поэтому, в *EFS* введена роль агента восстановления. Он может расшифровать зашифрованные другими пользователями данные.

Реализуется это примерно следующим образом. *Файл* шифруется с помощью симметричного криптоалгоритма на сгенерированном системой случайном ключе (назовем его **K1** ). *Ключ* **K1** шифруется на открытом ключе пользователя, взятом из сертификата, и хранится вместе с зашифрованным файлом. Также хранится **K1**, зашифрованный на открытом ключе агента восстановления. Теперь либо *пользователь*, осуществлявший *шифрование*, либо *агент* восстановления могут *файл* расшифровать.

При настройке *по* умолчанию роль агента восстановления играет встроенная учетная *запись* администратора (локального, если *компьютер* не в домене, или доменная).

## Задание 2

Зайдите в систему под встроенной учетной записью администратора и расшифруйте папку. То, какой *пользователь* является агентом восстановления, задается с помощью групповых политик. Запустим оснастку **Group Policy Management**. В политике домена найдем группу **Public Key Policies** и там **Encrypting File System**, где указан сертификат агента восстановления (рис. 7.3). Редактируя политику (*пункт* **Edit** в контекстном *меню*, далее **Policies —> Windows Settings —> Security Settings —> Public Key Policies —> Encrypting File System** ), можно отказаться от присутствия агентов восстановления в системе или наоборот, указать более одного агента (рис. 7.4).

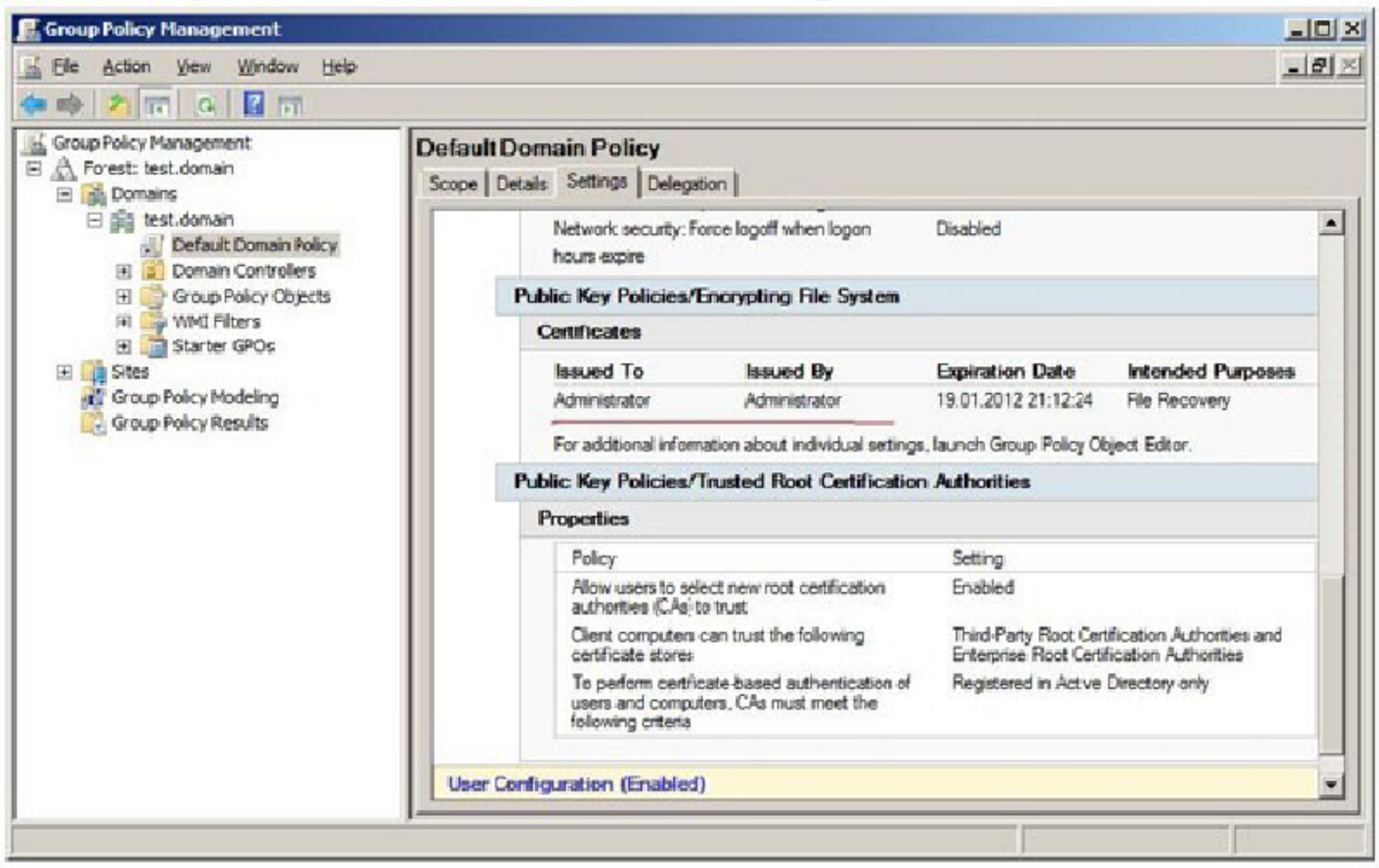



Рис. 7.3. Агент восстановления

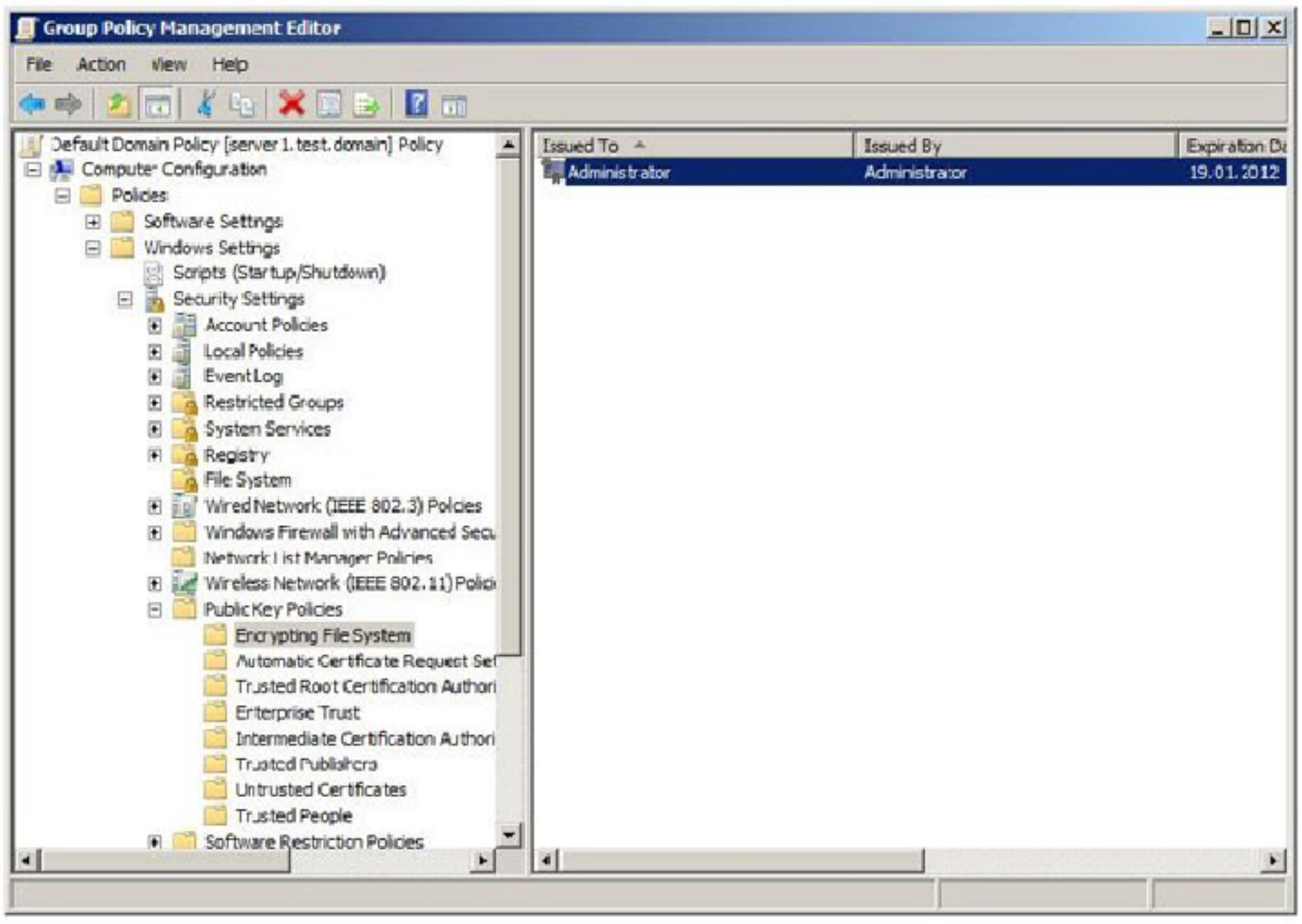



Рис. 7.4. Изменение агента восстановления

**Задание 3**

1. Отредактируйте политику таким образом, чтобы убрать из системы агента восстановления (удалите в *политике сертификат*). Выполнив команду "gpupdate /force" (меню **Start—>run—> gpupdate /force** ) примените политику.

2. Повторив действия из предыдущих заданий, убедитесь, что теперь только тот пользователь, который зашифровал файл, может его расшифровать.

3. Теперь вернем в систему агента восстановления, но будем использовать новый сертификат. В редакторе политик находим политику **Encrypting File System** и в контекстном меню выбираем **Create Data Recovery Agent**. Это приведет к тому, что пользователь **Administrator** получит новый сертификат и с этого момента сможет восстанавливать шифруемые файлы.

Теперь рассмотрим, как можно предоставить *доступ* к зашифрованному файлу более чем одному пользователю. Такая настройка возможна, но делается она для каждого файла в отдельности.

В свойствах зашифрованного файла откроем окно с дополнительными параметрами, аналогичное представленному на рис. 7.1 для папки. Если нажать кнопку **Details**, будут выведены подробности относительно того, кто может получить *доступ* к файлу. На рис. 7.5 видно, что в данный момент это *пользователь* **User1** и *агент* восстановления **Administrator**. Нажав кнопку **Add** можно указать сертификаты других пользователей, которым предоставляется *доступ* к файлу.

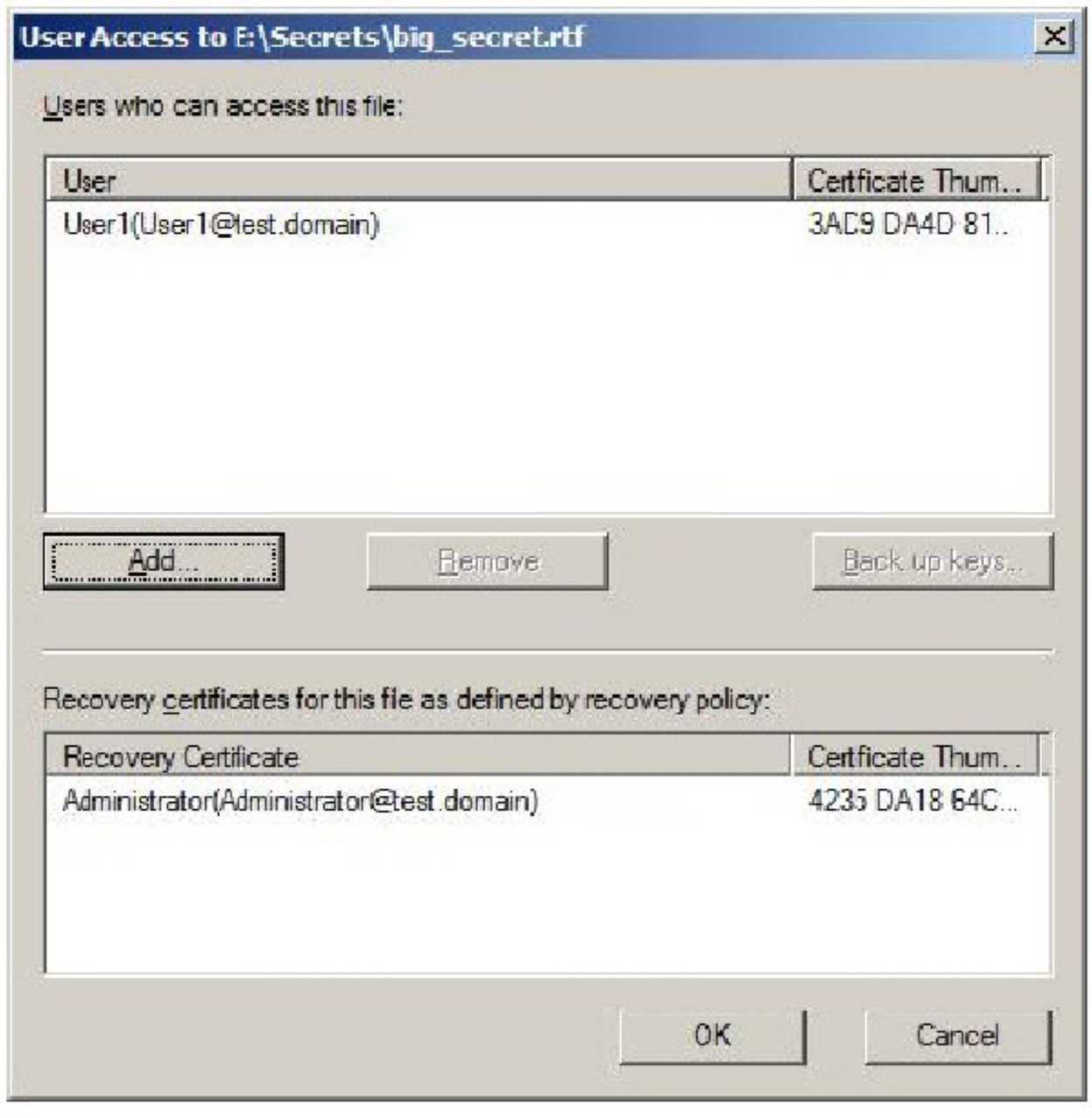



Рис. 7.5. Данные о пользователях, которые могут расшифровать файл

**Задание 4**

Зашифруйте *файл*. Предоставьте другому пользователю, не являющемуся агентом восстановления, возможность также расшифровать данный *файл*. Проверьте работу выполненных настроек.

Контроль выполнения задания производится по окончании занятия и на консультациях в форме защиты выполненной работы, предоставленной в электронном и в бумажном виде в форме «Отчет по лабораторной работе».

***Работа с литературой:***

| Рекомендуемые источники информации (№ источника) | | | |
|---|---|---|---|
| Основная | Дополнительная | Методическая | Интернет-ресурсы |
| 1-2 | 1-2 | 1-3 | 1-3 |

***Оценочные средства:*** отчет.

## 8. Лабораторная работа № 8. Защита программ от НСИ с помощью USB-ключей.

**Цель работы:**

Ознакомиться с устройством USB ключей HASP. Программное обеспечение, поставляемое с устройством. Научиться защищать программы от не санкционированного использования с помощью программного обеспечения производителя ключей HASP.

### 1.Теоретическая часть

**Проблема защиты программ**

Пиратство в сфере информационных технологий и в части нелегального использования чужого программного обеспечения, становится доходным и динамично развивающимся полулегальным бизнесом с такими естественными составляющими, как производство, логистика, дистрибуция, прямые продажи и пр.

Приобретение и использование контрафактной (т. е. незаконной, ворованной) продукции, и тем более кража программных разработок и нажива на них, — это серьёзные преступления, которые наносят гигантский ущерб правообладателям. В связи с этим всё более актуальными становятся технологические вопросы защиты ПО.

Вопрос о защите программного обеспечения появился практически тогда же, когда появилось и оно само, и приобрёл особый вес с распространением персональных компьютеров. Всеобщая компьютеризация, помимо безусловных плюсов, таких как новые отрасли экономики, новые рынки, доступность богатейших библиотек, возможность общения, самореализации т. д., породила и ряд очевидных побочных эффектов, одним из которых является теневой бизнес по изготовлению и продаже нелицензионного программного обеспечения.

Незаконно растиражированные программы свободно продаются на рынках, и, покупая всё это, пользователь поощряет процветание и развитие пиратского бизнеса. Чего же лишается потребитель, покупая «нелицензионки»?

Гарантий качества продукта, технической документации и поддержки, бесплатных или льготных обновлений более совершенных версий ПО и ряда внешних атрибутов (фирменного оформления, вкладок и пр.). Впрочем, последнее, естественно, не может сформировать комплекс вины у обладателя пиратского диска. Контроль выполнения задания производится по окончании занятия и на консультациях в форме защиты выполненной работы, предоставленной в электронном и в бумажном виде в форме «Отчет по лабораторной работе».

**Пути и методы решения проблемы**

Что на этом фоне остаётся делать программистам? Позаботиться о защите собственноручно написанных программ самим, не дожидаясь, пока представителей нелегального бизнеса вдруг заест совесть и они выйдут из тени.

На сегодняшний день есть несколько способов защиты своих программных продуктов от пиратов.

Подавляющее большинство создателей ПО используют различные программные модули, контролирующие доступ пользователей с помощью ключей активации, серийных номеров и т. д. Но невысокая стоимость — это, наверное, единственный плюс таких решений для борьбы со взломом программ. Интернет изобилует разного рода «крэками»1, патчами и эмуляторами, позволяющими обходить защиту и блокирующими запрос паролей или ключей активации. О серийных ключах говорить и вовсе не приходится. Легальный пользователь, купивший лицензионную программу, может просто обнародовать

код, по каким-то своим личным весьма альтруистическим соображениям, и не пройдёт и пары часов, как им воспользуется не одна сотня приверженцев всего бесплатного.

Эти очевидные недостатки привели к созданию аппаратной защиты программного обеспечения в виде электронного ключа. Первый ключ для защиты ПО на персональном компьютере был создан в начале 1980-х годов и с тех пор претерпел значительные изменения. Сегодня он представляет собой компактное USB-устройство (менее 4 см в длину), позволяющее шифровать данные, используя либо публичный, либо секретный алгоритмы.

Секретные алгоритмы разрабатываются самим производителем средств защиты, в том числе и индивидуально для каждого заказчика. Главным недостатком использования таких алгоритмов является невозможность оценки стойкости. С уверенностью сказать, насколько надёжен алгоритм, можно было лишь постфактум: взломали или нет. Кроме того, знание алгоритма даёт возможность создания эмулятора — программного продукта, полностью выполняющего все функции аппаратного устройства.

Публичный алгоритм, или «открытый исходник», обладает криптостойкостью несравнимо большей. Брюс Шнайер2 в своей статье «Открытые исходники и безопасность» объясняет преимущества публичных алгоритмов прежде всего тем, что они проверены не случайными людьми, а рядом экспертов, специализирующихся на анализе криптографии. «Прежде чем алгоритм будет признан надёжным, он должен быть изучен многими экспертами в течение ряда лет. И это — сильный довод в пользу криптографии с открытым кодом.

Поскольку единственный способ обрести уверенность в алгоритме — это предоставить его для изучения экспертам, а единственное условие, при котором они будут тратить время на проверку, — возможность публиковать результаты своих исследований, алгоритмы должны публиковаться», — пишет автор. Таким образом, становится очевидным, что «открытые исходники» разрабатываются с учётом того, что будут доступны всем желающим. Следовательно, риска их публикации нет.

Публичный криптоалгоритм кардинально усложняет задачу взлома ПО, сводя её к криптоанализу и вычислению ключей шифрования методом полного перебора. Самыми известными открытыми алгоритмами на сегодняшний день являются DES, Triple DES, Blowfish, RC2, CAST и, наверное, один из самых «продвинутых» на сегодняшний день, общепризнанный американский стандарт шифрования AES (Advanced Encryption Standard), пришедший на смену двум первым.

Для эффективного использования системы защиты HASP Вам следует ознакомиться с принципами её работы и терминологией, изложенными в настоящей главе. Если Вы собираетесь использовать ключ NetHASP.

*Ключи HASP*

Ключи HASP бывают следующих типов:

Локальные ключи – это ключи HASP, предназначенные для автономных (не являющихся частью сети) компьютеров. К этой категории относятся все ключи, кроме NetHASP. Демонстрационные ключи. В каждый комплект разработчика HASP входит демонстрационный ключ HASP (также демо-ключ). Демо-ключи обладают всеми возможностями ключей своего класса, но имеют стандартный демонстрационный код разработчика. Использовать демо-ключи для защиты программного обеспечения нельзя, поскольку они доступны для любого человека. Эти ключи лучше всего использовать для оценки системы защиты HASP.

*Память HASP*

Все ключи HASP, за исключением HASP4 без памяти, имеют перезаписываемую память. Используя память HASP, вы можете делать следующее: Управлять доступом к различным программным модулям и пакетам программ. Назначить каждому пользователю Ваших программ уникальный номер. Сдавать программы в аренду и распространять их

демо-версии с ограничением количества запусков. Хранить в ключе пароли, фрагменты кода программы, значения переменных и другую важную информацию.

