или выполнив команду **ipconfig** в командной строке («Пуск» – «Выполнить» – «cmd»).

2. Выберите компьютер-жертву среди компьютеров лаборатории. Определите его IP-адрес. Его можно узнать либо у других студентов, либо уточнить у лаборанта.

3. Откройте консоль («Пуск» – «Выполнить» – «cmd»). Запустите сканер NMap. Используя приведенный в лабораторной работе перечень стандартных опций определите, если возможно:

- имя сканируемого компьютера;
- количество открытых портов;
- имена открытых портов и названия сервисов, соответствующих им.

В отчет внести также использованные для этих целей команды или группу команд.

4. Используя маску, продемонстрируйте возможности NMap при сканировании диапазона ЭВМ в лаборатории. Использовать не менее трех команд. В отчет внесите маску, команды для сканера и полученные результаты с пояснением.

5. Запустить сканер безопасности Nessus. Ввести адрес сканируемой Вами ЭВМ в окне ввода IP адреса и осуществить сканирование безопасности.

Внесите в отчет открытые порты и сервисы, обнаруженные Nessus. Сравните результаты с полученными от NMap.

Возможна ли удаленная DoS-атака на ЭВМ? Если да, внесите в отчет порт, номер и имя сервиса.

6. После окончания сканирования сгенерируйте отчет средствами Nessus. Проанализируйте его.

Какие обнаружены уязвимости?

Какие рекомендации по их устранению выдает Nessus?

7. Используя продукт Metasploit Framework 3.2, реализуйте DoS-атаку на удаленную машину через уязвимость ms06-040, добейтесь успешной реализации данной атаки.

Исследуйте, какие еще из существующих уязвимостей позволяют реализовать успешно DoS-атаку.

8. Используя продукт Metasploit Framework 3.2 получите удаленную командную строку через уязвимость ms06-040. Создайте на удаленной машине нового пользователя и проверьте его наличие. Получите доступ к удаленной машине от имени данного пользователя.

Для создания пользователя через удаленную командную строку используйте команду:

```
net user ИМЯ_ПОЛЬЗОВАТЕЛЯ ПАРОЛЬ /add
```

9. Выполните возможные действия по устранению уязвимостей. Для исправления ошибок реестра используйте команду «Пуск» – «Выполнить» – «regedt32». Кроме того, возможно придется использовать средства администрирования Windows («Панель управления» – «Администрирование» – «Службы», «Управление компьютером»), а также средства аудита пользователей («Панель управления» – «Учетные записи пользователей»). Просканируйте систему заново. Есть ли изменения?

Внесите в отчет уязвимости, которые вам удалось исправить, а также ваши действия по их устранению.

Опираясь на опыт лабораторной работы, внесите в отчет ваши рекомендации по более надежной защите исследованного сетевого узла.

**Контрольные вопросы:**

1. Опишите функции сканера портов nmap и сканера безопасности Nessus.
2. Охарактеризуйте различные режимы сканирования сканера портов nmap.
3. Охарактеризуйте функции продукта Metasploit Framework.
4. В чем заключается принципиальное отличие между SYN сканированием и TCP сканированием?



## **4.7. Лабораторная работа № 7**

**Наименование лабораторной работы** – Работа со сканерами безопасности уровня ОС.

**Время на выполнение** – 4 часа.

**Цель** – познакомиться на практике с решением задач выявления уязвимостей локального узла при помощи сканеров безопасности. Определить способы устранения обнаруженных уязвимостей.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками применения сканеров безопасности для оценки эффективности защиты.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, сканер безопасности SYSTEM SCANNER.

### **Краткие теоретические сведения:**

Сканеры безопасности уровня ОС, в частности SYSTEM SCANNER, позволяют идентифицировать следующие типы уязвимостей локального сканируемого узла:

1. уязвимости прикладных программ и аппаратуры;
2. уязвимости, связанные с недостатками в конфигурировании системы;
3. недостатки в конфигурировании, не согласующиеся с политикой безопасности.

SYSTEM SCANNER может выявлять данные уязвимости в процессе интерактивного или автоматического сканирования и давать рекомендации по устранению данных уязвимостей.

SYSTEM SCANNER включает в себя два компонента:

1. агент, выполняющий собственно сканирование;
2. консоль, управляющая агентом и собирающая от него информацию в процессе сканирования.

Агент не обязательно должен быть установлен на машине с установленной консолью.

SYSTEM SCANNER включает в себя множество политик сканирования, организованных в группы. Политика сканирования каждой группы отвечает за проверку требований безопасности для отдельной области.

Начальные политики (Initial Policies) – при начальной защите ОС. Существует 6 групп политик: Initial 1 – Initial 6.

Политика	Имя	Область проверки
Initial-1	Внешние атаки	Сетевые сервисы – FTP, RAS, доступ к реестру, аудит входа в систему
Initial-2	Суперпользователи	Конфигурация аудита, анализ журналов аудита, учетные записи администраторских групп
Initial – 3	Обычные пользователи	Политика учетных записей, права и привилегии пользователей
Initial – 4	Файлы и устройства	Только для UNIX
Initial – 5	Системные файлы	Файлы системного каталога, критичные ключи в реестре
Initial – 6	Все файлы	Все файлы на хосте

Поддерживающие политики (Maintenance Policies) – предназначены для мониторинга безопасности системы. Политики сканирования данного вида должны запускаться периодически (чем выше номер политики – тем реже).

Политика	Имя	Частота	Область проверки
Maintenance – 1	Внешние атаки	Ежечасно, ежедневно	Сетевые сервисы – FTP, RAS, доступ к реестру, аудит

			входа в систему
Maintenance – 2	Суперпользователи	Ежечасно, ежедневно	Конфигурация аудита, анализ журналов аудита, учетные записи администраторских групп
Maintenance – 3	Системные файлы	Раз в день, раз в 2 дня	Файлы системного каталога, критичные ключи в реестре
Maintenance – 4	Обычные пользователи	Раз в день, раз в неделю	Политика учетных записей, права и привилегии пользователей
Maintenance – 5	Файлы и устройства	Раз в день, раз в неделю	Только для UNIX
Maintenance – 6	Все файлы	Раз в неделю, раз в месяц	Все файлы на хосте

Система SYSTEM SCANNER включает в себя удобные средства генерации отчетов по выполненному сканированию.

#### **Порядок выполнения лабораторной работы:**

1. С помощью системы SYSTEM SCANNER выполнить сканирование Вашего узла по всем 6 политикам типа Initial.
2. Изучить отчеты, сгенерируемые SYSTEM SCANNER для всех 6 групп политик. Проанализировать найденные уязвимости.
3. Используя рекомендации SYSTEM SCANNER закрыть уязвимости, найденные по результатам сканирования.

**Контрольные вопросы:**

1. В чем заключаются функции сканера безопасности уровня ОС?
2. В чем заключается различие между Initial Policies и Maintenance Policies?
3. Перечислите области проверки, относящиеся к политике сканирования на предмет уязвимости к внешним атакам?
4. Перечислите области проверки, относящиеся к политике сканирования на предмет уязвимости обычных пользователей?