*Тип ключа Размер памяти*
HASP4 без памяти Нет
HASP4 M1 112 байт
HASP4 M4 496 байт
HASP4 Time 512 байт
Все ключи NetHASP 496 байт
Идентификатор HASP

У каждого ключа HASP с памятью имеется уникальный опознавательный номер (ID-номер), или идентификатор, доступный для контроля защищёнными приложениями. Идентификаторы позволяют вам различать пользователей приложений. Проверяя в программе идентификатор HASP, Вы можете предпринимать те или иные действия в зависимости от наличия конкретного ключа. Вы не можете заказывать ключи HASP с заранее заданными идентификаторами. Они назначаются псевдослучайным образом в процессе изготовления ключей, чем гарантируется защита от повтора.

*Способы защиты HASP*

Система HASP позволяет защищать программное обеспечение двумя различными способами: Утилитой HASP Envelope (оболочка) HASP API (Application Programming Interface – программный интерфейс приложения) Оболочка HASP Использование HASP Envelope является основным способом защиты. Исполняемый файл заключается в защитную программную оболочку, кодирующую файл, и обладающую такими свойствами, как распознавание ключа и антиотладка. Оболочка не позволяет файлу выполняться без соответствующего ключа HASP.

Защита оболочкой производится быстро и без особых усилий. В то же время, она достаточно надёжна, так как делает отладку и дизассемблирование Ваших программ практически невозможными. Для защиты оболочкой исходные тексты программ не требуются.

Программный интерфейс пользователя HASP (API) Если у Вас имеются исходные тексты программы, которую надо защитить, то Вы можете пристыковать к ней модуль HASP API – объектный файл или библиотеку DLL.

Поскольку модуль API сам по себе защищён и зашифрован, этот метод обеспечивает высокую степень защиты. API позволяет обращаться к ключу.

## 8.2.Порядок выполнения работы

**Установка программного обеспечения HASP**

1. В списке оборудования компьютера убедиться в отсутствии устройств компании Aladdin (eToken,

HASP). Если эти устройства присутствуют в списке оборудования – удалить их.

2. Удалить, если присутствует, программное обеспечение eToken и HASP.
3. Подключиться к компьютеру Centurion, открыть ресурс Q и войти в каталог setup.
4. Запустить программу setup.exe.

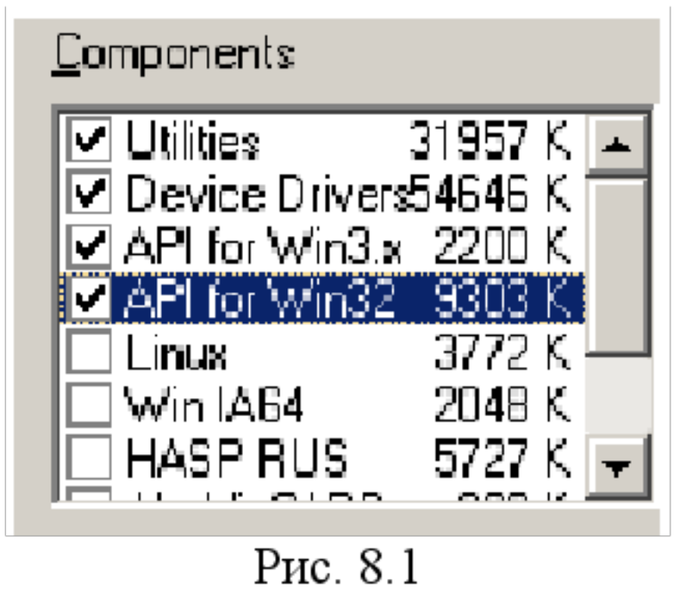



Рис. 8.1

6. Следуя указаниям программы установки установить на компьютере программное обеспечение системы HASP. Установить компоненты помеченные ниже.

6. Подключить HASP к разъему USB компьютера.

a) В случае правильной установки в электронном ключе HASP должен загореться световой индикатор.

b) Если индикатор не загорается, обновить драйвер устройства вручную.

7. В списке оборудования компьютера найти подключенное устройство.

**Работа с программным обеспечением HASP**

1. Запустить программу Aladdin DiagnostiX провести тестирование электронного ключа HASP.

2. Запустить демонстрационную программу – HASP Demo for Win16 . Описать работу этой программы.

3. Запустить демонстрационную программу – HASP Demo for Win32 . Описать работу этой программы.

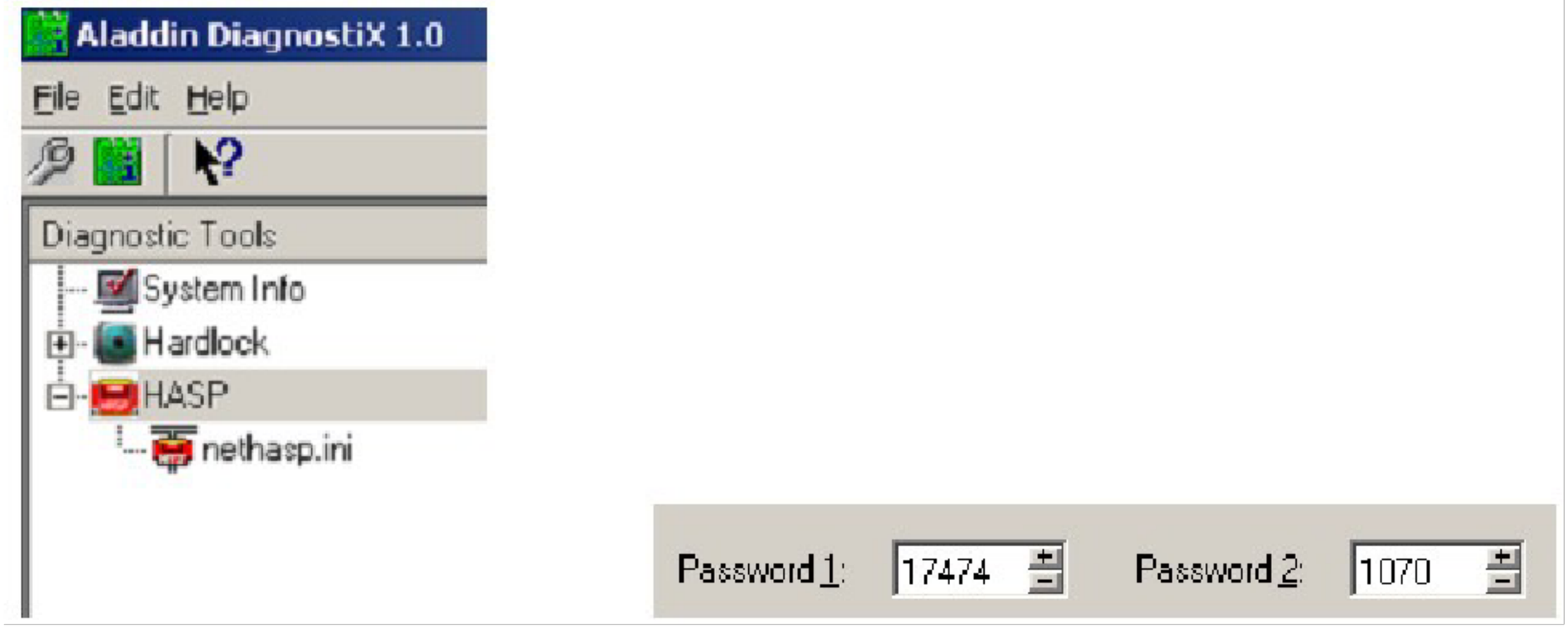



Рис. 8.2.

**Защита приложений с помощью HASP.**

Используя инструмент HASP Envelope защитить избранное приложение от не санкционированного запуска.

Санкционированный запуск – при наличии HASP ключа в системе.

Несанкционированный запуск – при отсутствии HASP ключа в системе.

1. Создать папку: «Защищенные приложения»
2. Скопировать в нее 3 исполняемых файла, например программу калькулятор – calc.exe
3. С помощью HASP Envelope защитить выбранные программы от несанкционированного запуска.
4. Запустить программу.
5. Закрыть программу
6. Вытащить HASP из разъема USB
7. Запустить защищенную программу.

Попробовать варианты защиты нескольких программ с одним ключом HASP.

## 8.3. Содержание отчета

Форма отчётности - подробное описания проделанной работы по каждому шагу в электронном документе (Microsoft Word). Привести структуру ключевой дискеты. Содержание файла открытых и закрытых ключей.

## 8.4. Контрольные вопросы

1. Рассказать об утилите HASP Envelope
2. Рассказать об утилите тестирования HASP
3. Рассказать об утилите HASPEdit
4. Что такой идентификатор HASP
5. Как шифруются и дешифруются данные для распознавания ключа HASP.

Контроль выполнения задания производится по окончании занятия и на консультациях в форме защиты выполненной работы, предоставленной в электронном и в бумажном виде в форме «Отчет по лабораторной работе».

***Работа с литературой:***

| Рекомендуемые источники информации (№ источника) | | | |
|---|---|---|---|
| Основная | Дополнительная | Методическая | Интернет-ресурсы |
| 1-2 | 1-2 | 1-3 | 1-3 |

***Оценочные средства:*** отчет.

## 9.Лабораторная работа № 9. Sandbox – файловые антивирусы.

**Цель работы:**

Получение теоретических и практических навыков работы с песочницами и файловыми антивирусами.

### 9.1.Теоретическая часть

Вредоносная программа (англ. malware, malicious software — «злонамеренное программное обеспечение») — любое программное обеспечение, предназначенное для получения несанкционированного доступа к вычислительным ресурсам самой ЭВМ или к информации, хранимой на ЭВМ, с целью несанкционированного использования ресурсов ЭВМ или причинения вреда (нанесения ущерба) владельцу информации, и/или владельцу ЭВМ, и/или владельцу сети ЭВМ, путем копирования, искажения, удаления или подмены информации. Многие антивирусы считают крэки, кейгены и прочие программы для взлома вредоносными программами, или потенциально опасными.

Песочница (англ. Sandbox) - это ограниченная среда в вашей системе для исполнения гостевых программ без доступа к главной операционной системе. Это закрытое для доступа извне виртуальное пространство, в котором можно работать с программным обеспечением без изменения системных файлов.

Песочница (от англ. Sandbox, схожие понятия — англ. honeypot, англ. fishbowl) - в компьютерной безопасности, механизм для безопасного исполнения программ. Песочницы часто используют для запуска непротестированного кода, непроверенного кода из неизвестных источников, а также для запуска и обнаружения вирусов.

Песочница — в компьютерной безопасности специально выделенная среда для безопасного исполнения компьютерных программ.

Песочница обычно представляет собой жёстко контролируемый набор ресурсов для исполнения гостевой программы — например, место на диске или в памяти. Доступ к сети, возможность сообщаться с главной операционной системой или считывать информацию с устройств ввода обычно либо частично эмулируют, либо сильно ограничивают. Песочницы представляют собой пример виртуализации. Повышенная безопасность исполнения кода в песочнице зачастую связана с большой нагрузкой на систему — именно поэтому некоторые виды песочниц используют только для неотлаженного или подозрительного кода.

Cuckoo Sandbox — система для автоматического исследования вредоносного ПО, эксплоитов, вредоносных скриптов, документов, архивов и ссылок. Система способна проверять документы pdf, doc, xls, rtf, скрипты Python, JS, DLL библиотеки, бинарники, jar и многое другое.

Файловый Антивирус — компонент модуля Защита компьютера, контролирующий файловую систему компьютера. Он запускается при старте операционной системы, постоянно находится в оперативной памяти компьютера и проверяет все открываемые, сохраняемые и запускаемые файлы. Каждый файл, к которому вы обратитесь, будет перехвачен Файловым Антивирусом и проверен на присутствие известных вирусов.

Clam AntiVirus — пакет антивирусного ПО, работающий во многих операционных системах, включая Unix-подобные ОС, OpenVMS, Microsoft Windows и Apple Mac OS X.

Главная цель Clam AntiVirus — интеграция с серверами электронной почты для проверки файлов, прикреплённых к сообщениям. В пакет входит масштабируемый многопоточный демон clamd, управляемый из командной строки сканер clamscan, а также модуль обновления сигнатур по Интернету freshclam.



### Задания к лабораторной работе

**Установите Cukoo Sandbox.**

**Примечание**

Если будут проблемы с установкой, воспользуйтесь документацией Cukoo sandboxhttp://docs.cuckoosandbox.org/en/latest/installation/

* Установка зависимостей

cd /tmp

apt-get update

apt-get install  git automake mongodb mingw32 dkms unzip wget python python-sqlalchemy python-bson python-pip python-dpkt python-jinja2 python-magic python-mysqldb python-gridfs python-libvirt python-bottle python-pefile python-chardet -y

apt-get install python-dev libxml2-dev libxslt1-dev libevent-dev libpcre3 libpcre3-dev zlib1g-dev libtool libpcre++-dev –y

apt-get install mariadb-server -y

pip install lxml

pip install cybox==2.0.1.4

pip install maec==4.0.1.0

pip install django

pip install py3compat

pip install pymongo

apt-get install ssdeep python-pyrex subversion libfuzzy-dev -y

svn checkout http://pyssdeep.googlecode.com/svn/trunk/ pyssdeep

cd pyssdeep

python setup.py build

python setup.py install

pip install pydeep

cd /tmp

wget https://github.com/plusvic/yara/archive/v2.1.0.tar.gz

tar xzf v2.1.0.tar.gz

cd yara-2.1.0

chmod +x build.sh

./build.sh

make install

cd yara-python

python setup.py build

python setup.py install

cd /tmp

wget http://distorm.googlecode.com/files/distorm3.zip

unzip distorm3.zip

cd distorm3

python setup.py build

python setup.py install

add-apt-repository ppa:pi-rho/security

apt-get update

apt-get install volatility

**Установка и настройка Virtualbox**:

wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add – sh -c 'echo "deb http://download.virtualbox.org/virtualbox/debian trusty contrib" >> /etc/apt/sources.list.d/virtualbox.list'

apt-get update
apt-get install virtualbox-4.3
cd /tmp
VBOX_LATEST_VERSION=$(curl http://download.virtualbox.org/virtualbox/LATEST.TXT)
wget http://download.virtualbox.org/virtualbox/${VBOX_LATEST_VERSION}/Oracle_VM_VirtualBox_Extension_Pack-{VBOX_LATEST_VERSION}.vbox-extpack
vboxmanage extpack install /tmp/Oracle_VM_VirtualBox_Extension_Pack-${VBOX_LATEST_VERSION}.vbox-extpack
cd /opt
wget http://dlc.sun.com.edgesuite.net/virtualbox/${VBOX_LATEST_VERSION}/VBoxGuestAdditions_${VBOX_LATEST_VERSION}.iso

**Установка Cuckoo Sandbox**:
useradd cuckoo
usermod -a -G vboxusers cuckoo
id cuckoo
cd /opt
wget http://downloads.cuckoosandbox.org/1.1/cuckoo_1.1.tar.gz
tar xzf cuckoo_1.1.tar.gz

**Настройка Cuckoo Sandbox**:
cd /opt/cuckoo
./utils/community.py --signatures --force

**Настройка БД**:
mysql -u root -p
> create database cuckoo;
> grant all privileges on cuckoo.* to cuckoo@localhost identified by 'cuck00pass' ;
> flush privileges;
> quit;

**Настраиваем cuckoo:**:
* Файл /opt/cuckoo/conf/cuckoo.conf
Включаем запись дампа памяти:
memory_dump = on
Настраиваем подключение к бд:
connection = mysql://cuckoo:cuck00pass\@localhost/cuckoo
Увеличиваем временные лимиты:
default = 240
critical = 1200
vm_state = 600
* Файл /opt/cuckoo/conf/memory.conf
Отключаем сохранение дампов памяти:
delete_memdump = yes
* Файл /opt/cuckoo/conf/processing.conf
Включаем анализ оперативной памяти:
memory = yes
* nano /opt/cuckoo/conf/virtualbox.conf
Меняем режим работы Virtualbox:
mode = headless

Меняем названия виртуальной машины с cuckoo1 на WindowsXP:
machines = WindowsXP
[WindowsXP]
label = WindowsXP
* Файл /opt/cuckoo/conf/reporting.conf
Включим импорт отчётов в MongoDB для работы веб интерфейса
[mongodb]
enabled = yes

**Примечание**

• На этом настройка Cuckoo закончена, теперь приступим к Virtualbox и гостевой ОС.

**Загрузка виртуальной ОС с сайта:**

wget https://az412801.vo.msecnd.net/vhd/VMBuild_20131127/VirtualBox/IE6_WinXP/Linux/IE6.WinXP.For.LinuxVirtualBox.sfx

chmod +x IE6.WinXP.For.LinuxVirtualBox.sfx

./IE6.WinXP.For.LinuxVirtualBox.sfx

vboxmanage import IE6\ -\ WinXP.ova --vsys 0 --unit 10 --disk=/root/VirtualBox\ VMs/WindowsXP/WindowsXP.vmdk --memory 1024 --vmname WindowsXP

**Настраиваем сеть**:

vboxmanage hostonlyif create

vboxmanage modifyvm "WindowsXP" --nic1 hostonly --hostonlyadapter1 vboxnet0 --nicpromisc1 allow-all --hwvirtex off --vtxvpid off

**Настраиваем общие папки**:

mkdir -p /opt/cuckoo/shares/setup

mkdir -p /opt/cuckoo/shares/WindowsXP

vboxmanage sharedfolder add "WindowsXP" --name "WindowsXP" --hostpath /opt/cuckoo/shares/WindowsXP --automount

vboxmanage sharedfolder add "WindowsXP" --name setup --hostpath /opt/cuckoo/shares/setup --automount --readonly

vboxmanage modifyvm "WindowsXP" --nictrace1 on --nictracefile1 /opt/cuckoo/shares/WindowsXP/dump.pcap

**Включаем доступ по RDP, порт можете указать любой**:

vboxmanage modifyvm "WindowsXP" --vrdeport 5000 --vrde on

**Примечание**

**На этом конфигурация виртуальных контейнеров полностью закончена, осталось настроить iptables, tcpdump.**:

iptables -A FORWARD -o eth0 -i vboxnet0 -s 192.168.56.0/24 -m conntrack --ctstate NEW -j ACCEPT

iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -A POSTROUTING -t nat -j MASQUERADE

sysctl -w net.ipv4.ip_forward=1

setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump

getcap /usr/sbin/tcpdump

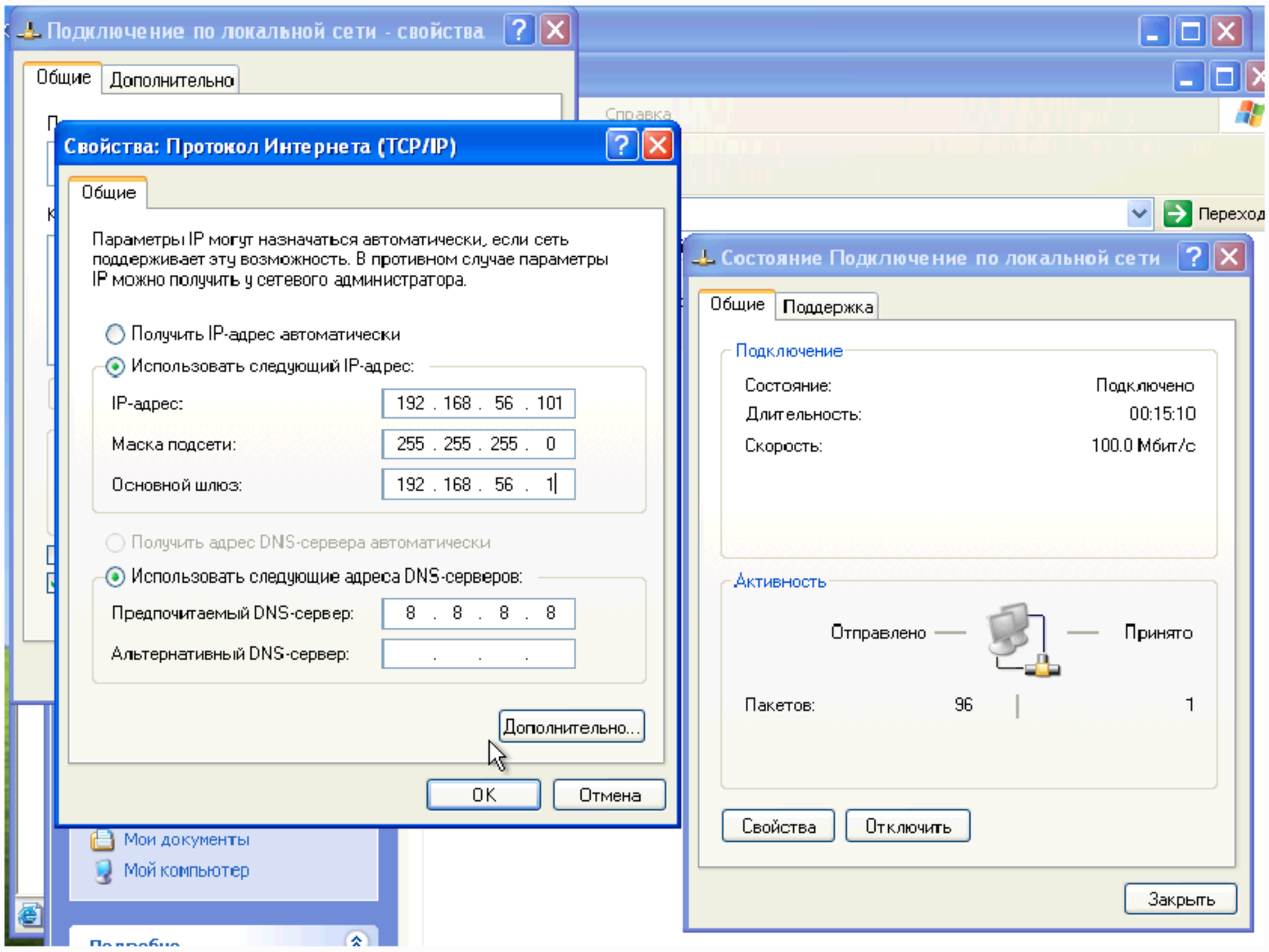
ifconfig vboxnet0 192.168.56.1

- После установки и настройки Windows переходим непосредственно на саму гостевую ОС.