#### **4.8. Лабораторная работа № 8**

**Наименование лабораторной работы** – Защита автоматизированных систем от НСД средствами СЗИ НСД Secret Net

**Время на выполнение** – 4 часа.

**Цель** – познакомиться на практике с вопросами проектирования защищенных от НСД автоматизированных систем.

**Требования к результатам обучения основной образовательной программы, достигаемые при проведении лабораторной работы** – владеть навыками применения СЗИ НСД, сертифицированных ФСТЭК.

**Лабораторное оборудование и программное обеспечение** – компьютерный класс с ЛВС, СЗИ НСД Secret Net.

##### **Краткие теоретические сведения:**

Автономный вариант системы защиты Secret Net предназначен для защиты ресурсов рабочей станции локальной сети или неподключенного к сети компьютера.

Система Secret Net дополняет стандартные защитные механизмы ОС Windows функциями, обеспечивающими:

1. идентификацию пользователей при помощи специальных аппаратных средств (iButton, eToken, Smart Card, Smarty);
2. дополнительно к избирательному (дискреционному) управлению доступом, реализованному в ОС Windows, полномочное (мандатное) управление доступом пользователей к конфиденциальной информации на локальных и подключенных сетевых дисках;
3. оперативный контроль работы пользователей компьютера путем регистрации событий, связанных с безопасностью ИС, удобные средства просмотра и представления зарегистрированной информации;
4. контроль целостности программ, используемых пользователями и операционной системой;

5. возможность создания для любого пользователя замкнутой программной среды (списка разрешенных и запрещенных для запуска программ).

Система Secret Net включает в себя следующие компоненты и подсистемы:

- ядро системы защиты;
- подсистема управления;
- подсистема криптографической защиты;
- база данных системы защиты;
- подсистема избирательного управления доступом;
- подсистема разграничения доступа к дискам;
- подсистема полномочного разграничения доступа;
- подсистема замкнутой программной среды;
- подсистема контроля целостности;
- подсистема контроля входа.

#### Основные подсистемы

Ядро системы защиты представляет собой программу, которая автоматически запускается на защищенном компьютере при его включении и функционирует на протяжении всего времени работы компьютера. Ядро системы осуществляет управление подсистемами и компонентами системы защиты и обеспечивает их взаимодействие.

Подсистема регистрации является одним из элементов ядра системы и предназначена для управления регистрацией в журнале безопасности Windows событий, связанных с работой ОС и Secret Net. Эта информация поступает от отдельных подсистем системы защиты, которые следят за происходящими в информационной среде событиями.

Регистрация событий осуществляется системными средствами ОС Windows или средствами системы защиты Secret Net.

Перечень регистрируемых событий устанавливается администратором с помощью подсистемы управления. Для просмотра журнала используется специальная программа подсистемы управления, обладающая развитыми сред-

ствами работы с журналами регистрации.

Подсистема управления располагает средствами для настройки защитных механизмов через управление общими параметрами работы компьютера, свойств пользователей и групп пользователей.

База данных Secret Net предназначена для хранения сведений, необходимых для работы защищенного компьютера. БД Secret Net размещается в реестре ОС Windows и содержит информацию об общих настройках системы защиты, свойствах пользователей и групп пользователей.

Доступ подсистем и компонент системы защиты к данным, хранящимся в БД Secret Net, обеспечивается ядром системы защиты.

Подсистема избирательного управления доступом обеспечивает разграничение доступа пользователей к ресурсам файловой системы, аппаратным ресурсам и ресурсам операционной системы компьютера. Для управления доступом к ресурсам файловой системы, системному реестру и системным средствам управления компьютером используются стандартные средства ОС Windows. Для управления доступом к дискам и портам используются средства Secret Net.

Подсистема полномочного разграничения доступа обеспечивает разграничение доступа пользователей к конфиденциальной информации, хранящейся в файлах на локальных и сетевых дисках. Доступ осуществляется в соответствии с категорией конфиденциальности, присвоенной информации, и уровнем допуска пользователя к конфиденциальной информации.

Подсистема замкнутой программной среды позволяет сформировать для любого пользователя компьютера программную среду, определив индивидуальный перечень программ, разрешенных для запуска.

Подсистема контроля входа обеспечивает идентификацию и аутентификацию пользователя при его входе в систему. Подсистема включает в себя модуль идентификации пользователя, а также может содержать аппаратные средства контроля входа, например, устройства Secret Net TM Card или электронный замок “Соболь”, если они установлены на компьютере, а также программу

драйвер, с помощью которой осуществляется управление этими устройствами. Драйверы входят в комплект поставки и устанавливаются на компьютер вместе с системой Secret Net.

Подсистема контроля целостности осуществляет слежение за неизменностью контролируемых объектов (файлов, ключей системного реестра и т.д.) с целью защиты их от модификации. Для этого определяется перечень контролируемых объектов. Для каждого из объектов рассчитываются эталонные значения контролируемых параметров. Эталонные значения для проверяемых объектов, а также информация о размещении объектов хранятся в пакетах контроля целостности.

Каждый раз при загрузке компьютера, а также при повторном входе пользователя, система получает информацию об актуальной аппаратной конфигурации и сравнивает ее с эталонной.

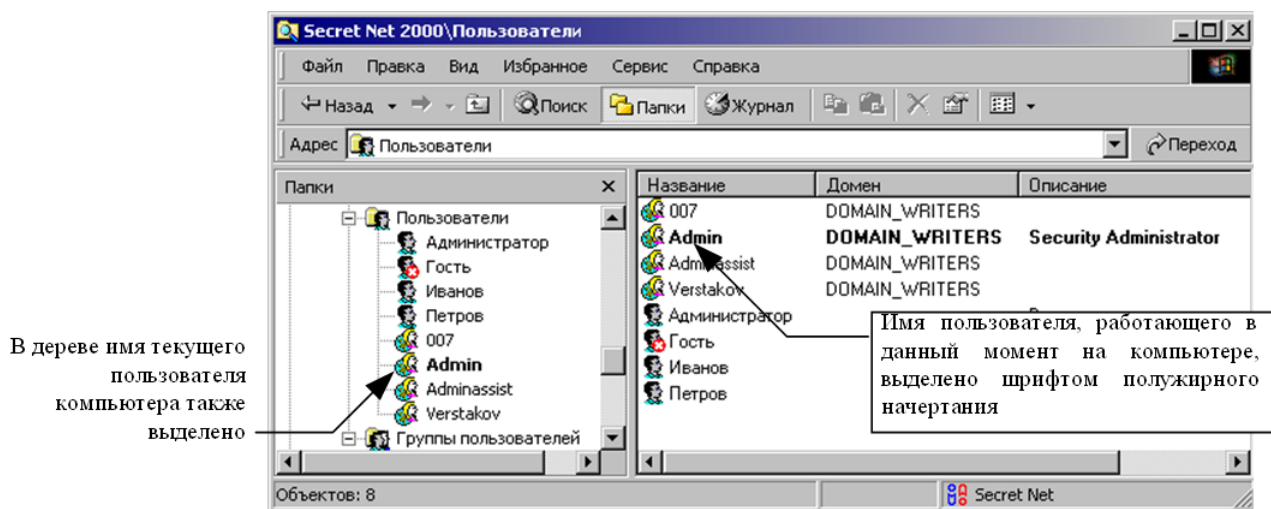
Контроль конфигурации аппаратных средств производится ядром Secret Net. По результатам контроля ядро принимает решение о необходимости блокировки компьютера. Решение принимается после входа пользователя и зависит от настройки свойств пользователя.