**Предупреждение**

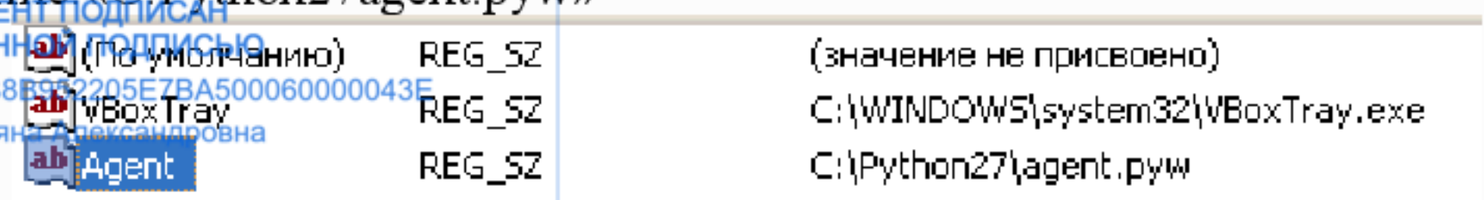
(Если возникли проблемы с установкой виртуальной машины, можете посмотреть, как это сделано в этой статье http://gwallgofi.com/cuckoo-sandbox-part-2-installing/)

- Следующим образом настроим подключение к сети (dns можете указать любой):




- Установим VboxTools с диска, который подключен к системе.
- Устанавливаем Pyton 2.7: http://python.org/download/
- Устанавливаем http://www.activestate.com/activepython.
- Устанавливаем PIL Python модуль, для создания скриншотов: http://www.pythonware.com/products/pil/.
- Отключаем автоматическое обновление Windows.
- Отключаем брандмауэр.
- Копируем агент из сетевой папки setup в папку C:Python27, Ставим агент на автозагрузку, для этого добавляем в ветку реестра(пуск->выполнить->regedit) HKLMSOFTWAREMicrosoftWindowsCurrentVersionRun строковый параметр

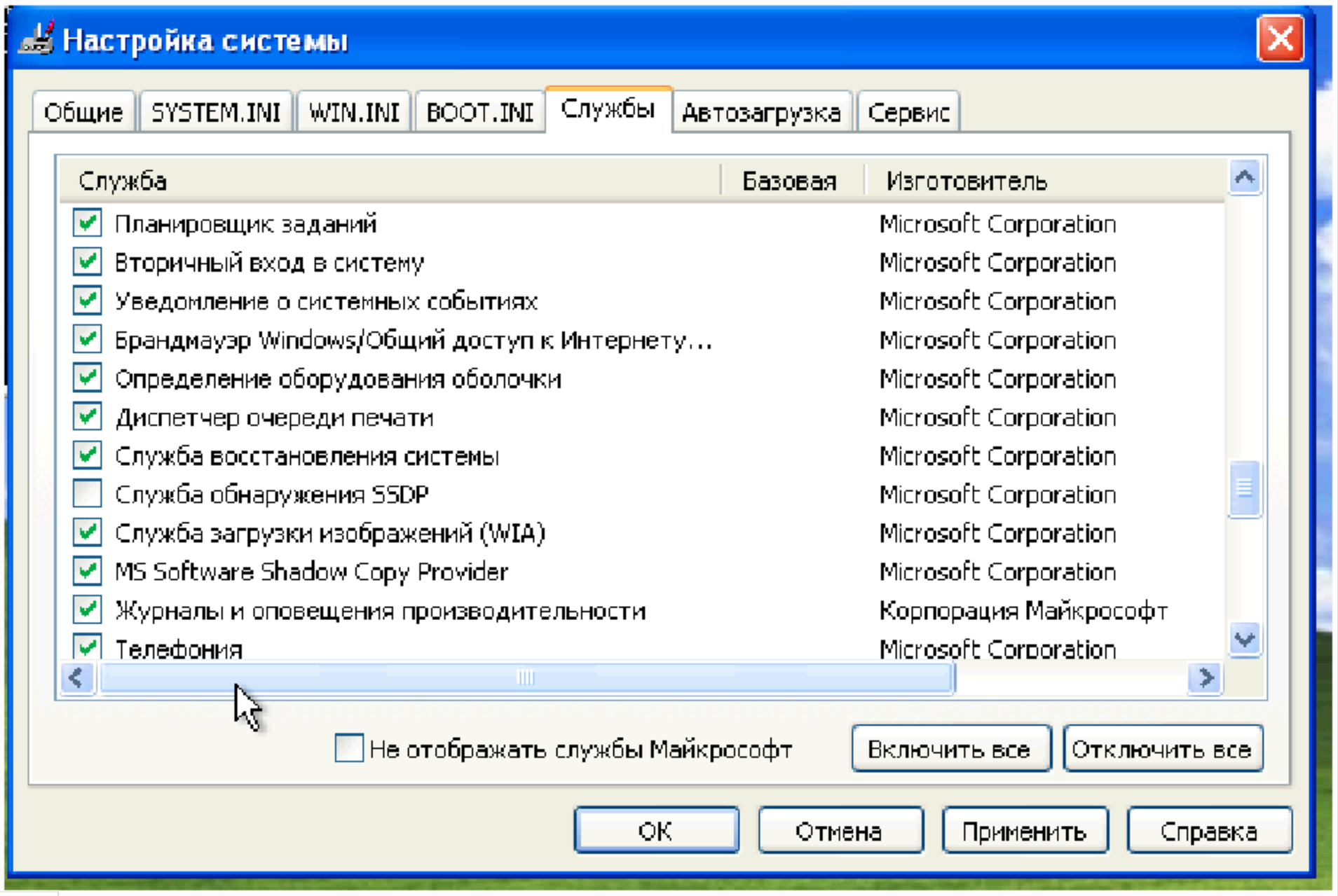
Имя:'Agent'
Тип:'REG_SZ'
Содержание:«C:Python27agent.pyw»



• Включаем IE, в настройках ставим домашней странищей пустую вкладку, по желанию в свойствах обозревателя выключаем все защитные механизмы.

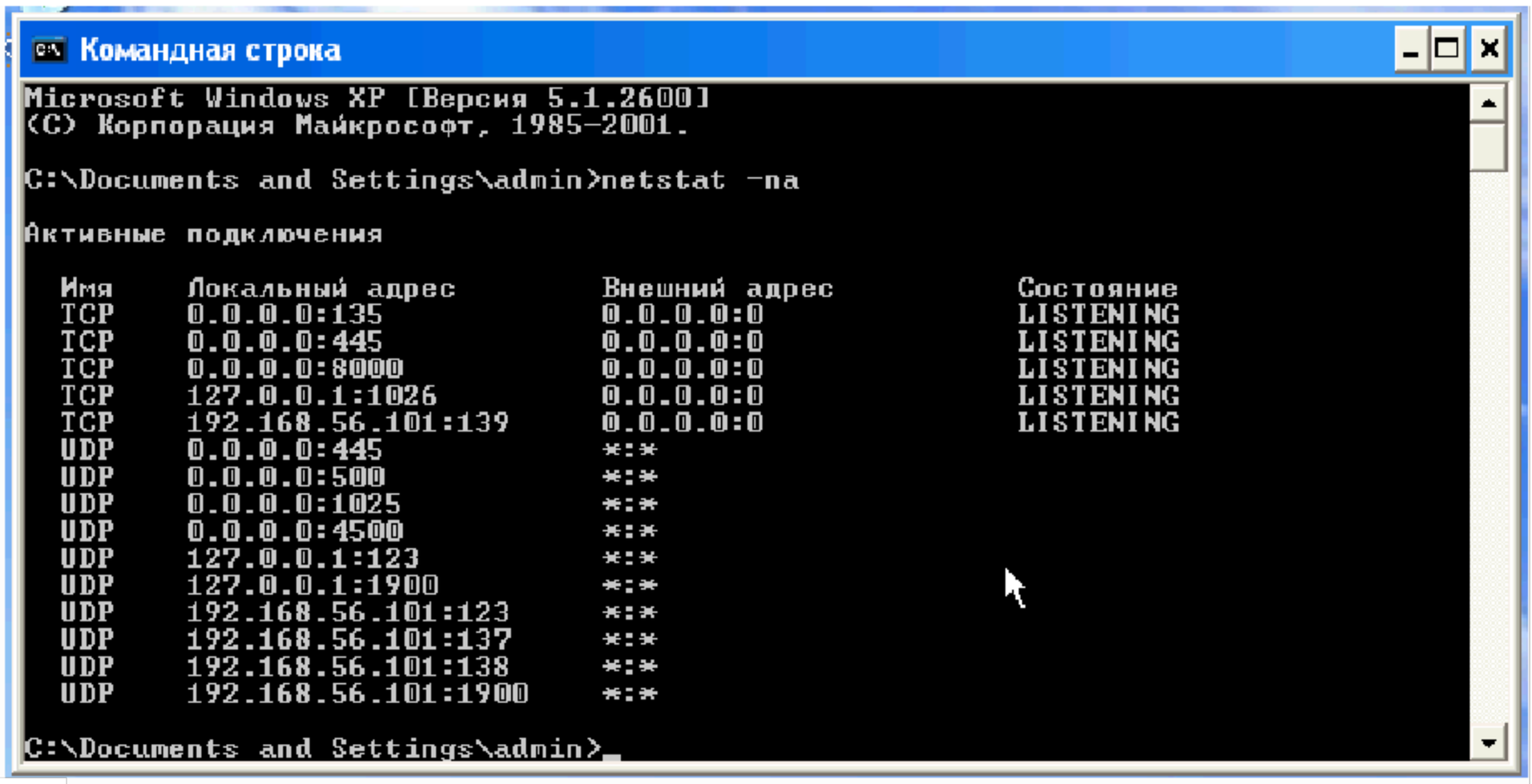
• Отключаем SSDP: пуск->выполнить->msconfig и в разделе службы отключаем «Служба обнаружения SSDP», чтобы в отчётах не фигурировали сетевые запросы этой службы.

•




• Перезагружаемся и в появившемся при загрузке окне выбираем «При перезагрузке не выводить это сообщение» и ОК.

• После перезагрузки гостевой ОС, пуск->выполнить->cmd и в консоли набираем netstat –na и смотрим есть ли агент на 8000-ом порту




- По желанию устанавливаем различное уязвимое ПО старых версий (браузеры, Flash player, Java, Acrobat Reader...)
- На этом установка гостевой ОС закончена. Делаем снапшот (не выключая гостевую ОС) <vboxmanage snapshot “WindowsXP” take “WindowsXPSnap01” –pause>
- И выключаем: <vboxmanage controlvm “WindowsXP” poweroff>
- Запуск Cukoo Sandbox <python cuckoo.py>
- Загрузите вредоносное ПО, для проверки работы Cukoo Sandbox. Здесь есть небольшой список ресурсов с образцами вредоносного ПО https://zeltser.com/malware-sample-sources/. Например, Можете загрузить с http://malshare.com/ или с https://malwr.com/ .
- Чтобы отправить файл в Cukoo на анализ используйте команду submit: <python submit.py /path/to/binary>
- Запустите web-интерфейс cukoo и просмотрите результаты <python web.py>

**Вопросы к лабораторной работе**

1. Что такое песочница?
2. Принцип работы песочниц.
3. Где используют песочницы?
4. Преимущества и недостатки песочниц.
5. Альтернативы использованию песочниц.
6. Что такое эмуляция?
7. Что такое эвристический анализ? В чем отличия от сигнатурного анализа?
8. Для чего нужен файловый антивирус?
9. Что такое вредоносное ПО?

**Составьте отчет о выполнении лабораторной работы. Включите в него копии экрана и ответы на вопросы лабораторной работы.**

Контроль выполнения задания производится по окончании занятия и на консультациях в форме защиты выполненной работы, предоставленной в электронном и в бумажном виде в форме «Отчет по лабораторной работе».

***Работа с литературой:***

| Рекомендуемые источники информации (№ источника) | | | |
|---|---|---|---|
| Основная | Дополнительная | Методическая | Интернет-ресурсы |
| 1 | 1-2 | 1-3 | 1-3 |

***Оценочные средства:*** Собеседование, отчет.

## УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

### 3.1. Рекомендуемая литература

#### 3.1.1. Основная литература:

1. Царев, Р.Ю. Программные и аппаратные средства информатики: учебник / Р.Ю. Царев, А.В. Прокопенко, А.Н. Князьков ; Министерство образования и науки Российской Федерации, Сибирский Федеральный университет. - Красноярск : Сибирский федеральный университет, 2015. - 160 с. : табл., схем., ил. - Библиогр. в кн. - ISBN 978-5-7638-3187-0 ; То же [Электронный ресурс]. - URL: //biblioclub.ru/index.php?page=book&id=435670

2. Лабораторный практикум по дисциплине Программно-аппаратные средства защиты информации [Электронный ресурс] / . — Электрон. Текстовые данные. — М. : Московский технический университет связи и информатики, 2016. — 31 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/61529.html

#### 3.1.2. Дополнительная литература:

1. Айдинян, А.Р. Аппаратные средства вычислительной техники : учебник / А.Р. Айдинян. - М. ; Берлин : Директ-Медиа, 2016. - 125 с. : ил., схем., табл. - Библиогр. в кн. - ISBN 978-5-4475-8443-6 ; То же [Электронный ресурс]. - URL: //biblioclub.ru/index.php?page=book&id=443412

2. Привалов, И. М. (Институт сервиса, туризма и дизайна (филиал)СКФУ в г. Пятигорске). Основы аппаратного и программного обеспечения : учеб.-метод. пособие / И.М. Привалов ; Сев.-Кав. федер. ун-т. - Ставрополь : СКФУ, 2015. - 145 с. - 144 с.

**1.3. Методическая литература**:

1. Методические указания по выполнению лабораторных работ по дисциплине «Программно-аппаратные средства защиты информации».
2. Методические указания по выполнению практических работ по дисциплине «Программно-аппаратные средства защиты информации».
3. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Программно-аппаратные средства защиты информации»

#### 3.1.4. Интернет-ресурсы:

http://www.biblioclub.ru/ - электронная библиотека
http://www.uts-edu.ru/ - «Электронные курсы»

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

# Методические указания

К практическим занятиям
по дисциплине
**«ПРОГРАММНО-АППАРАТНЫЕ СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ»**
для направления подготовки **10.03.01 Информационная безопасность**
направленность (профиль) **Безопасность компьютерных систем**

**Пятигорск**
**2023**

## СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

## 1. Цель и задачи изучения дисциплины

Цель дисциплины: формирование набора универсальных и общепрофессиональных компетенций будущего бакалавра (специалиста) по направлению подготовки 10.03.01 Информационная безопасность.

Задачами дисциплины являются: дать основы о методах и средствах защиты информации в компьютерных системах; дать основы правил разграничения доступа и основных функций СЗИ, его обеспечивающих; дать основы практических аспектов построения систем ограничения доступа и других СЗИ; дать основы аппаратной реализации различных средств защиты информации; дать основы о защитных механизмах, реализованных в средствах защиты компьютерных систем от несанкционированного доступа (НСД); дать основы вопросов защиты ПО от несанкционированного использования; дать основы о применении средств криптографической защиты информации и средств защиты от НСД для решения задач обеспечения информационной безопасности; дать основы методов защиты от РПВ; дать основы методов и особенностей защиты объектов ОС; дать основы принципов построения файловой системы и моделей разграничения доступа к объектам.

## 2. Оборудование и материалы

Для проведения практических занятий необходимо следующее материально-техническое обеспечение: персональный компьютер; проектор; возможность выхода в сеть Интернет для поиска по образовательным сайтам и порталам; интерактивная доска.

## 3. Наименование практических занятий

| № Темы дисциплины | Наименование тем дисциплины, их краткое содержание | Объем часов |
|---|---|---|
| 5 семестр | | |
| 1 | **Практическое занятие 1.** Сбор данных об информационной системе с помощью средств администрирования Windows. | 6.75 |
| 1 | **Практическое занятие 2.** Сбор данных о топологии сети с помощью средства администрирования сетей. | 6.75 |
| 5 | **Практическое занятие 3.** Идентификация и аутентификация систем семейства Microsoft Windows. | 6.75 |
| 5 | **Практическое занятие 4.** Аутентификация по протоколу Kerberos. | 6.75 |
| | Итого за 5 семестр | 27 |
| 6 семестр | | |
| 19 | **Практическое занятие 5.** Настройка локальной политики парольной безопасности операционной системы. | 3 |
| 19 | **Практическое занятие 6.** Инфраструктура открытых ключей. Цифровые сертификаты. | 3 |
| 20 | **Практическое занятие 7.** Использование цифровых сертификатов. | 3 |
| 20 | **Практическое занятие 8.** Резервное копирование в Windows Server 2012. | 3 |
| | Итого за 6 семестр | 12 |
| | **Итого** | **39** |

## 4. Содержание практических занятий

### 3.1.ПРАКТИЧЕСКАЯ РАБОТА 1. Сбор данных об информационной системе с помощью средств администрирования Windows.

**Цель работы:**

Целью данной лабораторной работы является сбор данных об имеющихся компьютерах, установленных на них операционных системах, предоставляемых в общий *доступ* файловых ресурсах.

**Содержание:**

Получение перечня компьютеров домена. Определение перечня общих ресурсов рабочей станции. Управление компьютером. Информация о членстве пользователя в доменных группах. Разрешения файловой системы.

**Теоретические основы.**

Для проведения оценки рисков необходимо провести инвентаризацию активов информационной системы (ИС). Если в ИС используются домены *Windows*, для получения данных о системе можно использовать средства администрирования, реализованные в виде оснасток консоли администрирования (Microsoft *management* console - *mmc*).

Используемые в данной работе инструменты могут быть запущены из раздела **"Администрирование"** *меню* **"Пуск"** или через **"Панель управления" (Пуск  Панель управления  Администрирование)**.

Целью данной лабораторной работы является сбор данных об имеющихся компьютерах, установленных на них операционных системах, предоставляемых в общий *доступ* файловых ресурсах.

Из раздела **"Администрирование"** запустите **Active Directory Users and Computers**. В раскрывающемся списке объектов выберите **Ваш домен**, там откройте перечень компьютеров (*папка* **Computers** - рис. 1.1).

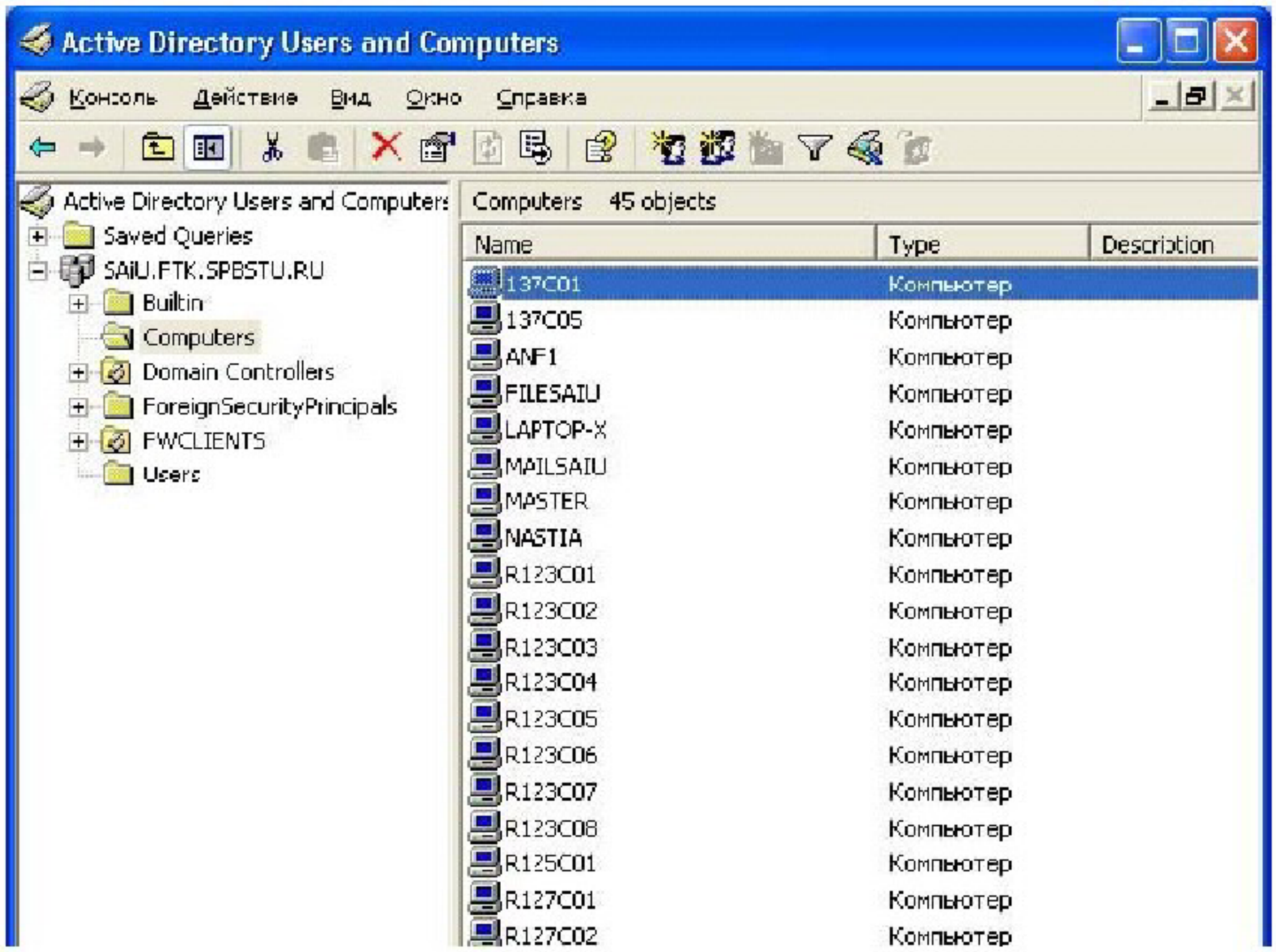



Рис. 1.1. Получение перечня компьютеров домена

С помощью кнопки панели инструментов **"Экспорт списка"** (на кнопке изображение списка и стрелки) *список* компьютеров можно экспортировать в *текстовый файл* для дальнейшей обработки. В свойствах компьютера отображается название и версия установленной операционной системы (рис. 1.2). Также там может быть дополнительная *информация*, например, описывающая *размещение*.

Аналогичные данные о контроллерах домена можно получить в разделе **Domain Controllers**. Данные о пользователях и их группах доступны в разделе **Users**. Надо отметить, что представленное распределение *по* разделам не является обязательным. В процессе администрирования могут создаваться новые *подразделения* (OU - Organization Unit) и объекты (например, пользователи или компьютеры) - помещаться в них.

Информацию о соответствии имен компьютеров IP-адресам можно получить, используя *утилиту командной строки* **nslookup** или административную оснастку **"DNS"**. Например, узнать *IP-адрес* компьютера http://comp1.mcompany.ru можно с помощью команды nslookup comp1.mcompany.ru Часто действующие настройки в сети таковы, что ip-адреса компьютерам выделяются динамически, с использованием службы **dhcp**, и могут периодически меняться. Как правило, у серверов ip-адреса постоянны.

Теперь перейдем к этапу сбора данных об информационных ресурсах, поддерживаемых на компьютере. Перечень предоставляемых в общий *доступ* папок можно получить с помощью оснастки **"Управление компьютером"**. На рис. 1.3 представлен пример перечня ресурсов рабочей станции, предоставляемых в общий *доступ* в служебных целях. Этот *список* можно также экспортировать в *текстовый файл*.

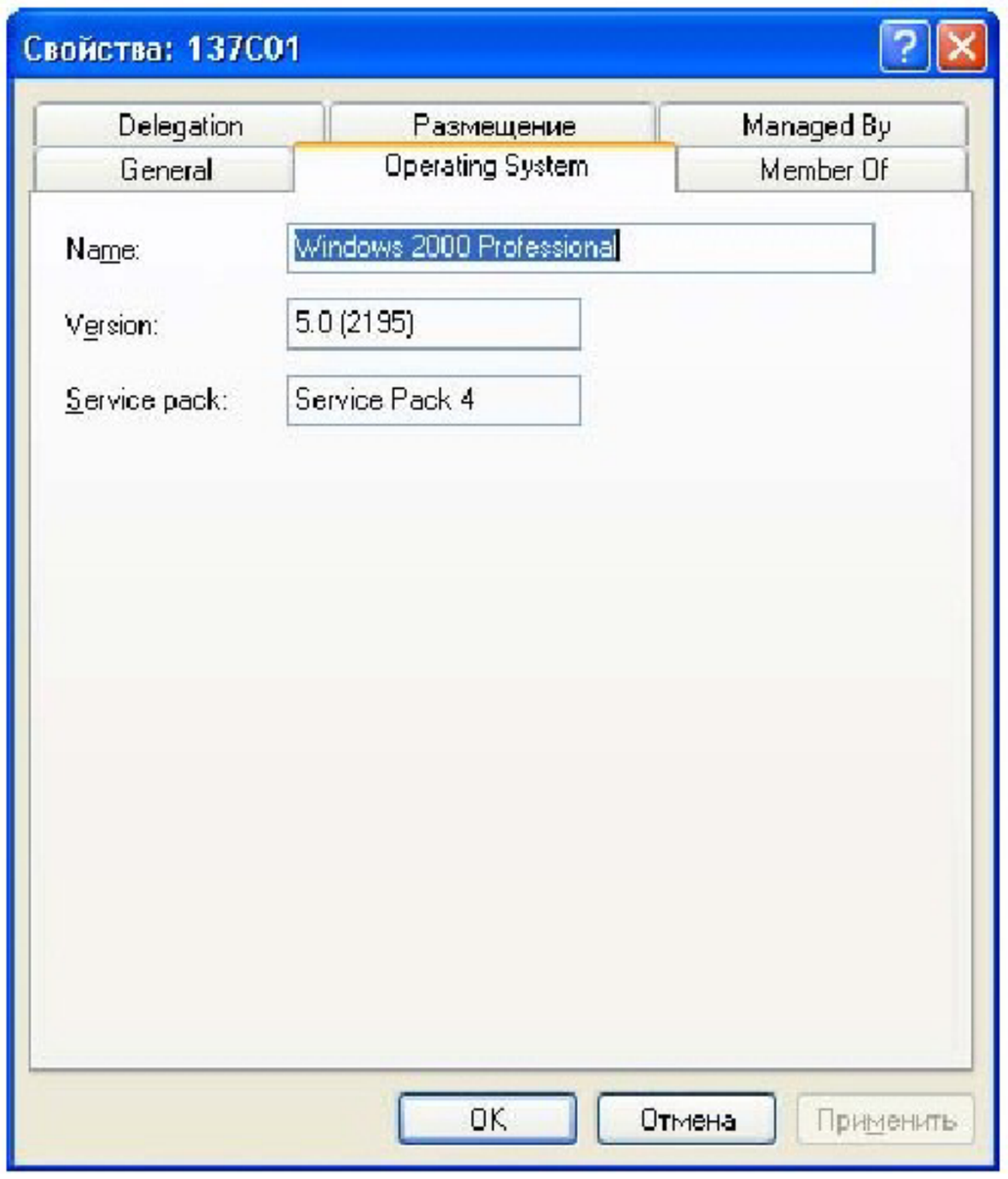



Рис. 1.2. Информация о компьютере

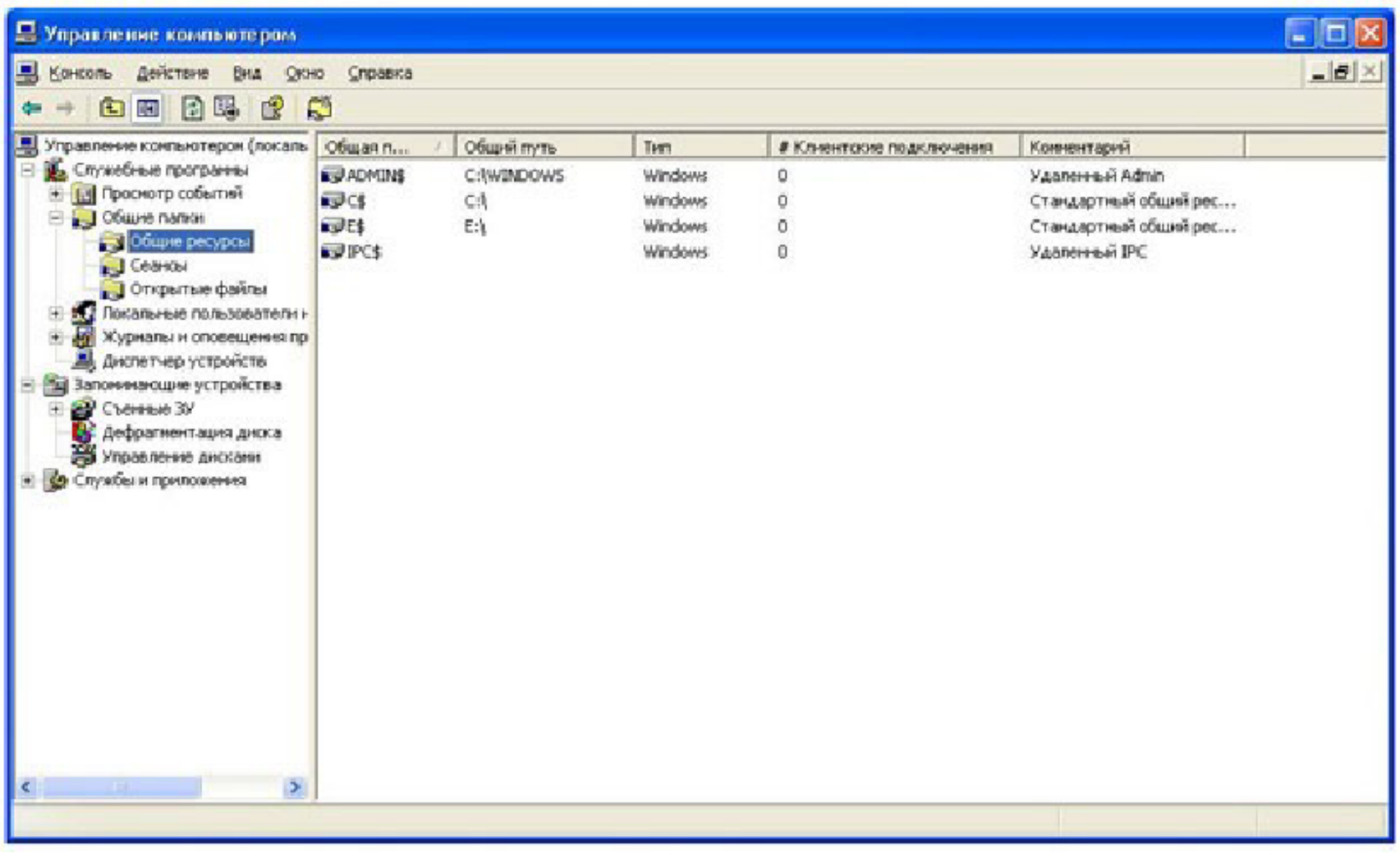



Рис. 1.3. Пример перечня общих ресурсов рабочей станции

Более интересен будет подобный *список* для файлового сервера. Чтобы его увидеть, надо подключить оснастку **"Управление компьютером"** для сервера. Запустите *консоль MMC* (**Пуск→Выполнить→mmc**), в *меню* выберите добавление новой оснастки (рис. 1.4), выберите оснастку **"Управление компьютером"** и укажите, что она будет использоваться для другого компьютера (рис. 1.5).

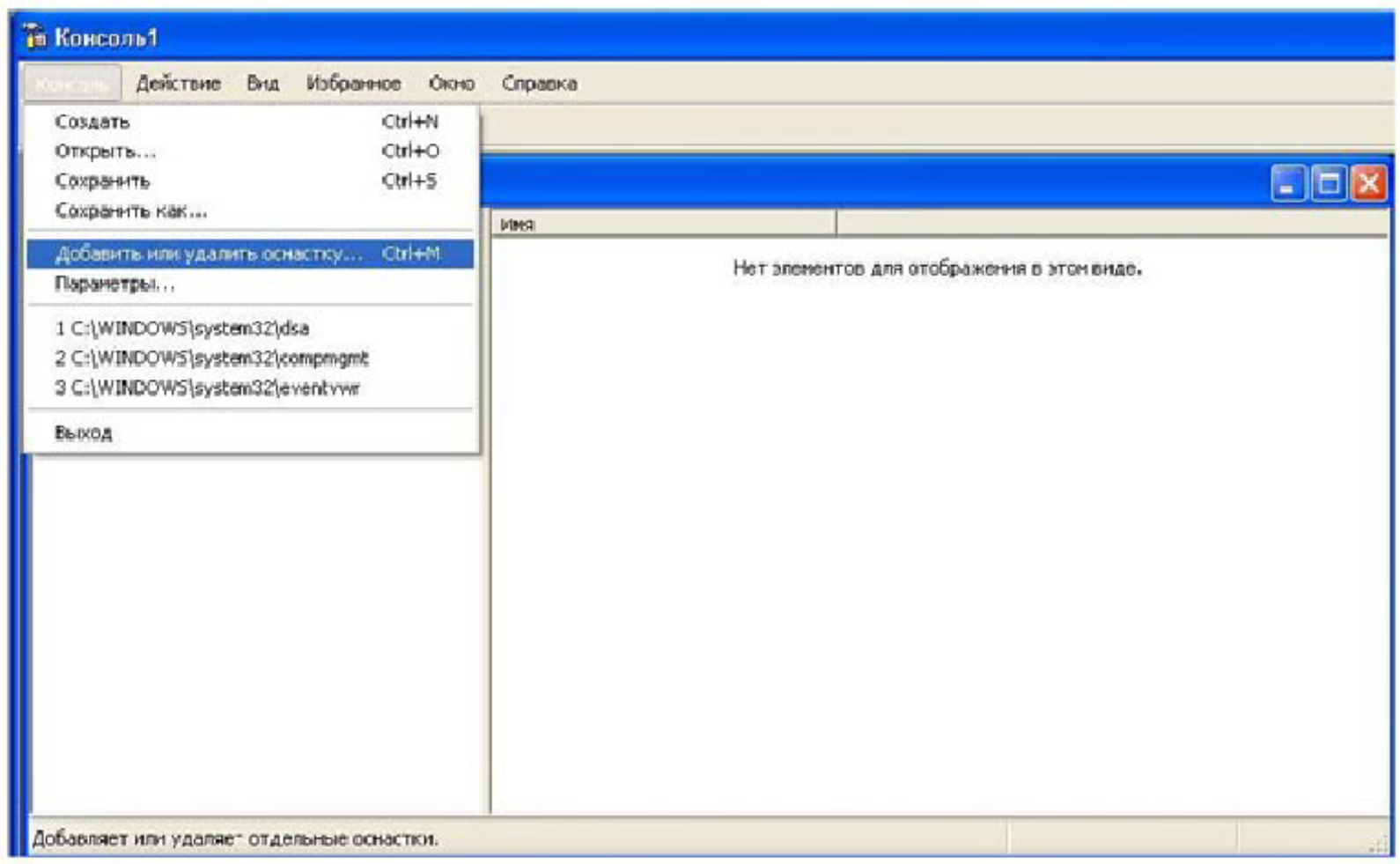



Рис. 1.4. Добавление новой оснастки

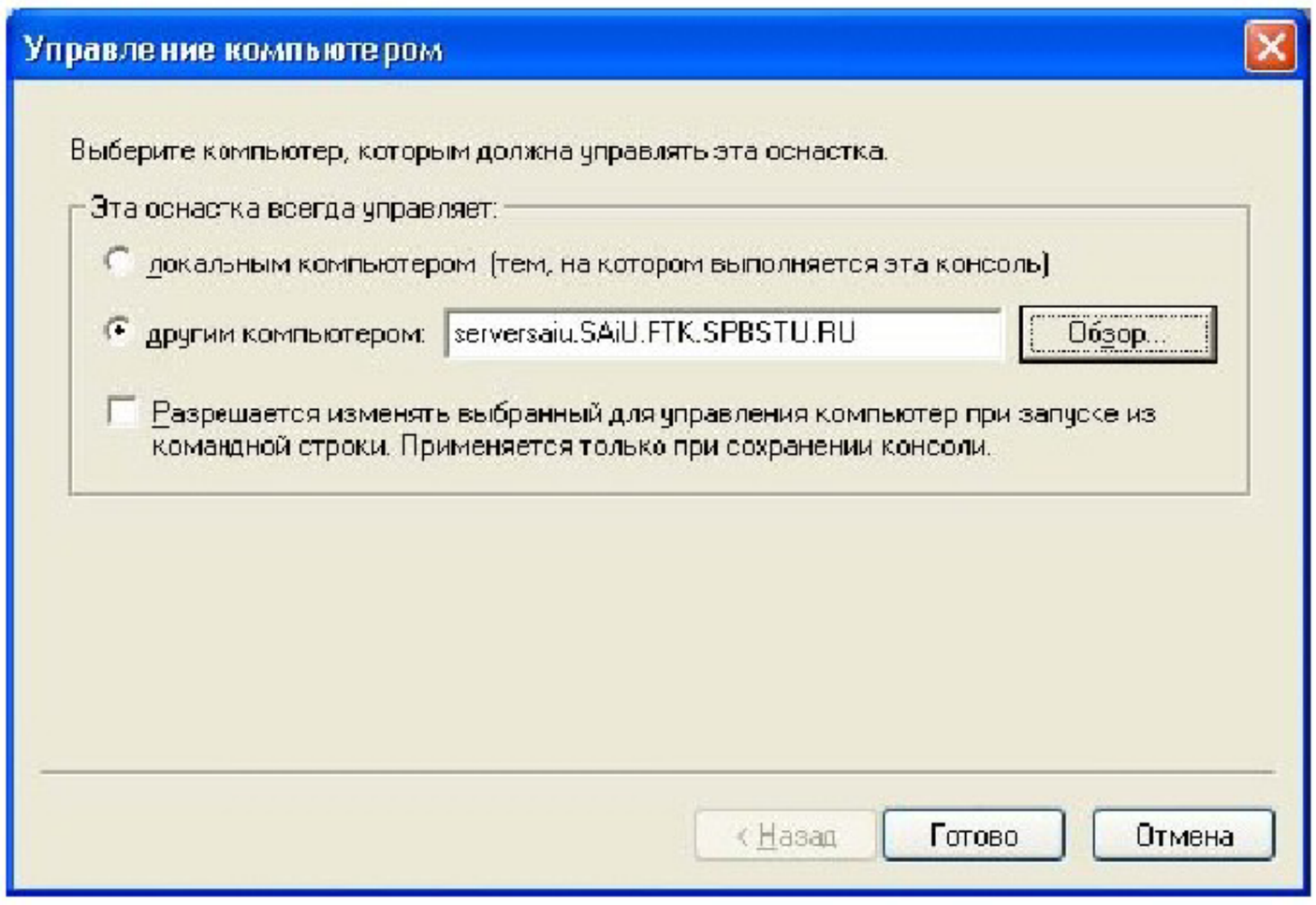



Рис. 1.5. Выбор компьютера

В остальном для пользователя все будет происходить так же, как и при работе с локальным компьютером.

В свойствах ресурса можно узнать о разрешениях, которые установлены на него как для разделяемого ресурса (рис. 1.6), а на вкладке **"Безопасность"** - разрешениях файловой системы *NTFS* (если *папка* расположена на разделе с этой файловой системой, а не с *FAT*).