## Порядок выполнения лабораторной работы:

### Управление пользователями

#### Создание локального пользователя

1. В окне программы "Проводник" выберите папку "Пользователи".





В правой части окна отобразится список всех пользователей, зарегистрированных на компьютере.

2. Перейдите к правому окну, вызовите контекстное меню и выберите команду “Создать”.

3. Измените имя объекта на имя “User” и нажмите <Enter>.

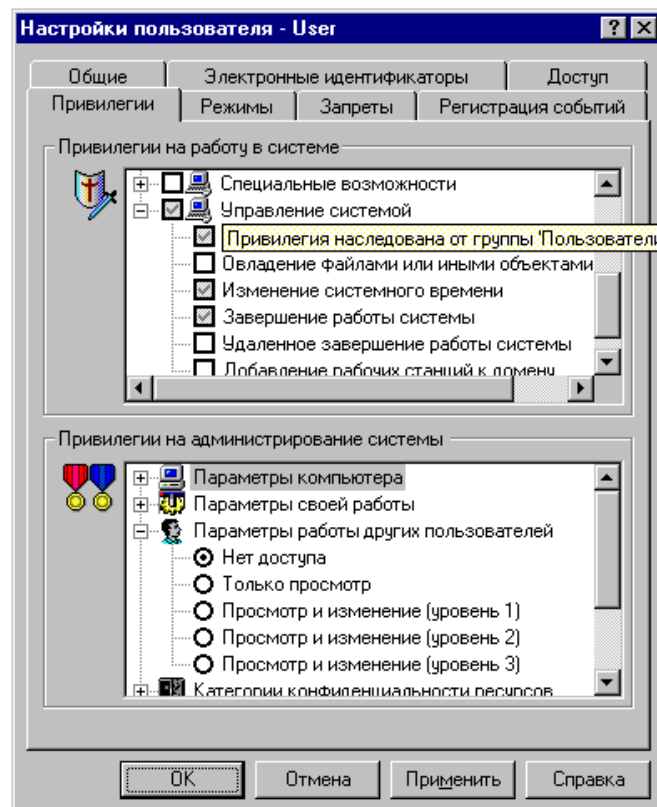
### Предоставление привилегий и управление другими свойствами

#### Непосредственное предоставление привилегий

1. В окне программы "Проводник" выберите пользователя «User», вызовите на экран окно настройки свойств и перейдите к диалогу "Привилегии".

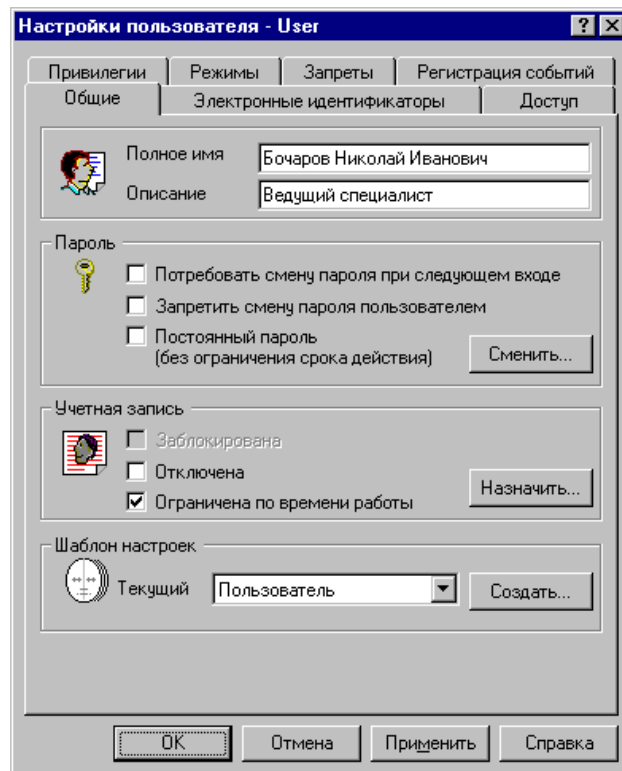
2. Отредактируйте список привилегий, предоставляемых пользователю. Для этого установите или удалите отметки из полей рядом с названиями привилегий.

3. Завершив настройку, нажмите кнопку "Применить".

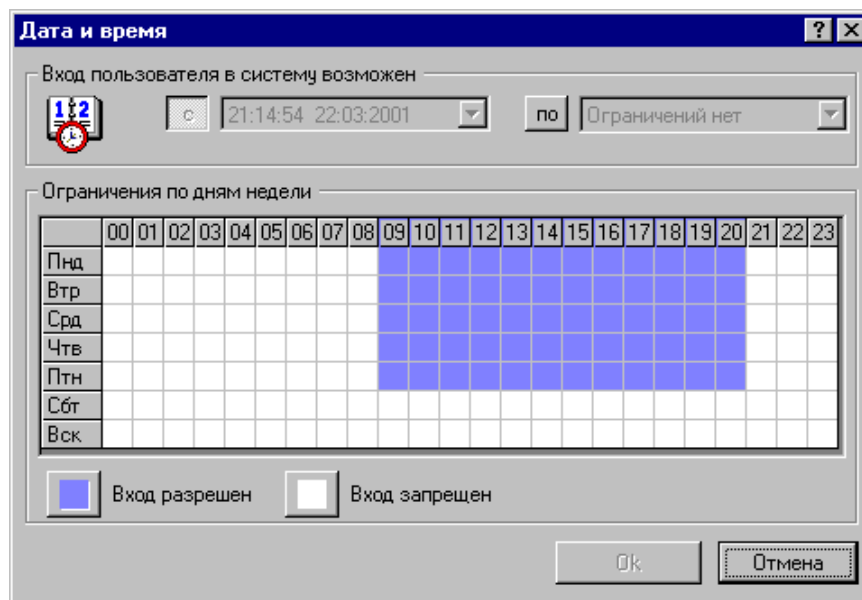


#### Управление состоянием учетных записей

1. Перейдите к вкладке "Общие".



2. Ведите полное имя пользователя и его описание.
3. Ограничьте его работу по времени.
4. Нажмите кнопку "Назначить" для вызова на экран диалога "Дата и время".