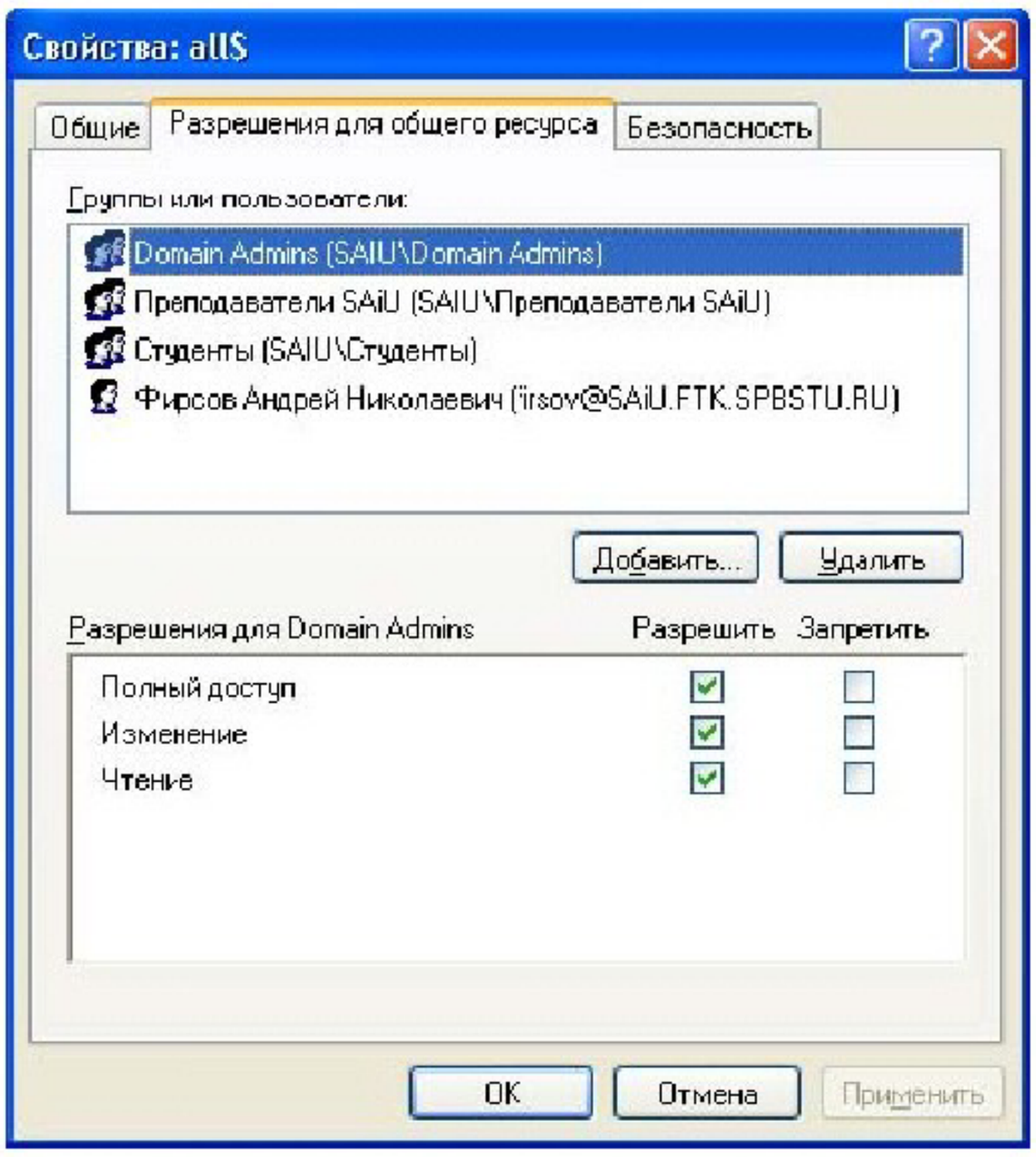



Рис. 1.6. Разрешения

## Задание

1. Получите перечень компьютеров и контроллеров домена. Для указанных преподавателем 1-2 компьютеров выясните установленную операционную систему и используемые ими ip-адреса. Занесите данные в отчет.

2. Получите перечень предоставляемых в общий доступ каталогов на вашем компьютере и на компьютерах, данные о которых Вы собирали на этапе 1. Опишите хранимые там данные и охарактеризуйте степень их важности. Занесите полученную информацию в отчет.

3. Для указанных ресурсов и выбранных пользователей опишите действующие разрешения на доступ. При этом надо учитывать, что:

- эффективное (действующее) разрешение складывается из разрешений для пользователя лично и разрешений всех групп, в которые пользователь входит;
- запрещение имеет больший приоритет, чем разрешение;
- при комбинации разрешений для общего ресурса с разрешениями NTFS, приоритетными будут разрешения, максимально ограничивающие доступ.

Информацию о членстве пользователя в доменных группах можно получить через оснастку **Active Directory Users and Computers**, о локальных группах – через **"Управление компьютером"**

**Контрольные вопросы:**

1. Линии связей локальных сетей.
2. Обеспечение секретности передаваемой информации в локальных сетях.
3. Типы линий связи локальных сетей.
4. Бескабельные каналы связи локальных сетей.

5. Базовые технологии локальных сетей.
6. Нестандартные топологии локальных сетей.
7. Согласование и экранирование линий связи.

**Работа с литературой:**

| Рекомендуемые источники информации (№ источника) | | | |
|---|---|---|---|
| Основная | Дополнительная | Методическая | Интернет-ресурсы |
| 1-2 | 1-3 | 1-2 | 1-3 |

**Оценочные средства:** собеседование.

## 3.2.ПРАКТИЧЕСКАЯ РАБОТА 2. Сбор данных о топологии сети с помощью средства администрирования сетей.

**Цель работы:**
Изучение различных вариантов сбора данных о топологии сети с помощью средств администрирования сетей с помощью *3Com Network Supervisor* .

**Содержание:**

Выбор документируемой сети. Функции, доступные для выбранного узла. Соединения по типам подключений. Данные о подключениях и графики. Свойства коммутатора. Средства удаленного администрирования.

Продолжая тему инвентаризации активов информационной системы (ИС), перейдем к рассмотрению средств, позволяющих получить данные о составе и топологии сети. В качестве примера в данной лабораторной работе будет использоваться *утилита 3Com Network Supervisor*, которую можно бесплатно получить с сайта компании *3Com* (http://www.3com.com). Аналогичные *по* функциональности продукты есть и у других производителей сетевого оборудования.

При запуске программы предлагается выбор - строить новую карту сети или открыть существующую. При выборе создания новой карты надо указать, какая *подсеть* документируется (рис. 2.1). На рисунке выбрана локальная *подсеть*, т.е. та ip-*сеть*, к которой относится *компьютер*, на котором выполняется *3Com Network Supervisor*.

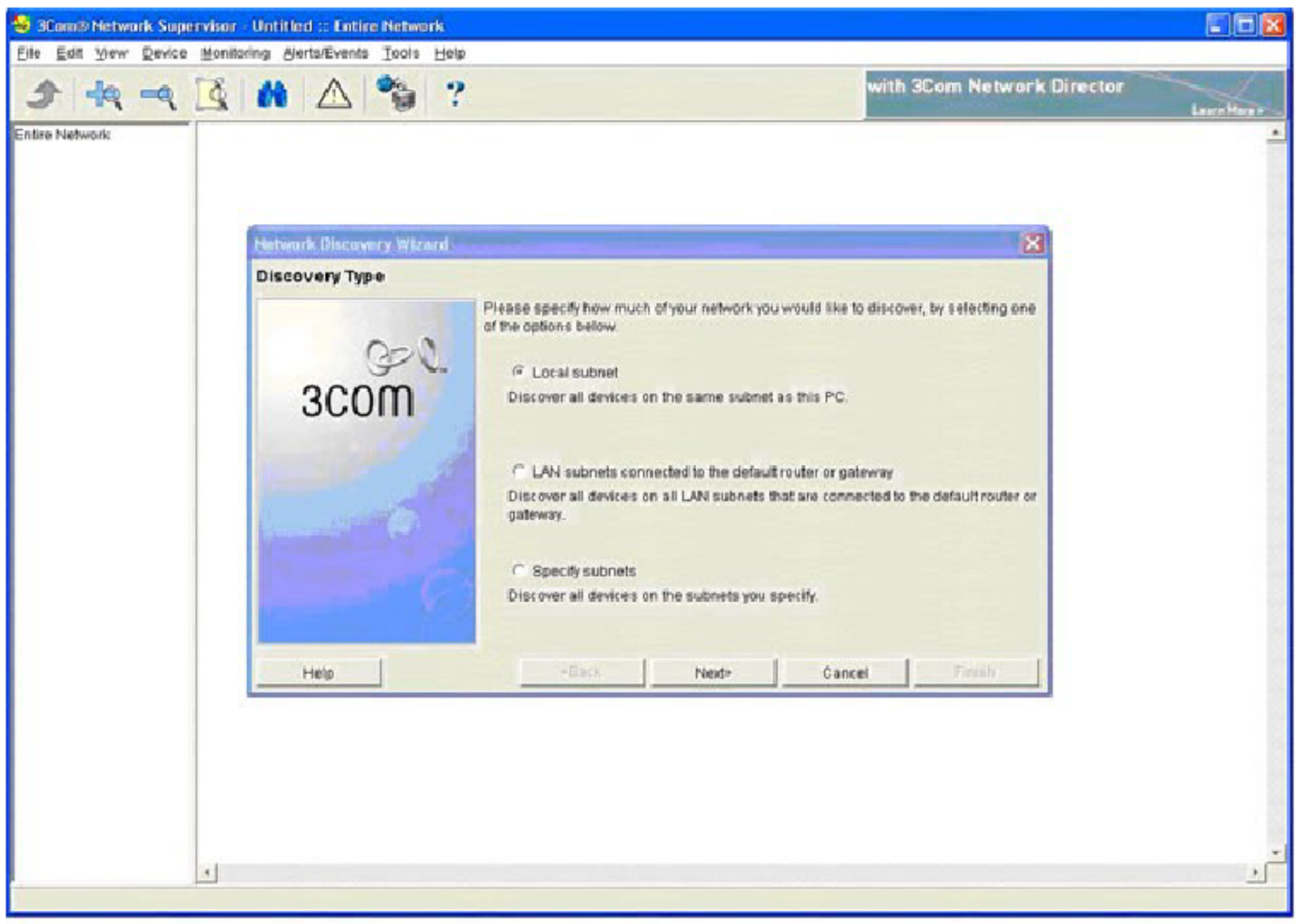



Рис. 2.1. Выбор документируемой сети

На рис. 2.2 представлен пример карты сети, которую строит *утилита*. Надо отметить, что наиболее информативна такая карта будет в том случае, если в сети используется управляемое сетевое оборудование *3Com*, поддерживающее, в частности, протокол *SNMP*. В то же время, польза от составления карты будет и в случае отсутствия в сети подобного оборудования.

Для того, чтобы это продемонстрировать, были сделаны следующие настройки. Каждому из компьютеров были присвоены ip-адреса из двух сетей класса C - 192.168.1.0 и 192.168.100.0. Управляемому коммутатору *3Com* SuperStack II *Switch* 3000 назначен

*адрес* 192.168.100.6, т.е. он "виден" только при построении карты сети 192.168.100.0. *DNS* серверы доступны только в сети 192.168.1.0, поэтому на рисунках, относящихся ко второй сети, компьютеры идентифицируются только ip-адресами. Карта сети 192.168.1.0 представлена на рис. 2.3.

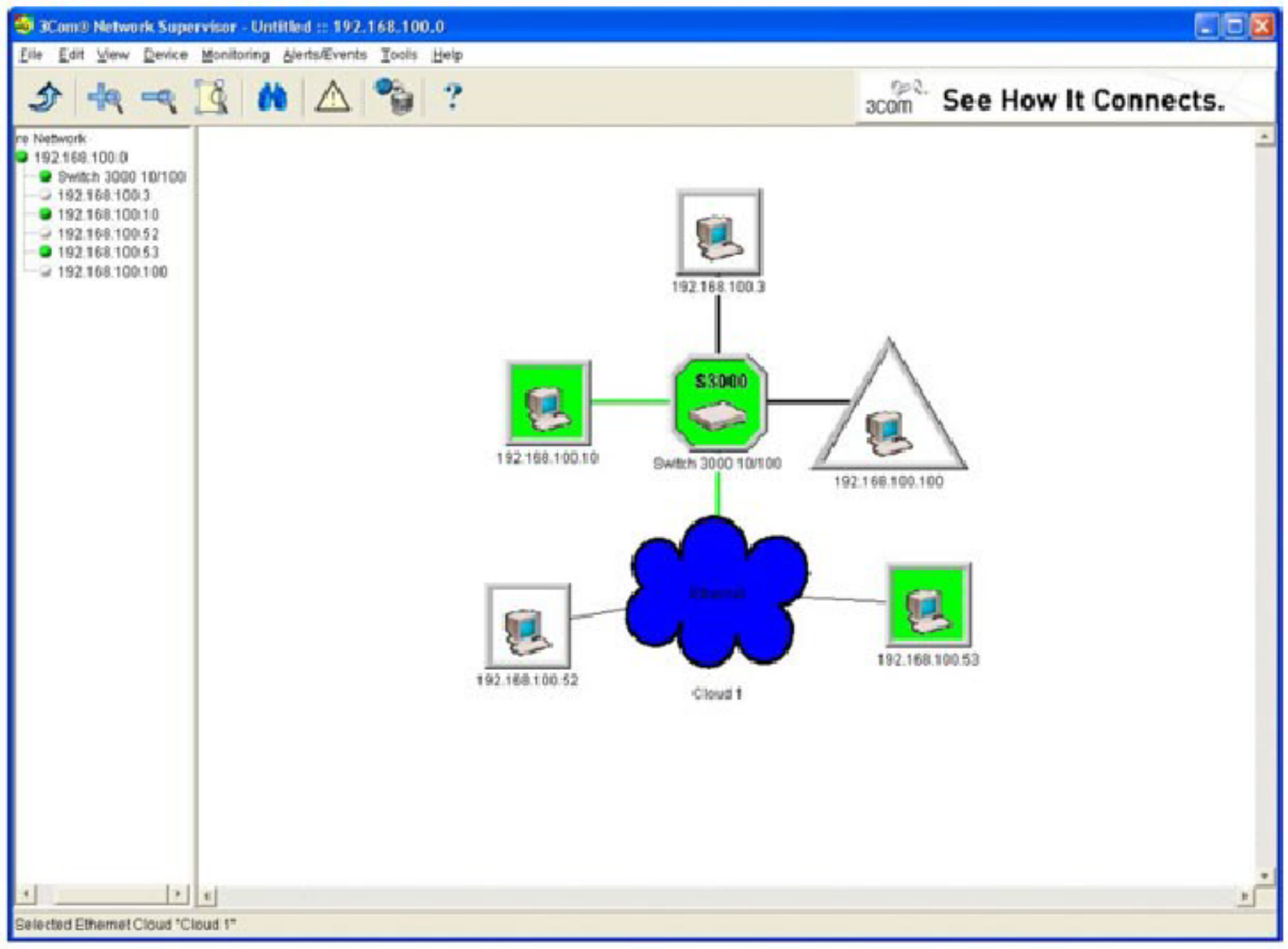



Рис. 2.2. Карта сети 192.168.100.0. Cloud 1 скрывает неуправляемый коммутатор

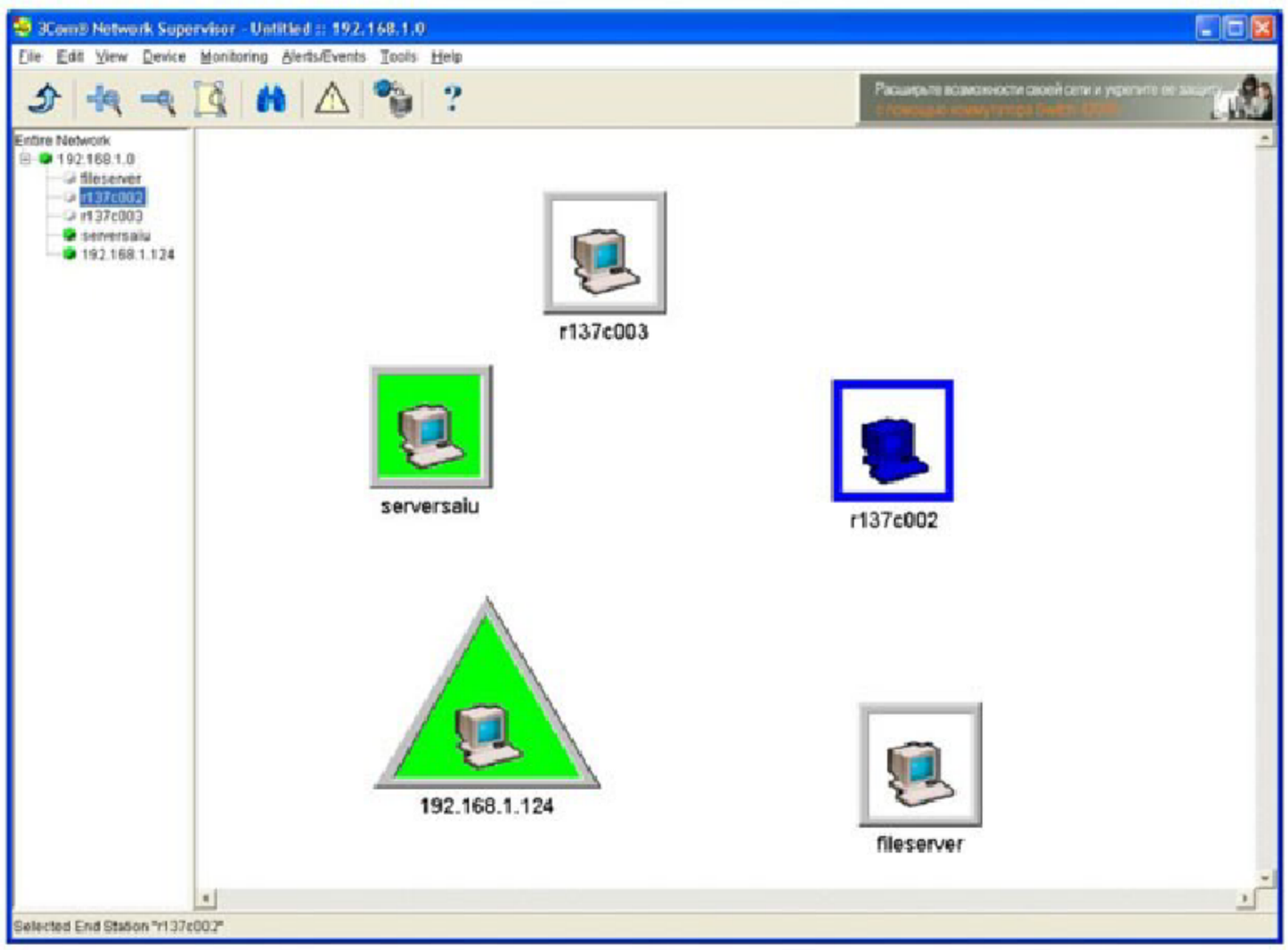



Рис. 2.3. Карта сети 192.168.1.0. Информация от управляемого коммутатора недоступна

Для выбранного узла можно потребовать провести *мониторинг* загрузки различных сетевых сервисов или обратиться к средствам *удаленного администрирования*, использующим протоколы http, telnet или *ssh* ( рис. 2.4,2.5).

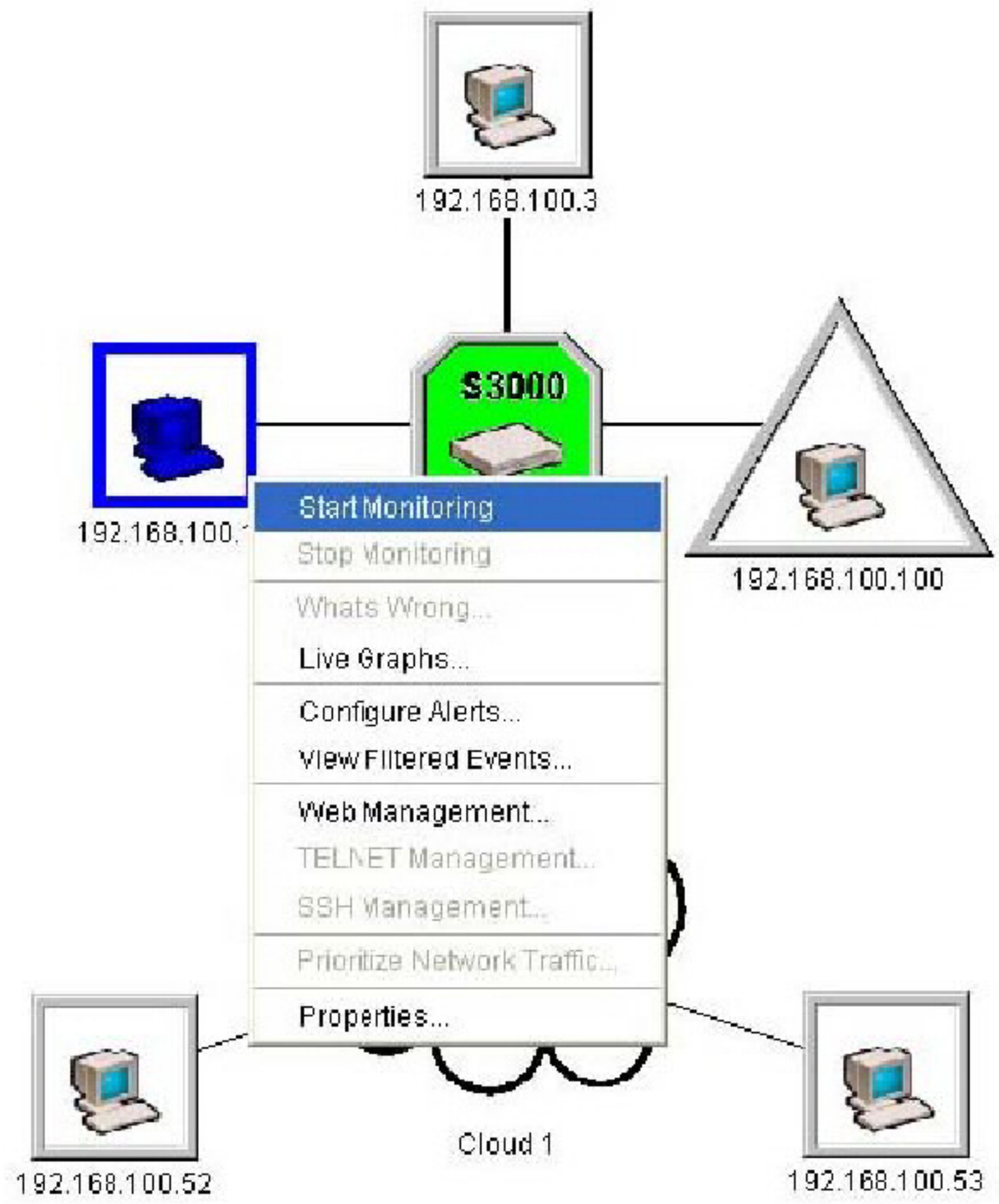



Рис. 2.4. Функции, доступные для выбранного узла

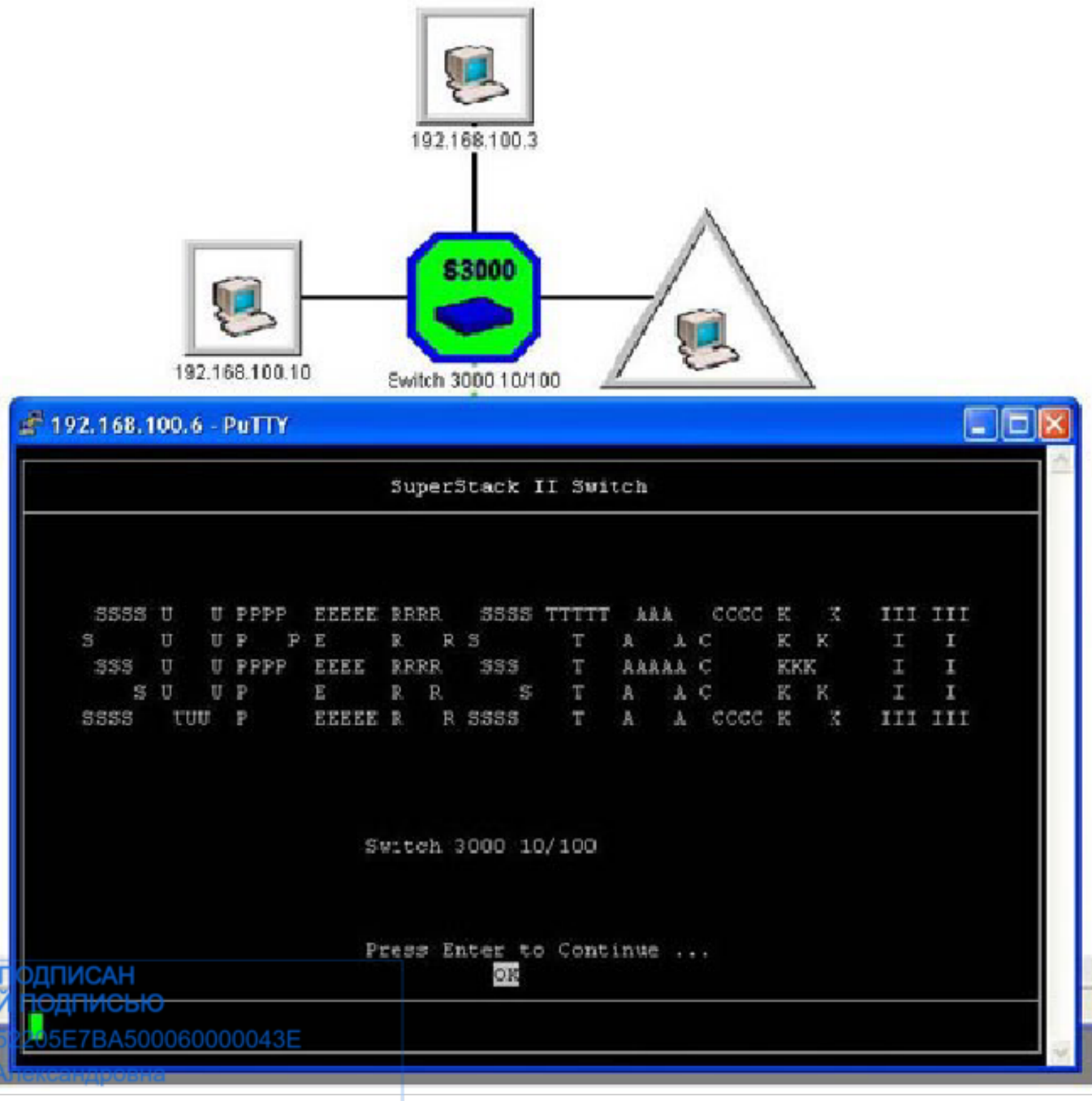


Рис. 2.5. Запуск удаленного терминала для администрирования коммутатора Switch 3000

*Функция* поиска (кнопка панели инструментов с изображением бинокля) позволяет, в частности, отобразить информацию о типах используемых сетевых подключений ( рис. 2.6,2.7). Через свойства управляемого коммутатора доступна *информация* о том, к какому порту какой узел подключен и графики загрузки ( рис. 2.8,2.9).

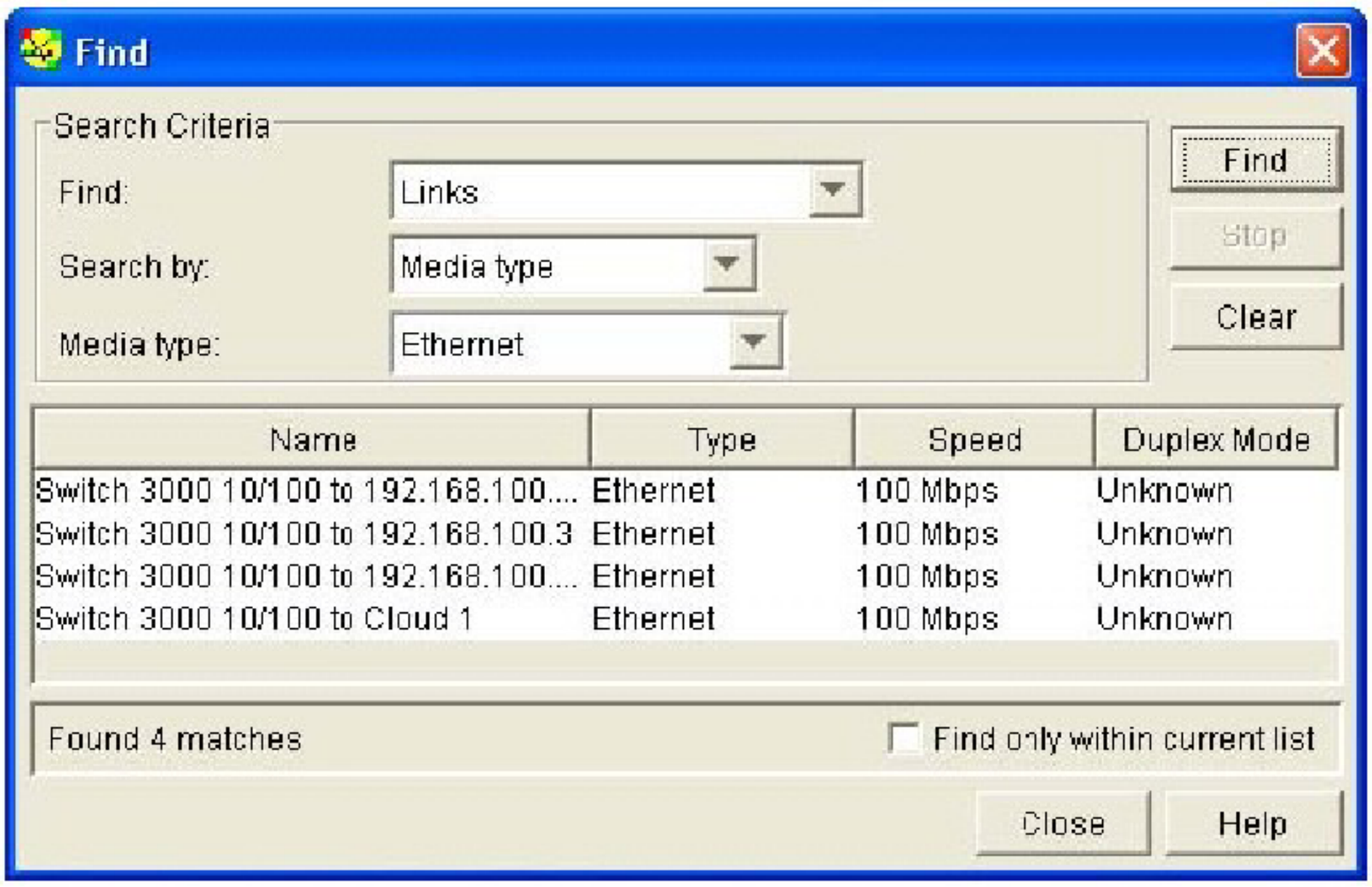



Рис. 2.6. Соединения по типам подключений. Ethernet

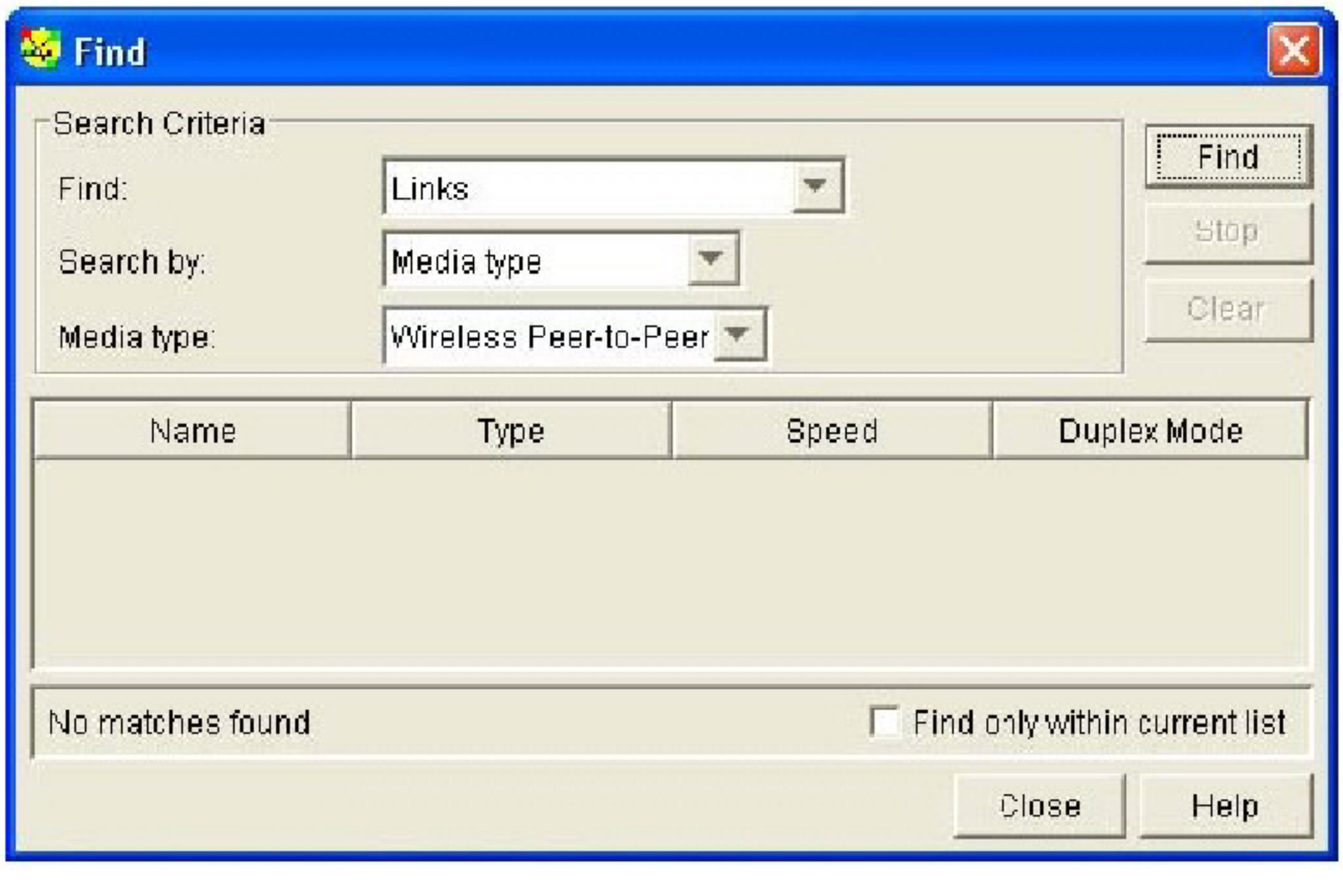



Рис. 2.7. Соединения по типам подключений. Беспроводные подключения (отсутствуют)

Собранная *информация* может отображаться в виде отчетов, формируемых в формате *HTML*. *Опция* доступна через *меню* **Tools пункт Reports**. Для задач, связанных с инвентаризацией системы, наибольший интерес представляют отчеты **Inventory Report** и **Topology Report**.

Отчет *по* топологии сети 192.168.1.0 состоит из записи "Нет данных", т.к. данные о топологии *программа 3Com Network Supervisor* получить не смогла (в этой сети управляемый *коммутатор* "невидим", т.к. его *адрес* принадлежит другой ip-сети).

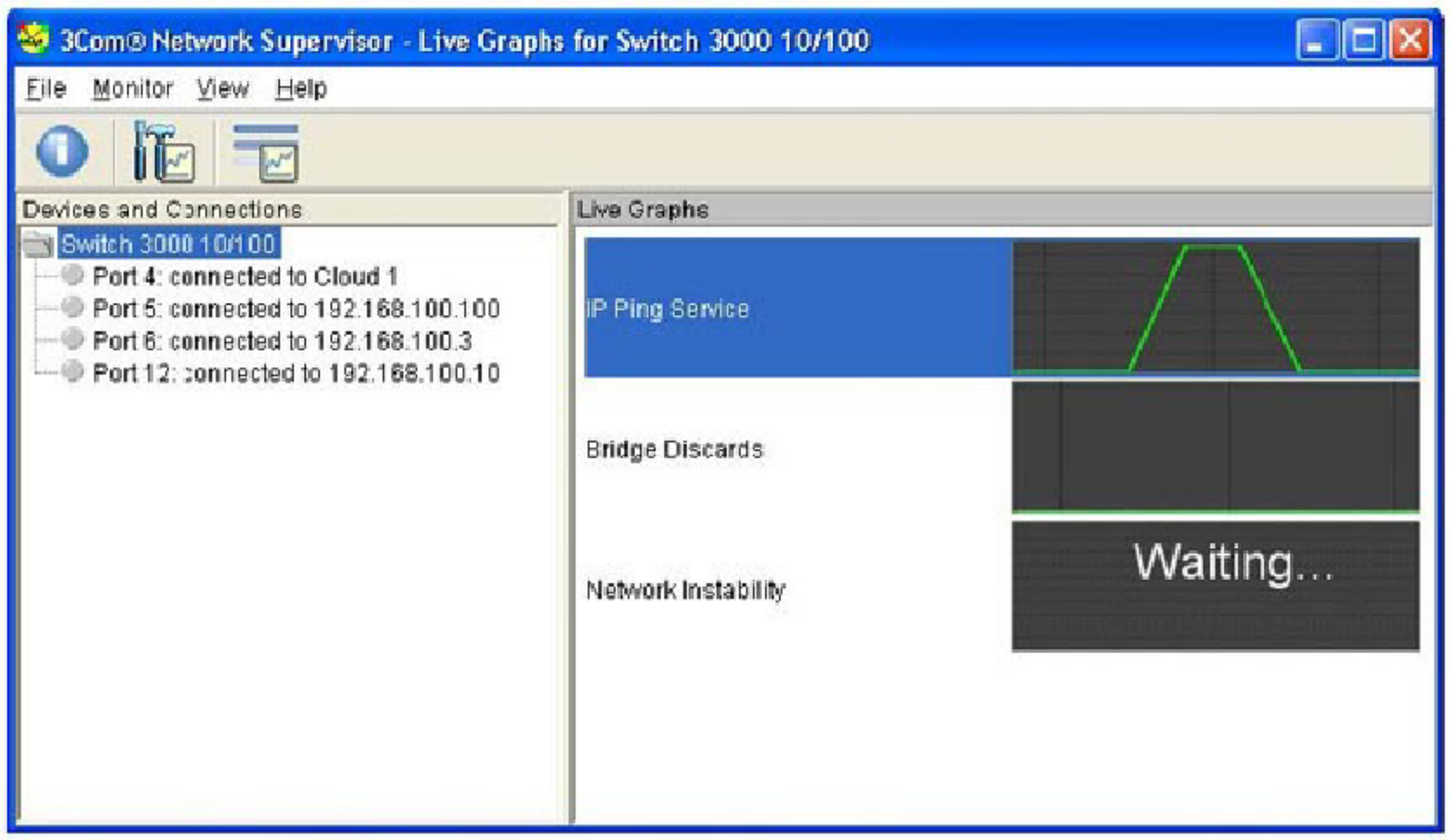



Рис. 2.8. Данные о подключениях и графики

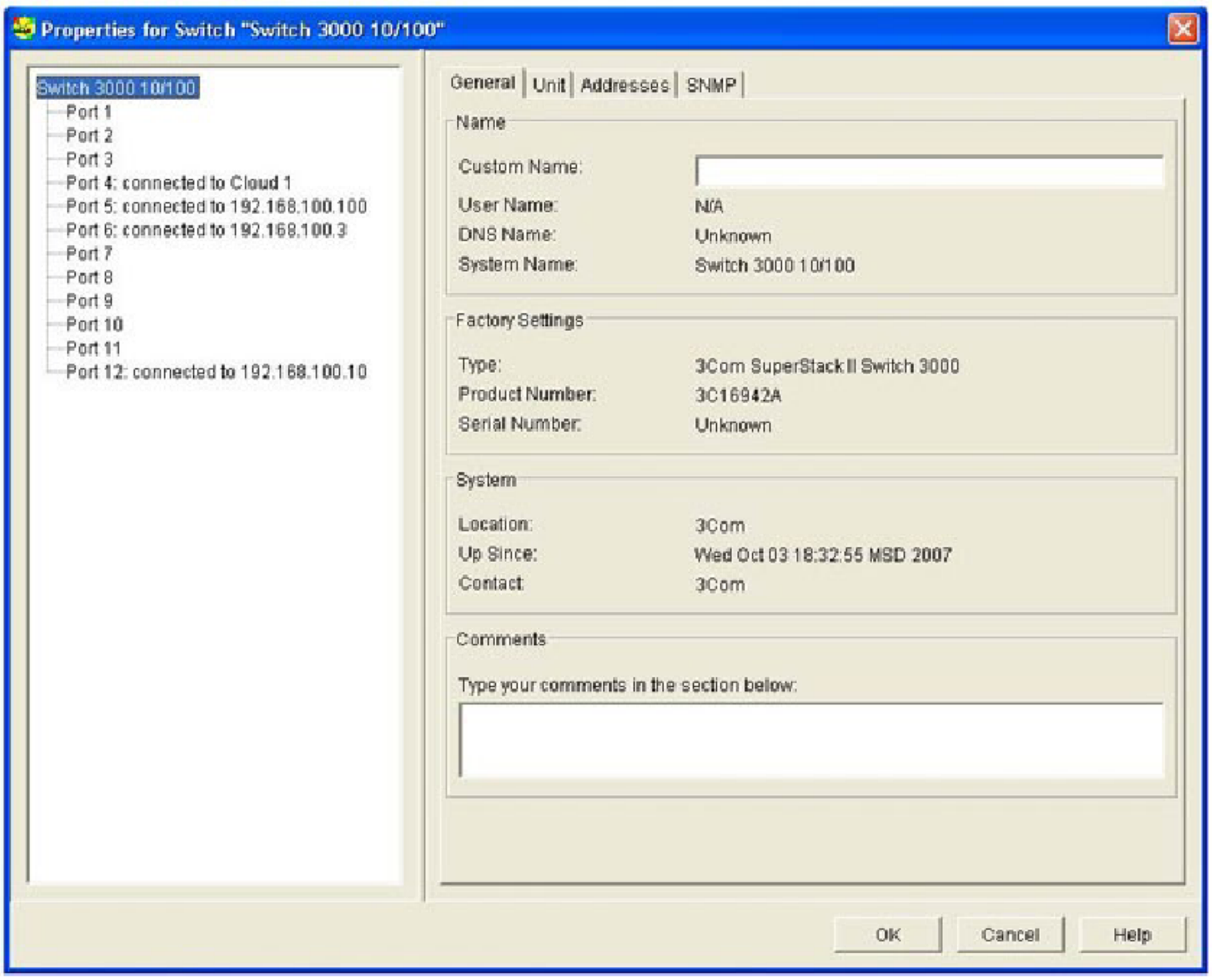



Рис. 2.9. Свойства коммутатора



## Задание

С помощью *3Com Network Supervisor* постройте карту сети учебной лаборатории. Опишите узлы сети, используемые типы соединений, доступные средства *удаленного администрирования*.

Перечислите используемые сетевые устройства и укажите, какие последствия будут при выходе из строя (или некорректной работе) каждого из них.

**Работа с литературой:**

| Рекомендуемые источники информации (№ источника) | | | |
|---|---|---|---|
| Основная | Дополнительная | Методическая | Интернет-ресурсы |
| 1-2 | 1-3 | 1-2 | 1-3 |

**Оценочные средства:** собеседование.

## 1.2. ПРАКТИЧЕСКАЯ РАБОТА 3. Идентификация и аутентификация систем семейства Microsoft Windows.

**Цель работы:**

Изучить способы идентификации и аутентификации систем симейства Microsoft Windows.