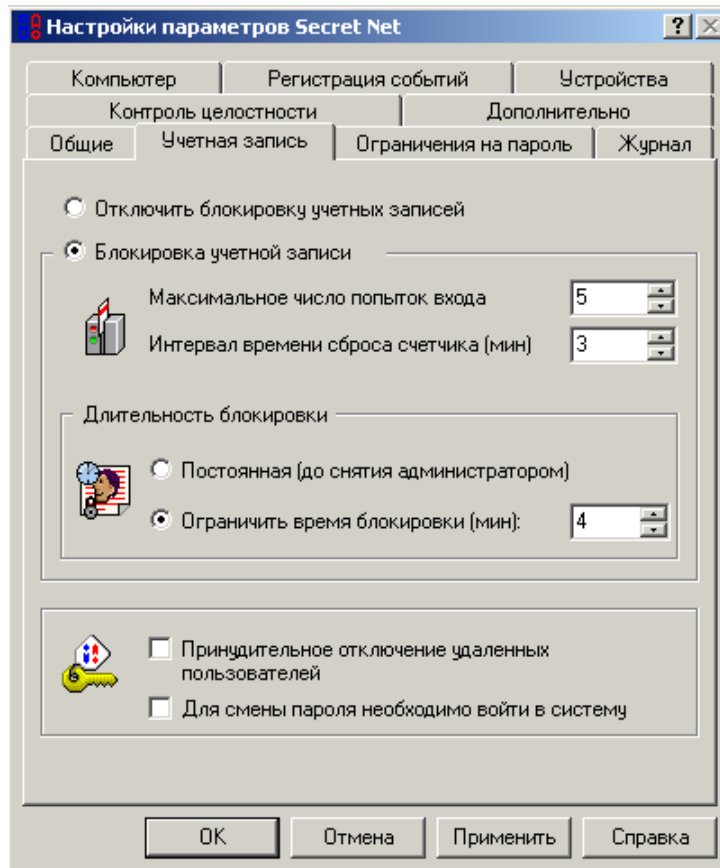
3. Укажите разрешенный период работы и составьте нужное расписание.
4. Нажмите кнопку "ОК" в диалоге "Дата и время".
5. Нажмите кнопку "Применить" в окне настройки свойств пользователя.

## Настройка механизмов контроля входа

### Учетные записи и пароли

#### Параметры блокировки учетной записи

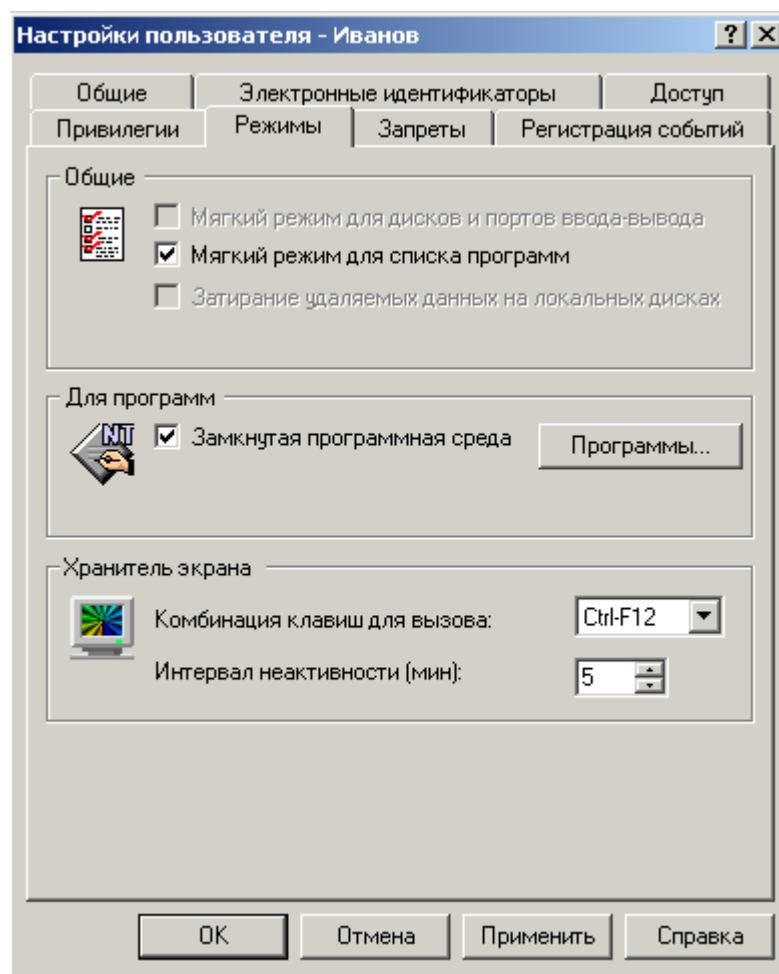
1. Вызовите окно настройки общих параметров и перейдите к вкладке "Учетная запись".
2. Установите максимальное число попыток входа = 5;
3. Интервал времени сброса счетчика = 3 мин.;
4. Ограничить время блокировки 4 мин.;
5. Нажмите кнопку "ОК" или "Применить".



#### Временная блокировка компьютера

1. Откройте окно управления общими параметрами.
2. Установите в поле "Максимальный период неактивности" предельное значение интервала неактивности, действующего для всех пользователей компьютера. Индивидуальные значения периода неактивности устанавливаются в границах этого диапазона и не могут его превышать.

3. Нажмите на кнопку "ОК" или "Применить".
4. В программе "Проводник" выберите пользователя "User", вызовите на экран окно настройки свойств и перейдите к вкладке "Режимы".
5. В поле "Комбинация клавиш для вызова" выберите комбинацию "горячих" клавиш Ctrl-F12.
6. Установите значение интервала неактивности = 5 мин.
7. Нажмите кнопку "Применить".



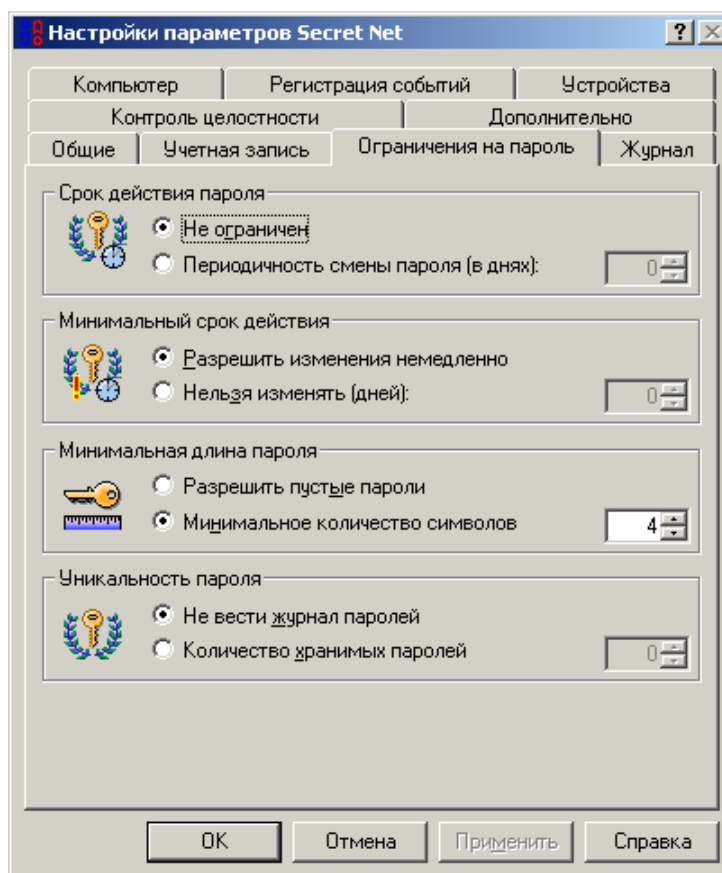
### Управление режимами использования пароля

#### Общие ограничения на пароль

1. Вызовите окно настройки общих параметров и перейдите к диалогу "Ограничения на пароль".
2. Установите неограниченный срок действия пароля.
3. Разрешите изменять немедленно минимальный срок действия пароля.
4. Минимальное количество символов в длине пароле установите 4.