Система идентификации и аутентификации. Утилиты Microsoft Baseline Security Analyzer. Двухфакторная аутентификация. Парольные системы аутентификации. Учетная запись пользователя. Задание минимальной длины используемых в системе паролей. Проверка администраторами безопасности качества используемых паролей путем имитации атак. Ограничение числа неудачных попыток ввода пароля. Журнала истории паролей.

**Теоретическая часть.**

В данном разделе будут рассмотрены некоторые технические меры повышения защищенности систем. Выбор рассматриваемых мер обусловлен возможностью их реализации встроенными средствами операционных систем семейства Microsoft *Windows*. Соответственно, уровень защищенности может быть повышен без дополнительных затрат на специализированные средства защиты.

В теоретической части курса будут методы, лежащие в основе соответствующих средств и механизмов. В лабораторных работах рассматриваются конкретные настройки операционных систем.

Рассматриваемые вопросы можно разделить на две группы:

- вопросы, связанные с *идентификацией и аутентификацией* пользователей;
- защита передаваемых сообщений.

**Идентификация и аутентификация**

**Идентификация** - присвоение пользователям идентификаторов (уникальных имен или меток) под которыми система "знает" пользователя. Кроме идентификации пользователей, может проводиться *идентификация* групп пользователей, ресурсов ИС и т.д. *Идентификация* нужна и для других системных задач, например, для ведения журналов событий. В большинстве случаев *идентификация* сопровождается аутентификацией.

**Аутентификация** - установление подлинности - проверка принадлежности пользователю предъявленного им идентификатора. Например, в начале сеанса работы в ИС *пользователь* вводит имя и *пароль*. На основании этих данных система проводит идентификацию (*по* имени пользователя) и аутентификацию (сопоставляя *имя пользователя* и введенный *пароль*).

Система идентификации и аутентификации является одним из ключевых элементов инфраструктуры защиты от несанкционированного доступа (НСД) любой информационной системы. В соответствии с рассмотренной ранее моделью многоуровневой защиты, *аутентификация* пользователя компьютера относится к уровню защиты узлов.

Обычно выделяют 3 группы методов аутентификации.

1. Аутентификация по наличию у пользователя уникального объекта заданного типа. Иногда этот класс методов аутентификации называют по-английски "I have" ("у меня есть"). В качестве примера можно привести аутентификацию с помощью смарт-карт или электронных USB-ключей.

2. Аутентификация, основанная на том, что пользователю известна некоторая конфиденциальная информация - "I know" ("я знаю"). Например, аутентификация по паролю. Более подробно парольные системы рассматриваются далее в этом разделе.

3. Аутентификация пользователя по его собственным уникальным характеристикам - "I am" ("я есть"). Эти методы также называются биометрическими.

Нередко используются комбинированные схемы аутентификации, объединяющие методы разных классов. Например, двухфакторная *аутентификация* - *пользователь* предъявляет системе смарт-карту и вводит пин-код для ее активации.

Наиболее распространенными на данный момент являются парольные *системы аутентификации*. У пользователя есть *идентификатор* и *пароль*, т.е. секретная *информация*, известная только пользователю (и возможно - системе), которая используется для прохождения аутентификации.

В зависимости от реализации системы, *пароль* может быть одноразовым или многоразовым. Операционные системы, как правило, проводят аутентификацию с использованием многоразовых паролей. Совокупность идентификатора, пароля и, возможно, дополнительной информации, служащей для описания пользователя составляют **учетную запись пользователя**.

Если нарушитель узнал *пароль* легального пользователя, то он может, например, войти в систему под его учетной записью и получить *доступ* к конфиденциальным данным. Поэтому безопасности паролей следует уделять особой внимание.

Как отмечалось при рассмотрении стандарта *ISO 17799*, рекомендуется, чтобы пользователи системы подписывали документ о сохранении конфиденциальности пароля. Но нарушитель также может попытаться подобрать *пароль*, угадать его, перехватить и т.д. Рассмотрим некоторые рекомендации *по* администрированию парольной системы, позволяющие снизить *вероятность* реализации подобных угроз.

1. Задание минимальной длины используемых в системе паролей. Это усложняет атаку путем подбора паролей. Как правило, рекомендуют устанавливать минимальную длину в 6-8 символов.

2. Установка требования использовать в пароле разные группы символов - большие и маленькие буквы, цифры, специальные символы. Это также усложняет подбор.

3. Периодическая проверка *администраторами безопасности* качества используемых паролей путем *имитации атак*, таких как подбор паролей "по словарю" (т.е. проверка на использование в качестве пароля слов естественного языка и простых комбинаций символов, таких как "1234").

4. Установка максимального и минимального сроков жизни пароля, использование механизма принудительной смены старых паролей.

5. Ограничение числа неудачных попыток ввода пароля (блокирование учетной записи после заданного числа неудачных попыток войти в систему).

6. Ведение журнала истории паролей, чтобы пользователи, после принудительной смены пароля, не могли вновь выбрать себе старый, возможно скомпрометированный пароль.



Современные операционные системы семейства *Windows* позволяют делать установки, автоматически контролирующие выполнение требований 1,2,4-6. При

использовании домена *Windows*, эти требования можно распространить на все компьютеры, входящие в *домен* и таким образом повысить защищенность всей сети.

Но при внедрении новых механизмов защиты могут появиться и нежелательные последствия. Например, требования "сложности" паролей могут поставить в *тупик* недостаточно подготовленного пользователя. В данном случае потребуется объяснить, что с точки зрения операционной системы *Windows* надежный *пароль* содержит 3 из 4 групп символов (большие буквы, малые буквы, цифры, служебные знаки).

Аналогичным образом, надо подготовить пользователей и к внедрению блокировки учетных записей после нескольких неудачных попыток ввода пароля. Требуется объяснить пользователям, что происходит, а сами правила блокировки должны быть хорошо продуманы. Например, если высока *вероятность* того, что *пользователь* заблокирует свою *запись* в период отсутствия администратора, лучше ставить блокировку не насовсем, а на какой-то срок (30 минут, час и т.д.).

Это приводит к тому, что должна быть разработана политика *управления паролями*, сопровождающие ее документы, а потом уже проведено внедрение.

При использовании ОС семейства *Windows*, выявить учетные записи со слабыми или отсутствующими паролями можно, например, с помощью утилиты Microsoft *Baseline Security Analyzer*. Она же позволяет выявить и другие ошибки администрирования. Вопросам использования этой утилиты, а также настройке политики паролей посвящена лабораторная работа № 3.

**Работа с литературой:**

| Рекомендуемые источники информации (№ источника) | | | |
|---|---|---|---|
| Основная | Дополнительная | Методическая | Интернет-ресурсы |
| 1-2 | 1-3 | 1-2 | 1-3 |

**Оценочные средства:** собеседование.

## 3.4.ПРАКТИЧЕСКАЯ РАБОТА 4. Аутентификация по протоколу Kerberos.

**Цель работы:**

Изучение стандарта системы централизованной аутентификации и распределения ключей в рамках операционной системы Windows.

**Содержание:**

Аутентификация с использованием протокола Kerberos v.5. Централизованное распределение ключей симметричного шифрования. Серверная часть Kerberos как центр распределения ключей. Протокол Kerberos. Взаимодействие между Kerberos-областями.

**Теоретическая часть.**

Протокол Kerberos был разработан в Массачусетском технологическом институте в середине 1980-х годов и сейчас является фактическим стандартом системы централизованной аутентификации и *распределения ключей* симметричного шифрования. Поддерживается операционными системами семейства Unix, *Windows* (начиная с *Windows*'2000), есть реализации для *Mac* OS.

В сетях *Windows* (начиная с *Windows*'2000 Serv.) *аутентификация по* протоколу Kerberos v.5 (*RFC* 1510) реализована на уровне доменов. Kerberos является основным протоколом аутентификации в домене, но в целях обеспечения совместимости с с предыдущими версиями, также поддерживается протокол *NTLM*.

Перед тем, как рассмотреть порядок работы Kerberos, разберем зачем он изначально разрабатывался. **Централизованное** *распределение ключей* симметричного шифрования подразумевает, что у каждого абонента сети есть только один основной *ключ*, который используется для взаимодействия с *центром распределения ключей* (сервером ключей). Чтобы получить *ключ* шифрования для защиты обмена данными с другим абонентом, *пользователь* обращается к серверу ключей, который назначает этому пользователю и соответствующему абоненту сеансовый *симметричный ключ*.

Протокол Kerberos обеспечивает *распределение ключей* симметричного шифрования и проверку подлинности пользователей, работающих в незащищенной сети. Реализация Kerberos - это программная система, построенная *по* архитектуре "клиент-*сервер*". Клиентская часть устанавливается на все компьютеры защищаемой сети, кроме тех, на которые устанавливаются компоненты сервера Kerberos. В роли клиентов Kerberos могут, в частности, выступать и сетевые серверы (файловые серверы, *серверы печати* и т.д.).

Серверная часть Kerberos называется *центром распределения ключей* (англ. Key *Distribution Center*, сокр. *KDC*) и состоит из двух *компонент*:

- сервер аутентификации (англ. *Authentication Server*, сокр. AS);
- сервер выдачи разрешений (англ. *Ticket* Granting Server, сокр. *TGS*).

Каждому субъекту сети *сервер* Kerberos назначает разделяемый с ним *ключ* симметричного шифрования и поддерживает базу данных субъектов и их секретных ключей. Схема функционирования протокола Kerberos представлена на рис. 4.1.

Пусть клиент C собирается начать взаимодействие с сервером SS (англ. *Service Server - сервер*, предоставляющий сетевые сервисы). В несколько упрощенном виде, протокол предполагает следующие шаги:

1. C→AS: {c}.

Клиент C посылает серверу аутентификации AS свой идентификатор c (идентификатор передается открытым текстом).

2. AS→C: $\{K_{C\_TGS}, \{TGT\}K_{AS\_TGS}\}K_C$,

где:

- $K_C$ - основной ключ C ;

- $K_{C\_TGS}$ - ключ, выдаваемый C для доступа к серверу выдачи разрешений *TGS* ;
- {TGT} - *Ticket* Granting *Ticket* - билет на доступ к серверу выдачи разрешений

{TGT}={c,*tgs*,$t_1$,$p_1$, $K_{C\_TGS}$}, где *tgs* - идентификатор сервера выдачи разрешений, $t_1$ - отметка времени, $p_1$ - *период действия* билета.

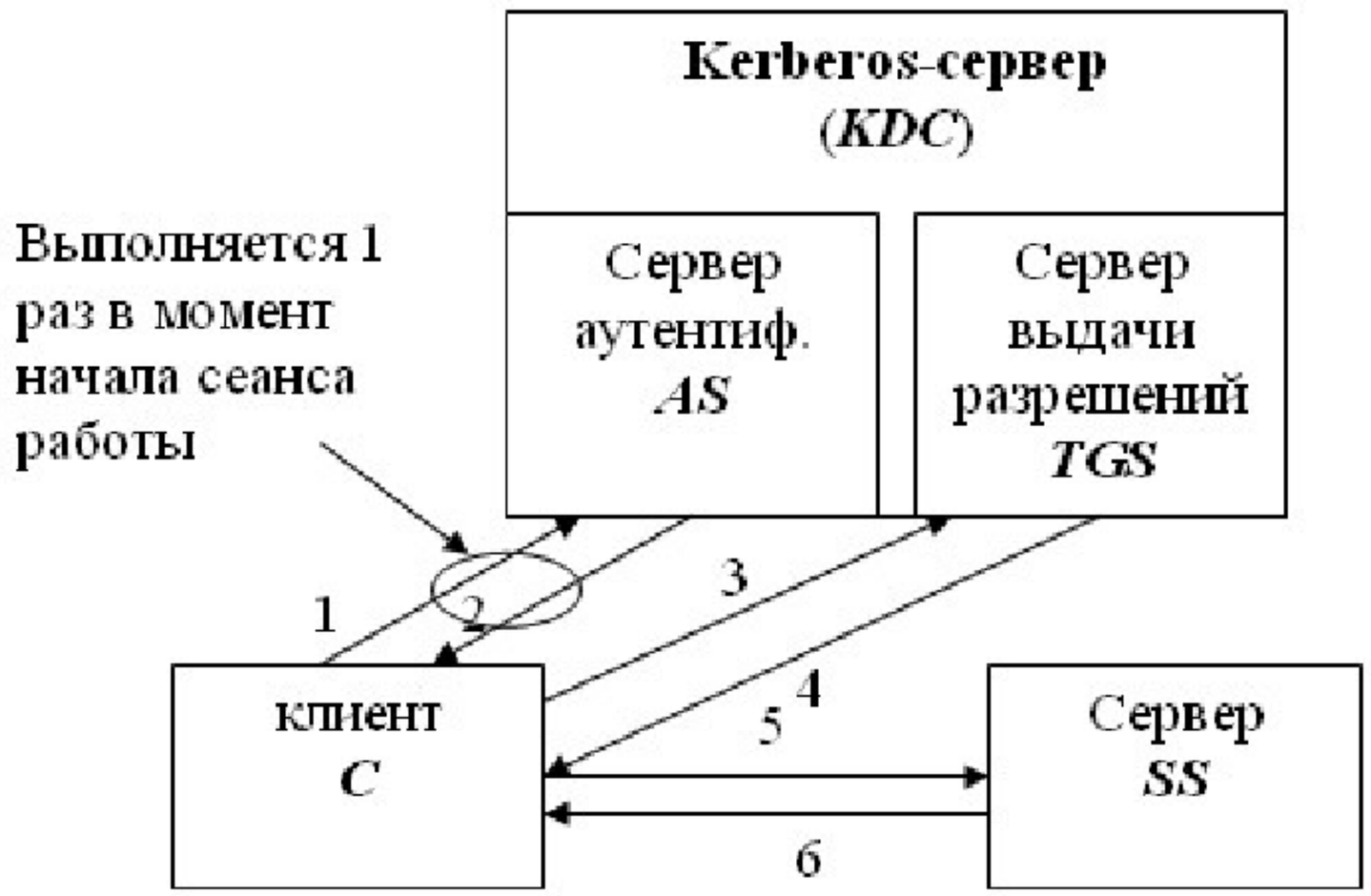



Рис. 4.1. Протокол Kerberos

Запись $\{\cdot\}K_X$ здесь и далее означает, что содержимое фигурных скобок зашифровано на ключе $K_X$.

На этом шаге сервер аутентификации AS, проверив, что клиент C имеется в его базе, возвращает ему билет для доступа к серверу выдачи разрешений и ключ для взаимодействия с сервером выдачи разрешений. Вся посылка зашифрована на ключе клиента C. Таким образом, даже если на первом шаге взаимодействия идентификатор c послал не клиент C, а нарушитель X, то полученную от AS-посылку X расшифровать не сможет.

Получить доступ к содержимому билета TGT не может не только нарушитель, но и клиент C, т.к. билет зашифрован на ключе, который распределили между собой сервер аутентификации и сервер выдачи разрешений.

3. C→*TGS*: {TGT}$K_{AS\_TGS}$, {$Aut_1$} $K_{C\_TGS}$, {ID}

где {$Aut_1$} - аутентификационный блок - $Aut_1$ = {c,$t_2$}, $t_2$ - метка времени; ID - идентификатор запрашиваемого сервиса (в частности, это может быть идентификатор сервера SS).

Клиент C на этот раз обращается к серверу выдачи разрешений TGS. Он пересылает полученный от AS билет, зашифрованный на ключе $K_{AS\_TGS}$, и аутентификационный блок, содержащий идентификатор c и метку времени, показывающую, когда была сформирована посылка. Сервер выдачи разрешений расшифровывает билет TGT и получает из него информацию о том, кому был выдан билет, когда и на какой срок, ключ шифрования, сгенерированный сервером AS для взаимодействия между клиентом C и сервером *TGS*. С помощью этого ключа расшифровывается аутентификационный блок. Если метка в блоке совпадает с меткой в билете, это доказывает, что посылку сгенерировал на самом деле C (ведь только он знал ключ $K_{C\_TGS}$ и мог правильно зашифровать свой идентификатор).

Далее делается проверка времени действия билета и времени оправления посылки **3** ). Если проверка проходит и действующая в системе политика позволяет клиенту C обращаться к клиенту SS, тогда выполняется шаг **4** ).

4. *TGS*→C: $\{\{TGS\}K_{TGS\_SS},K_{C\_SS}\}K_{C\_TGS}$,

где $K_{C\_SS}$ - ключ для взаимодействия C и SS, {*TGS*} - *Ticket* Granting Service - билет для доступа к SS (обратите внимание, что такой же аббревиатурой в описании протокола обозначается и сервер выдачи разрешений). $\{TGS\} =\{c,ss,t_3,p_2, K_{C\_SS}\}$.

Сейчас сервер выдачи разрешений *TGS* посылает клиенту C ключ шифрования и билет, необходимые для доступа к серверу SS. Структура билета такая же, как на шаге 2): идентификатор того, кому выдали билет; идентификатор того, для кого выдали билет; отметка времени; *период действия*; ключ шифрования.

5. C→SS: $\{TGS\}K_{TGS\_SS}$, $\{Aut_2\}\ K_{C\_SS}$

где $Aut_2=\{c,t_4\}$.

Клиент C посылает билет, полученный от сервера выдачи разрешений, и свой аутентификационный блок серверу SS, с которым хочет установить сеанс защищенного взаимодействия. Предполагается, что SS уже зарегистрировался в системе и распределил с сервером *TGS* ключ шифрования $K_{TGS\_SS}$. Имея этот ключ, он может расшифровать билет, получить ключ шифрования $K_{C\_SS}$ и проверить подлинность *отправителя сообщения*.

6. SS→C: $\{t_4+1\}K_{C\_SS}$

Смысл последнего шага заключается в том, что теперь уже SS должен доказать C свою подлинность. Он может сделать это, показав, что правильно расшифровал предыдущее сообщение. Вот поэтому, SS берет отметку времени из аутентификационного блока C, изменяет ее заранее определенным образом (увеличивает на 1), шифрует на ключе $K_{C\_SS}$ и возвращает C.

Если все шаги выполнены правильно и все проверки прошли успешно, то стороны взаимодействия C и SS, во-первых, удостоверились в подлинности друг друга, а во-вторых, получили *ключ* шифрования для защиты сеанса связи - *ключ* $K_{C\_SS}$.

Нужно отметить, что в процессе сеанса работы клиент проходит шаги 1) и 2) только один раз. Когда нужно получить билет на *доступ* к другому серверу (назовем его SS1 ), клиент C обращается к серверу выдачи разрешений *TGS* с уже имеющимся у него билетом, т.е. протокол выполняется начиная с шага 3).

При использовании протокола Kerberos компьютерная *сеть* логически делится на области действия серверов Kerberos. Kerberos-область - это участок сети, пользователи и серверы которого зарегистрированы в базе данных одного сервера Kerberos (или в одной базе, разделяемой несколькими серверами). Одна область может охватывать сегмент локальной сети, всю локальную *сеть* или объединять несколько связанных локальных сетей. Схема взаимодействия между Kerberos-областями представлена на рис. 4.2.

Для взаимодействия между областями, должна быть осуществлена взаимная *регистрация* серверов Kerberos, в процессе которой *сервер* выдачи разрешений одной области регистрируется в качестве клиента в другой области (т.е. заносится в базу сервера аутентификации и разделяет с ним *ключ*).

После установки взаимных соглашений, клиент из области 1 (пусть это будет $K_{11}$ ) может установить *сеанс* взаимодействия с клиентом из области 2 (например, $K_{21}$ ). Для этого $K_{11}$ должен получить у своего сервера выдачи разрешений билет на *доступ* к Kerberos-серверу, с клиентом которого он хочет установить взаимодействие (т.е. серверу Kerberos KDC2). Полученный билет содержит отметку о том, в какой области зарегистрирован владелец билета. Билет шифруется на ключе, разделенном между серверами KDC1 и KDC2. При успешной расшифровке билета, удаленный Kerberos-*сервер* может быть уверен, что билет выдан клиенту Kerberos-области, с которой установлены *доверительные отношения*. Далее протокол работает как обычно.