5. Установите не вести журнал паролей.

6. Нажмите кнопку "Применить".

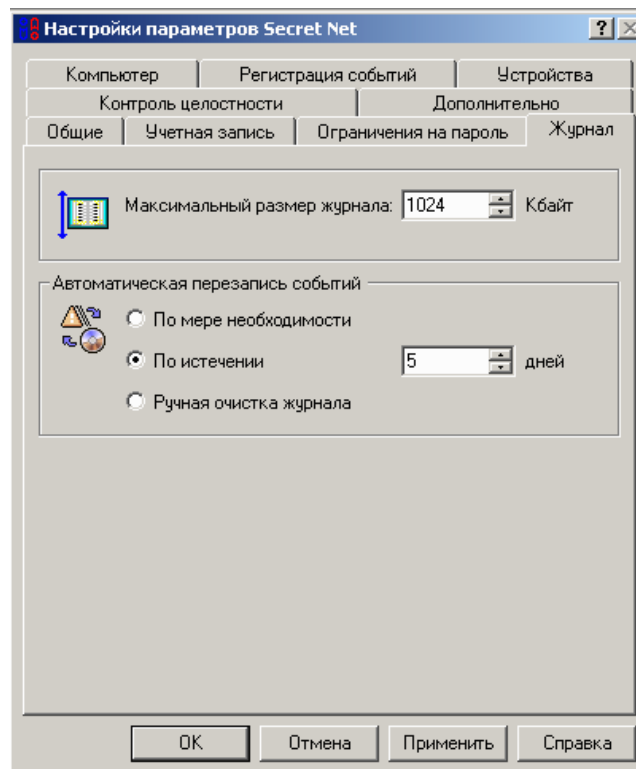


## Настройка механизмов контроля и регистрации

### Регистрация событий

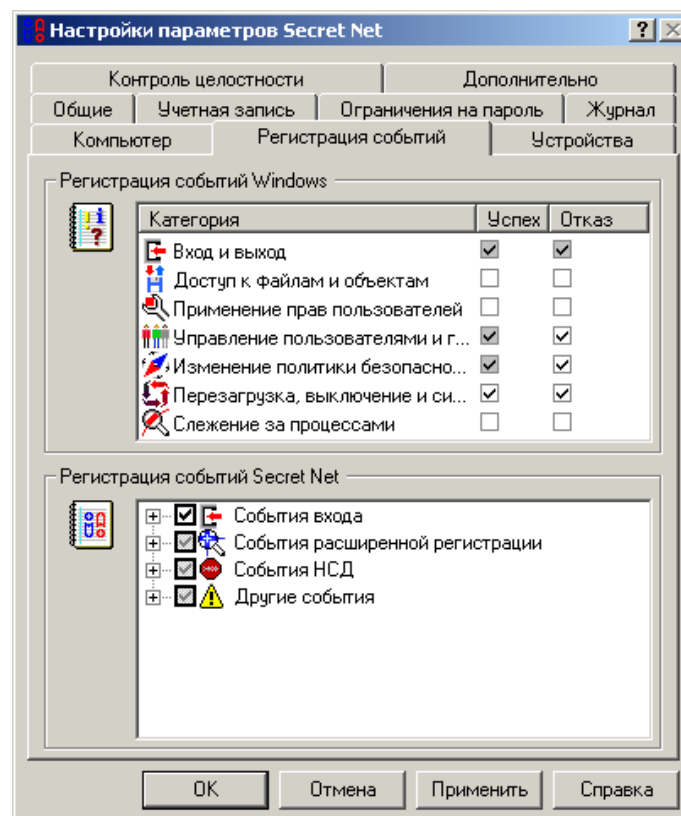
#### Настройка параметров журнала безопасности

1. Вызовите окно настройки общих параметров и перейдите к вкладке "Журнал".
2. Установите значение максимального размера журнала = 1024 Кбайт.
3. Установите механизм очистки журнала при его переполнении по истечении 5 дней.
4. Нажмите кнопку "Применить".



### Настройка общего перечня регистрируемых событий

1. В окне настройки общих параметров выберите вкладку "Регистрация событий".

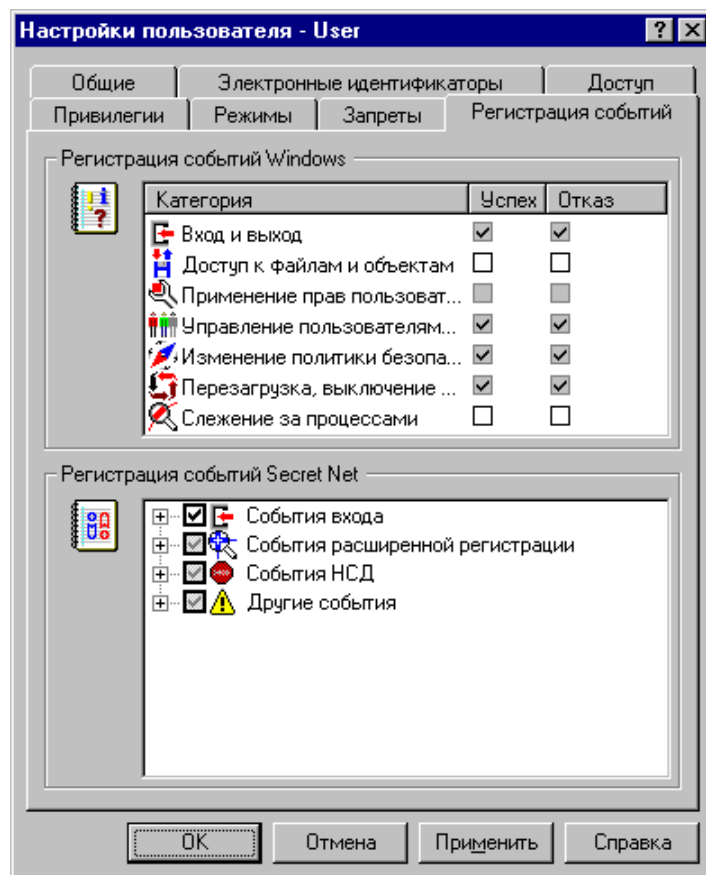


2. Отметьте категории событий Windows и события Secret Net, которые должны регистрироваться в журнале безопасности.

3. Нажмите кнопку "OK".

### Настройка персонального перечня регистрируемых событий

1. В программе "Проводник" выберите пользователя «User», вызовите на экран окно настройки свойств и перейдите к вкладке "Регистрация событий".

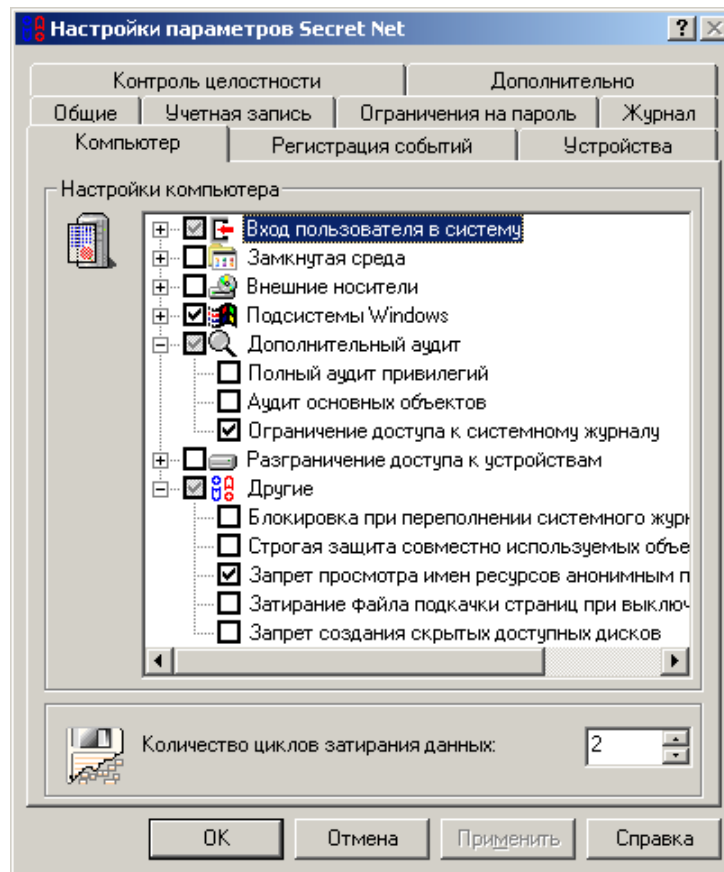


2. Отметьте категории событий Windows и события Secret Net, которые необходимо регистрировать в журнале безопасности.

3. Нажмите кнопку "OK" или "Применить".

### Дополнительный аудит

1. Вызовите окно настройки общих параметров и выберите диалог "Компьютер".



2. Установите отметки рядом с нужными параметрами, которые относятся к дополнительному аудиту и контролю переполнения журнала безопасности.

3. Нажмите кнопку или "Применить".

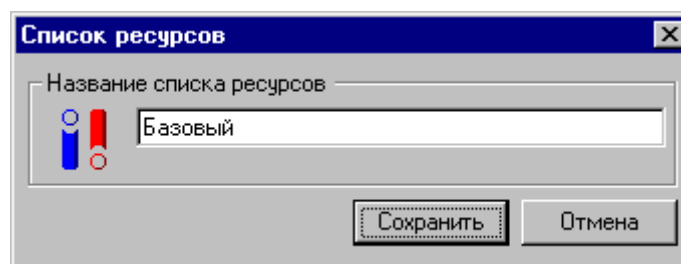
### Контроль целостности

### Настройка заданий контроля

### Списки ресурсов

1. Вызовите окно настройки общих параметров и перейдите к вкладке "Контроль целостности".

2. Нажмите кнопку "Создать список ресурсов".

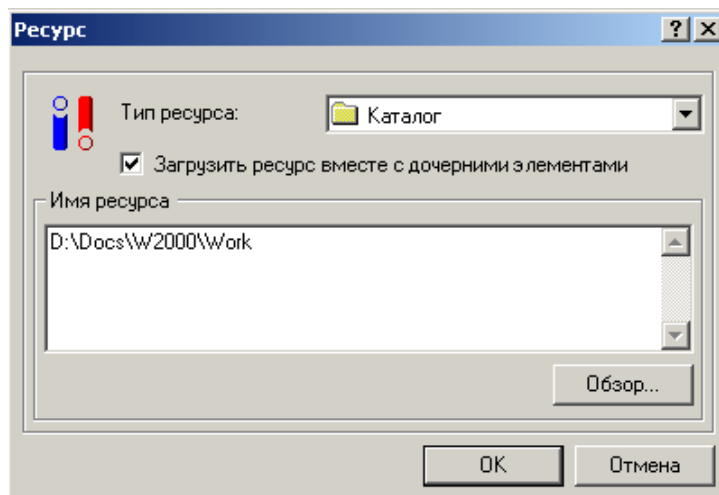


3. Введите название создаваемого списка ресурсов и нажмите кнопку "Сохранить". Вкладка закроется, а название нового списка появится в поле



верхней части диалога "Контроль целостности".

4. Нажмите кнопку "Ресурс" и в появившемся меню выберите пункт "Добавить ресурс".



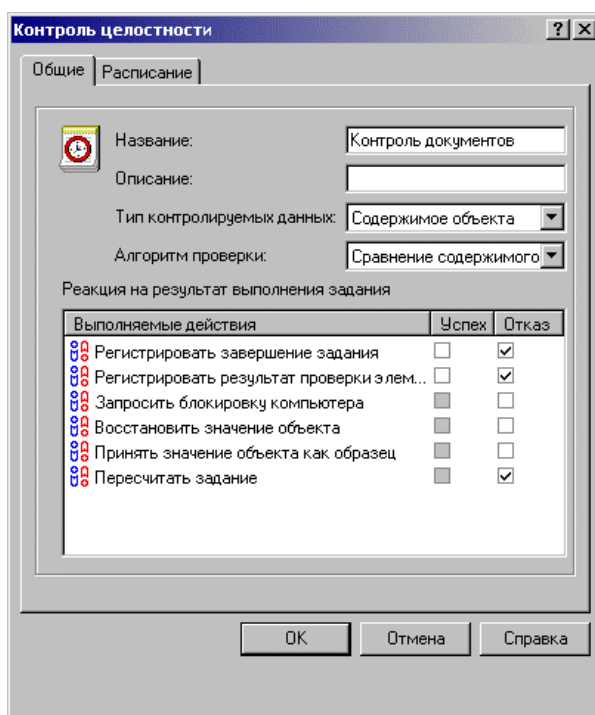
5. Укажите необходимые значения параметров.

6. Для добавления в список следующего ресурса, повторите процедуру, начиная с пункта 4.

7. Завершив формирование списка, нажмите кнопку "ОК" в окне настройки общих параметров.

### Задания

1. Во вкладке "Контроль целостности" выберите список ресурсов, нажмите кнопку "Задание" и выберите в меню команду "Добавить задание".



2. Во вкладке "Общие" введите название задания "Контроль документов" и комментарий к нему.

3. Выберите тип контролируемых данных "Содержимое объекта", т.е. контролируемые параметры (характеристики) ресурсов.

4. Отметьте с помощью переключателей желаемую реакцию системы защиты на результат выполнения контроля в случае успеха и в случае неудачи (отказа).

5. Перейдите к вкладке "Расписание" и составьте расписание контроля:

Контроль целостности

Общие Расписание

☒ При загрузке ☒ При входе  
☐ Срочный контроль ☐ При выходе

	Пнд	Втр	Срд	Чтв	Птн	Сбт	Вск
Январь							
Февраль							
Март							
Апрель							
Май							
Июнь							
Июль							
Август							
Сентябрь							
Октябрь							
Ноябрь							
Декабрь							

☒ Учет временных параметров  
Часы контроля, (0 - 23)  Интервал (мин):

OK Отмена Справка

4. В нижней части диалога в группе полей "Учет временных параметров" указать периодичность контроля в течение суток = 4 часам.

5. Нажмите кнопку "OK".

### Регистрация событий

1. Вызовите окно настройки общих параметров и перейдите к вкладке "Регистрация событий".

2. В категории событий Secret Net установите отметки у событий групп "События расширенной регистрации" и "События НСД".

Группа событий	Событие
События расширенной регистрации	Контроль целостности выполнен с исправлениями
	Ошибка при контроле целостности данных
	Восстановлено значение объекта
События НСД	Нарушена целостность объекта

3. Нажмите кнопку "ОК".

4. В программе "Проводник" выберите пользователя "User", вызовите на экран окно настройки свойств, перейдите к вкладке "Регистрация событий" и повторите действия 2-3.

#### Анализ нарушений и восстановление ресурсов

1. Найти в журнале безопасности событие, зафиксированное как нарушение целостности ресурса или несанкционированный доступ.

2. В диалоге "Контроль целостности" окна настройки общих параметров по описанию события и настройкам параметров задания найти ресурс, проверка которого выявила нарушение целостности.

3. Установить причину нарушения целостности ресурса.

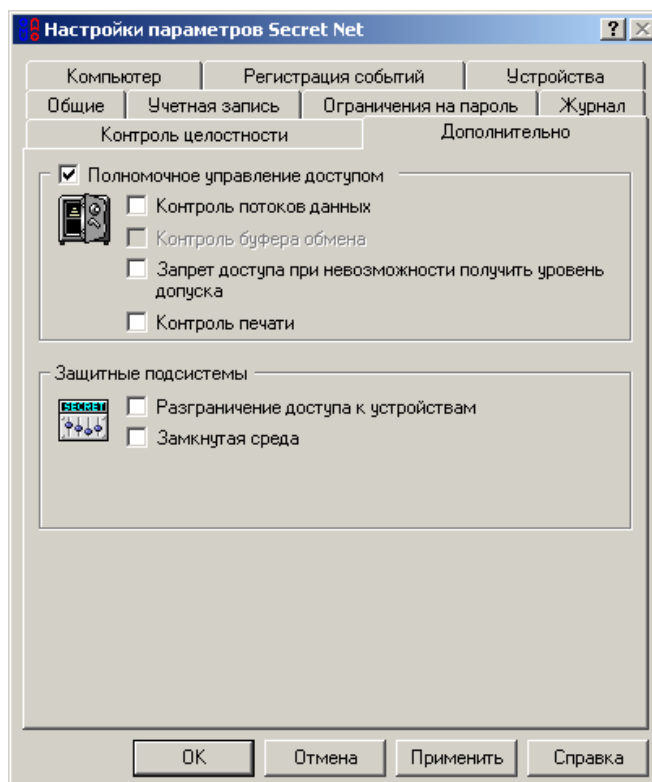
#### Настройка механизмов управления доступом и защиты ресурсов

##### Замкнутая программная среда

1. Откройте окно управления общими параметрами и перейдите к вкладке "Дополнительно".

2. Установите отметку в поле выключателя "Замкнутая среда" и нажмите кнопку "ОК".

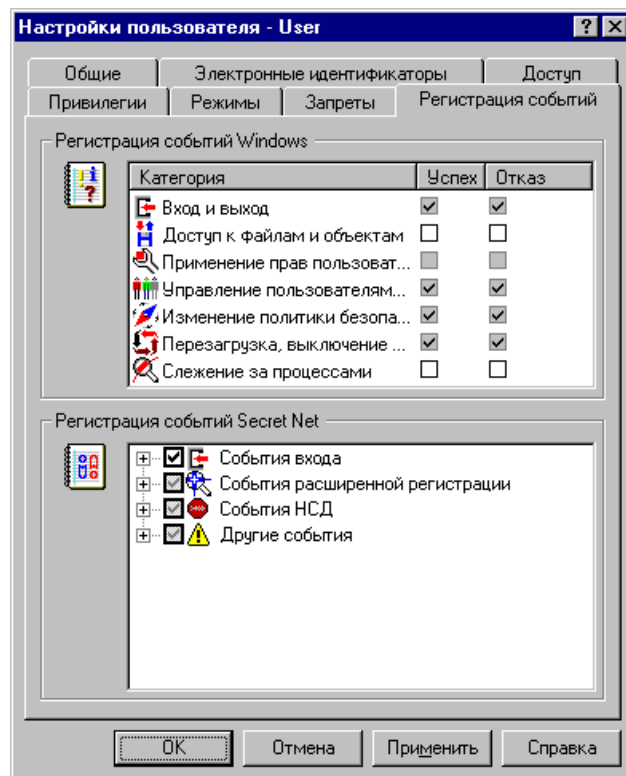
Для того чтобы установленные параметры вступили в силу, необходимо перезагрузить компьютер.



### Настройка регистрации событий

#### Персональный перечень

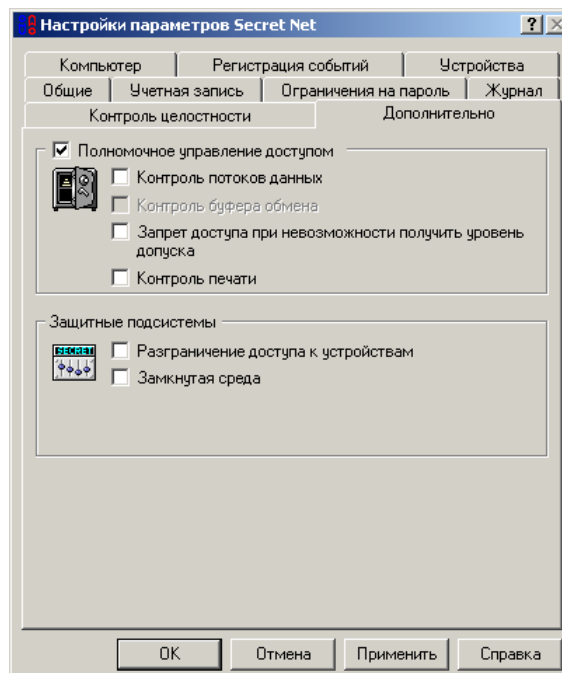
1. В программе "Проводник" выберите пользователя, вызовите на экран окно настройки свойств и перейдите к вкладке "Регистрация событий".
2. Отметьте события Secret Net, которые необходимо регистрировать для настройки замкнутой среды:
  - В группе "События расширенной регистрации": "Запуск программы";
  - В группе "События НСД": "Запрет запуска программы".
3. Нажмите кнопку "Применить".



## Доступ к дискам и портам

### Включение механизма разграничения доступа к дискам и портам

1. Откройте окно управления общими параметрами и перейдите к вкладке "Дополнительно".

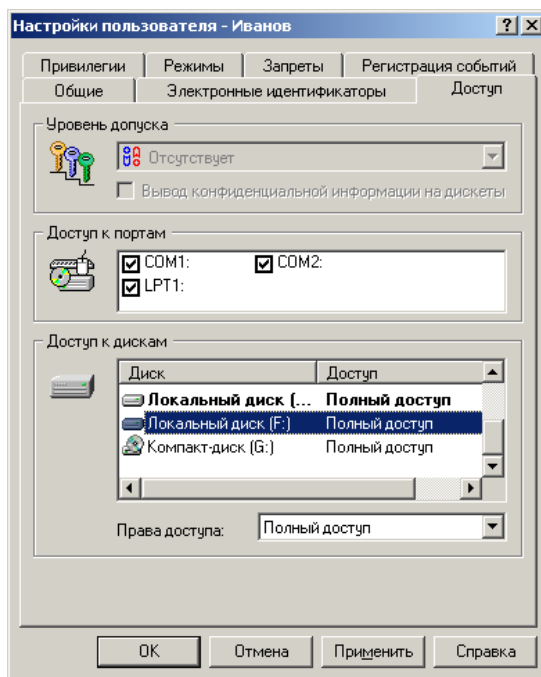


2. Установите отметку в поле выключателя "Разграничение доступа к устройствам" и нажмите кнопку "Применить".

Для того чтобы установленные параметры вступили в силу, необходимо перезагрузить компьютер.

### Предоставление прав доступа

1. В программе "Проводник" выберите пользователя "User", вызовите на экран окно настройки свойств и перейдите к вкладке "Доступ".



2. Установите отметку в списке "Доступ к портам".

3. Установите отметку "Доступ к дискам".

4. Установите отметку "Права доступа" для дисков.

5. Нажмите кнопку "ОК".

### **Контрольные вопросы:**

1. Перечислите основные компоненты Secret Net?

2. Перечислите отличительные особенности СЗИ НСД Secret Net от системы защиты Windows?

3. Каким образом можно настроить механизм регистрации событий?

4. Каким образом можно настроить контроль целостности?

5. Каким образом можно настроить задания контроля целостности?

6. Что означает термин «замкнутая программная среда»?

7. Какие аппаратные средства контроля входа в систему поддерживает Secret Net?

## СПИСОК ЛИТЕРАТУРЫ

1. Нестеров С.А. Информационная безопасность: Учебник и практикум / С. А. Нестеров. - М: Издательство Юрайт, 2018. - 321 с.
2. Лось Б.А. Криптографические методы защиты информации: Учебник / А. Б. Лось. - 2-е изд. - М : Издательство Юрайт, 2018. - 473 с.
3. Щеглов Ю.А. Защита информации: основы теории: Учебник / А. Ю. Щеглов. - М: Издательство Юрайт, 2018. - 309 с.
4. Казарин О.В. Программно-аппаратные средства защиты информации. защита программного обеспечения: Учебник и практикум / О. В. Казарин. – М.: Издательство Юрайт, 2018. - 312 с.
5. Казарин О.В. Надежность и безопасность программного обеспечения: учеб. пособие для бакалавров и магистратуры / О. В. Казарин, И. Б. Шубинский. - М.: Юрайт, 2018. - 342 с.
6. Шаньгин В.Ф. Информационная безопасность и защита информации / В. Ф. Шаньгин. – М.: ДМК Пресс, 2017. - 702 с.
7. Мельников В.П. Защита информации: учебник / В.П. Мельников, А.И. Куприянов, А.Г. Схиртладзе – М.: Академия, 2014. – 304 с.
8. Платонов В.В. Программно-аппаратные средства защиты информации: учеб. для студ. вузов / В. В. Платонов, 2013. – М.: Издательский центр «Академия», - 336 с.
9. Хорев П.Б. Программно-аппаратная защита информации: учеб. пособие / П. Б. Хорев. М.: Форум, 2012 - 352 с.
10. Аникин И.В. Методы и средства защиты компьютерной информации: лабораторный практикум: учеб. пособие для студ. вузов / И. В. Аникин, В. И.

Глова; Мин-во образования и науки РФ, ФГБОУ ВПО КНИТУ-КАИ им. А.Н. Туполева. - Казань: Изд-во КГТУ им. А.Н. Туполева, 2011. - 131 с.

11. Проскурин В.Г. Программно-аппаратные средства обеспечения информационной безопасности. Защита в операционных системах: учеб. пособие для вузов / В.Г. Проскурин. М.: Радио и связь, 2000. - 168 с.
12. Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа / А.Ю. Щеглов; под ред. М.В. - СПб: Наука и техника. 2004. - 384 с.



## СОДЕРЖАНИЕ

<b>Введение .....</b>	<b>5</b>
<b>1. Защита программного обеспечения от несанкционированного копирования .....</b>	<b>6</b>
<b>2. Защита программного обеспечения от несанкционированного исследования.....</b>	<b>12</b>
<b>3. Защита автоматизированных систем от несанкционированного доступа.....</b>	<b>25</b>
<b>4. Лабораторные работы.....</b>	<b>39</b>
4.1. Лабораторная работа № 1 .....	39
4.2. Лабораторная работа № 2 .....	60
4.3. Лабораторная работа № 3 .....	66
4.4. Лабораторная работа № 4.....	86
4.5. Лабораторная работа № 5 .....	103
4.6. Лабораторная работа № 6 .....	110
4.7. Лабораторная работа № 7 .....	121
4.8. Лабораторная работа № 8 .....	125
<b>Список литературы.....</b>	<b>143</b>

*ДЛЯ ЗАМЕТОК*

**Игорь Вячеславович Аникин**

**ПРОГРАММНО-АППАРАТНЫЕ СРЕДСТВА ЗАЩИТЫ  
ИНФОРМАЦИИ**

**Учебно-методическое пособие**

**Редактор Яруллина Г.Х.**

**Техническое редактирование Гапсаламов А.**

**Сдано в набор 25.08.2018. Подписано к печати 28.08.2018.**

**Формат 60х84 <sup>1/16</sup>. Бумага офсетная.**

**Гарнитура «Таймс». Печать цифровая.**

**Усл. печ. л. 9,30. Печ. л. 10. Тираж 500 экз. Заказ № 117.**

---

420111, Казань, Дзержинского, 9/1. Тел.: 8-917-264-84-83  
Отпечатано в РИЦ «Школа».