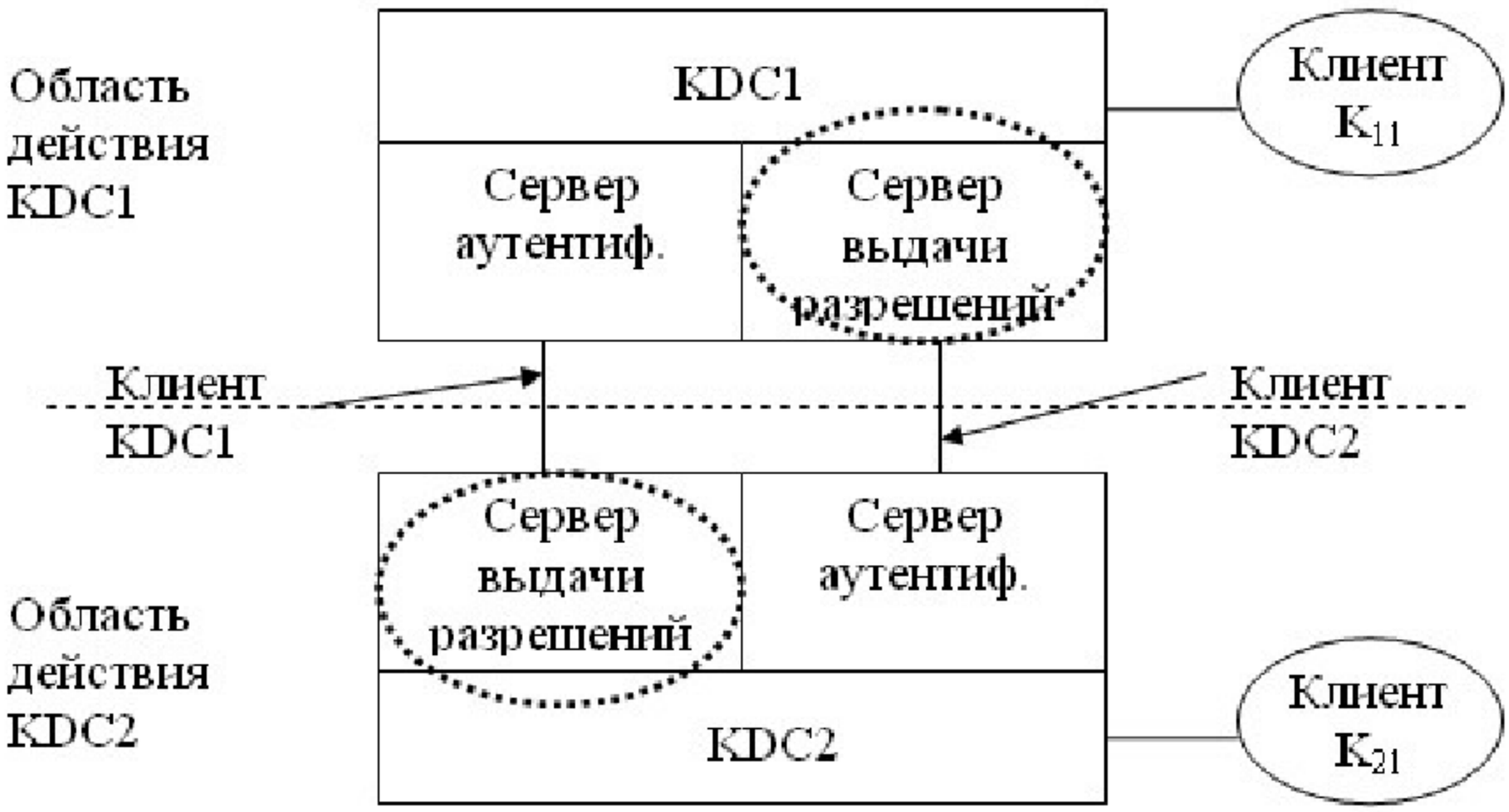



Рис. 4.2. Взаимодействие между Kerberos-областями

Кроме рассмотренных, Kerberos предоставляет еще ряд дополнительных возможностей. Например, указанный в структуре билета *параметр* p(период времени) задается парой значений "время начала действия" - "время окончания действия", что позволяет получать билеты *отложенного действия*.

Имеется тип билета "с правом передачи", что позволяет, например, серверу выполнять действия от имени обратившегося к нему клиента.

Два слова об аутентификации. Если Kerberos - протокол *распределения ключей*, корректно ли использовать его для аутентификации?! Но как отмечалось выше, если все стадии протокола прошли успешно, взаимодействующие стороны не только распределили *ключ*, но и убедились в подлинности друг друга, иными словами - аутентифицировали друг друга.

Что касается реализации протокола Kerberos в *Windows*, то надо отметить следующее.

1. Ключ пользователя генерируется на базе его пароля. Таким образом, при использовании слабых паролей эффект от надежной защиты процесса аутентификации будет сведен к нулю.

2. В роли Kerberos-серверов выступают контроллеры домена, на каждом из которых должна работать служба Kerberos Key Distribution Center (*KDC*). Роль хранилища информации о пользователях и паролях берет на себя служба каталога Active Directory. Ключ, который разделяют между собой сервер аутентификации и сервер выдачи разрешений формируется на основе пароля служебной учетной записи **krbtgt** - эта запись автоматически создается при организации домена и всегда заблокирована.

3. Microsoft в своих ОС использует расширение Kerberos для применения криптографии с открытым ключом. Это позволяет осуществлять регистрацию в домене и с помощью смарт-карт, хранящих ключевую информацию и цифровой *сертификат пользователя*.

4. Использование Kerberos требует синхронизации внутренних часов компьютеров, входящих в домен Windows.

Для увеличения уровня защищенности процесса аутентификации пользователей, рекомендуется отключить использование менее надежного протокола *NTLM* и оставить только Kerberos (если использования *NTLM* не требуют устаревшие клиентские ОС).

Кроме того, рекомендуется запретить администраторским учетным записям получать билеты "с правом передачи" (это убережет от некоторых угроз, связанных автоматическим запуском приложений от имени таких записей).

**Контрольные вопросы:**

1. Центр распределения ключей (серверная часть Kerberos).
2. Порядок взаимной *регистрации* серверов Kerberos.
3. Основные этапы реализации протокола Kerberos в *Windows*.
4. Служба Kerberos Key Distribution Center.
5. Как осуществляется взаимодействие между Kerberos-областями.

**Работа с литературой:**

| Рекомендуемые источники информации (№ источника) | | | |
|---|---|---|---|
| Основная | Дополнительная | Методическая | Интернет-ресурсы |
| 1-2 | 1-3 | 1-2 | 1-3 |

**Оценочные средства:** собеседование.

## 3.5.ПРАКТИЧЕСКАЯ РАБОТА 5. Выявление уязвимостей с помощью Microsoft Baseline Security Analyzer.

**Цель работы:**

Изучить программу Microsoft Baseline Security analyzer, позволяющую проверить уровень безопасности установленной конфигурации операционной системы *Windows*.

**Содержание:**

Системы анализа защищенности операционных систем. Порядок установки Service pack. Запуск утилиты в режиме проверки обновлений. Уязвимости, связанные с администрированием операционной системы.

Microsoft *Baseline Security analyzer - программа*, позволяющая проверить уровень безопасности установленной конфигурации операционной системы (ОС) *Windows 2000*, XP, *Server* 2003, Vista *Server* 2008. Также проверяется и ряд других приложений разработки Microsoft. Данное средство можно отнести к разряду систем *анализа защищенности*. Оно распространяется бесплатно и доступно для скачивания с *web-сервера* Microsoft (*адрес* страницы данной утилиты на момент подготовки описания был: http://technet.microsoft.com/ru-ru/security/cc184924(en-us).aspx).

В процессе работы *BSA* проверяет наличие обновлений безопасности операционной системы, офисного пакета Microsoft Office(для версий XP и более поздних), серверных приложений, таких как MS *SQL Server*, MS *Exchange Server*, *Internet Information Server* и т.д. Кроме того, проверяется ряд настроек, касающихся безопасности, например, действующая политика паролей.

Перейдем к знакомству с программным продуктом. При запуске открывается окно, позволяющее выбрать *объект* проверки - один *компьютер* (выбирается *по* имени или ip-адресу), несколько (задаваемых диапазоном IP-адресов или доменным именем) или просмотреть ранее сделанные отчеты сканирования системы. При выборе сканирования отдельного компьютера *по* умолчанию подставляется имя локальной станции, но можно указать имя или *IP-адрес* другого компьютера.

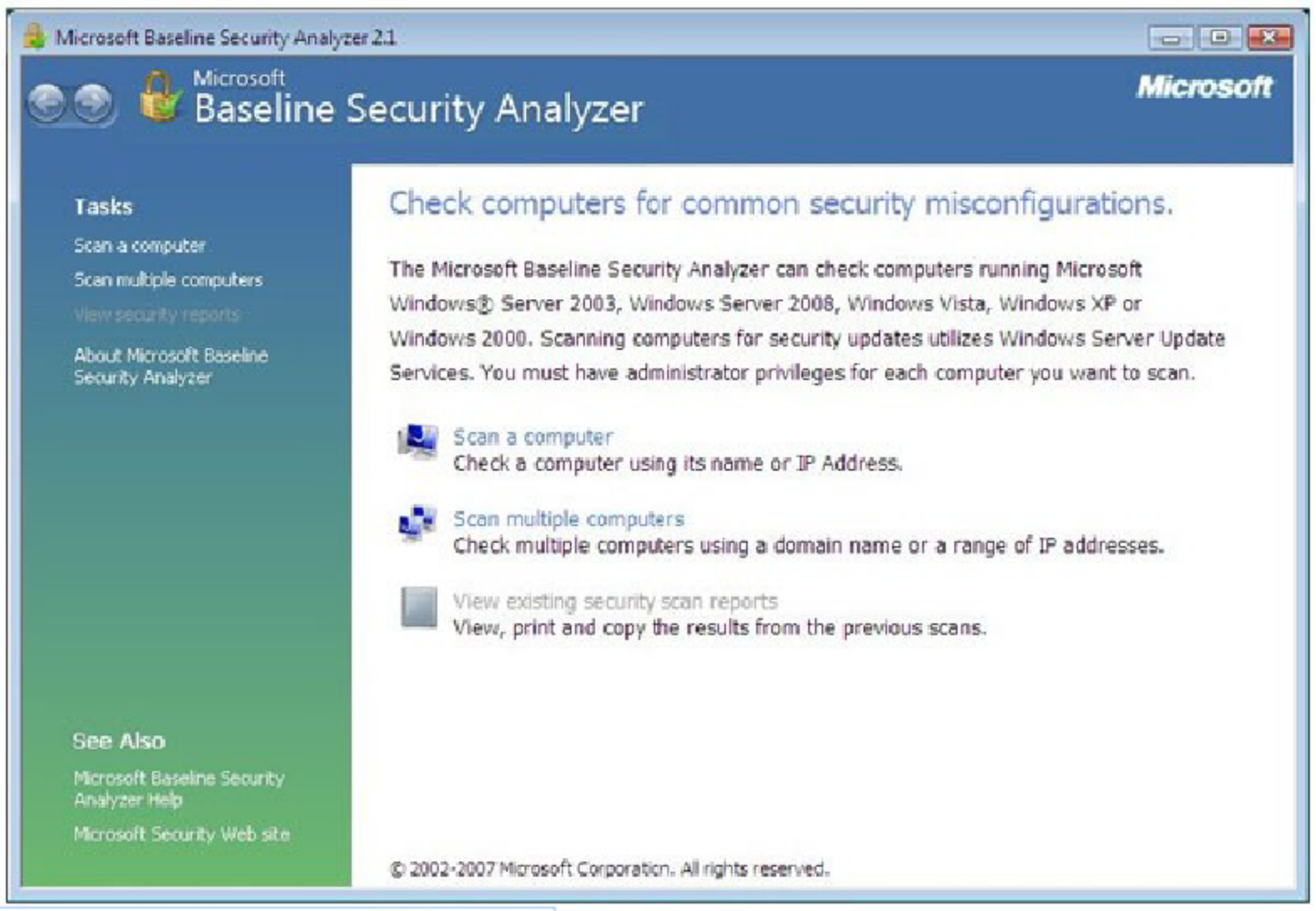



Рис. 5.1. Выбор проверяемого компьютера

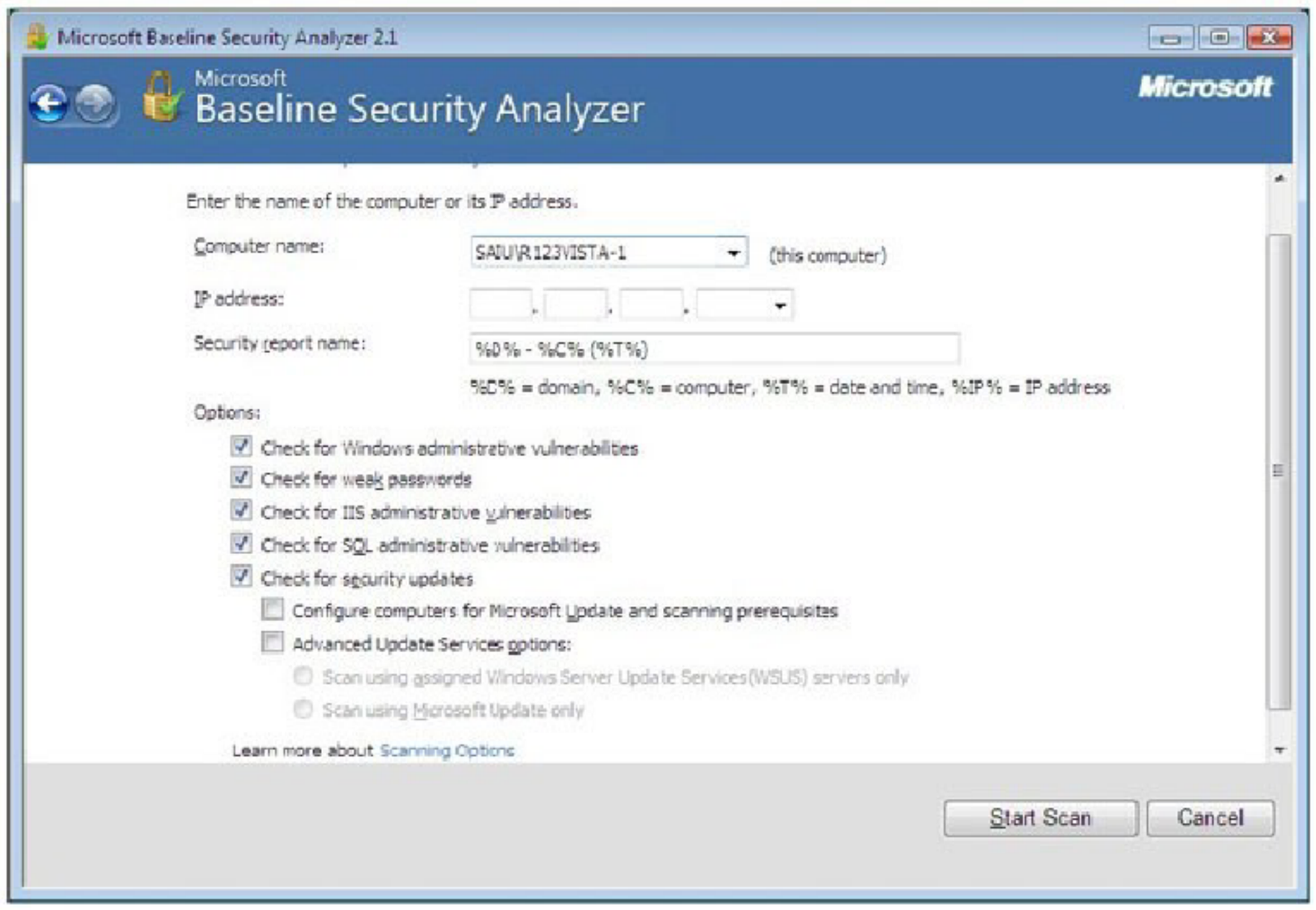



Рис. 5.2. Задание параметров проверки

Можно задать перечень проверяемых параметров. На рис. 5.2 представлен выбор вариантов проверки:

• проверка на наличие уязвимостей Windows, вызванных некорректным администрированием;

• проверка на "слабые" пароли (пустые пароли, отсутствие ограничений на срок действия паролей и т.д.);

• проверка на наличие уязвимостей web-сервера IIS, вызванных некорректным администрированием;

• аналогичная проверка в отношении СУБД MS SQL Server;

• проверка на наличие обновлений безопасности.

Перед началом работы *программа* обращается на *сервер* Microsoft для получения перечня обновлений для ОС и известных уязвимостей. Если на момент проведения проверки *компьютер* не подключен к *Интернет*, база уязвимостей не будет обновлена, *программа* об этом сообщит и дальнейшие проверки выполняться не будут. В подобных случаях нужно отключать проверку обновлении безопасности (сбросив соответствующую галочку на экране рис. 3.2 или с помощью ключа при использовании *утилиты командной строки*, о чем речь пойдет ниже).

Для успешной проверки локальной системы необходимо, чтобы *программа* выполнялась от имени учетной записи с правами локального администратора. Иначе проверка не может быть проведена и о чем будет выдано сообщение: "You do *not* have sufficient *permissions* to*perform this command*. Make sure that you are *running as* the *local administrator* or have opened the *command prompt* using the 'Run *asadministrator' option*".

*По* результатам сканирования формируется отчет, в начале которого дается общая оценка уровня безопасности конфигурации проверяемого компьютера. В приведенном на рис. 5.3 примере уровень риска оценивается как "серьезный" (Severe *risk*).

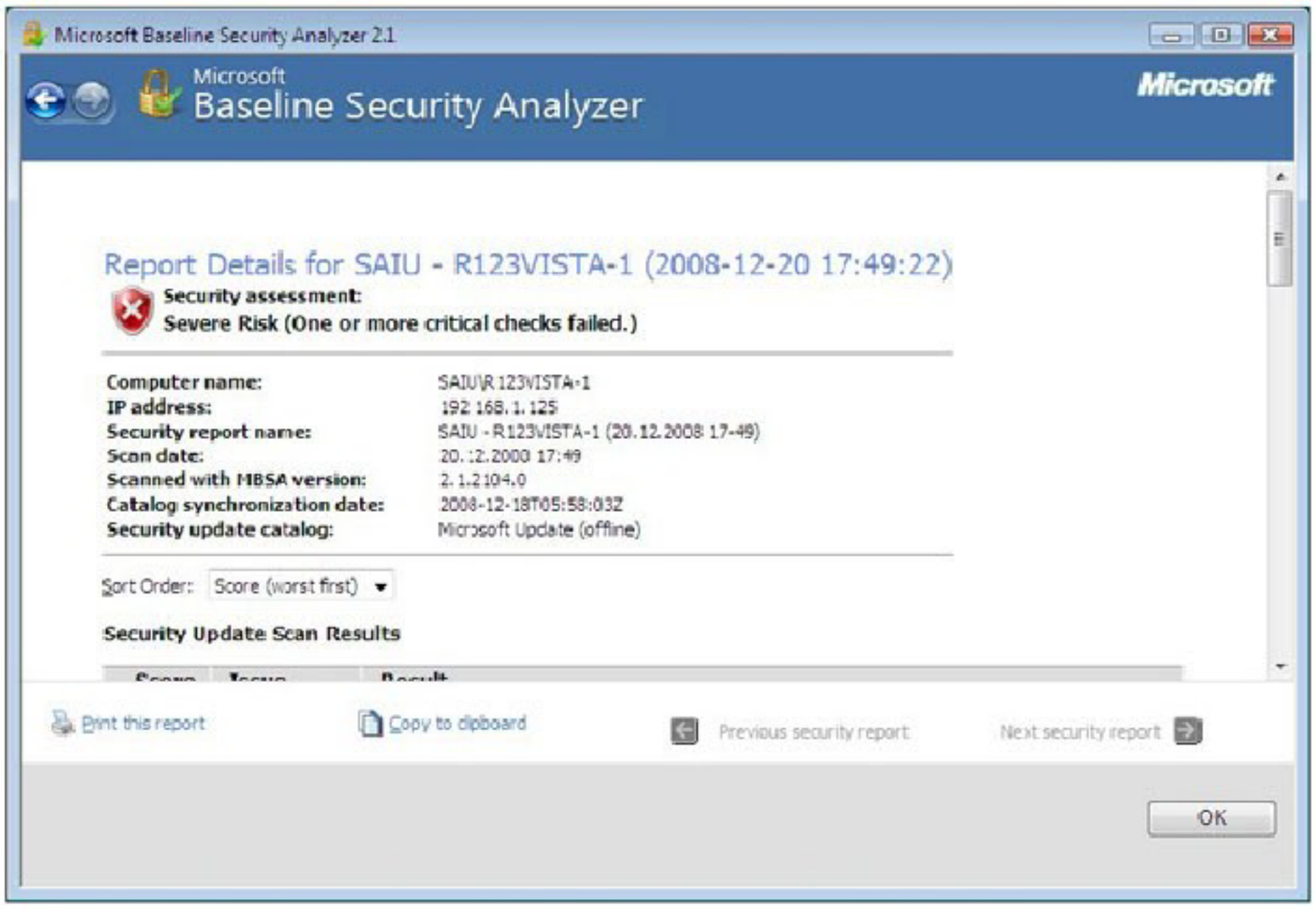



Рис. 5.3. Заголовок отчета

Далее приводится перечень обнаруженных уязвимостей, разбитый на группы: результаты проверки установки обновлений, результаты проверки *Windows* и т.д. Надо отметить, что выпускаемые Microsoft обновления бывают различных типов:

**Security updates** - собственно обновления безопасности, как правило, посвященные исправлению одной уязвимости программного продукта;

**Update rollups** - набор исправлений безопасности, который позволяет одновременно исправить несколько уязвимостей. Это упрощает обслуживание процесса обновления программного обеспечения (*ПО*);

**Service packs** - набор исправлений, как связанных, так и несвязанных с *безопасностью*. *Установка Service* pack, как правило, исправляет все уязвимости, обнаруженные с момента выхода предыдущего *Service* pack, таким образом устанавливать промежуточные обновления уже не надо.

В описании рассматриваемого результата проверки (рис. 5.4) можно выбрать ссылку **Result details** и получить более подробное описание найденных проблем данной группы. При наличии подключения к *Интернет*, перейдя *по* приводимой в отчете ссылке, можно получить информацию об отсутствующем обновлении безопасности и скачать его из сети.

Нужно отметить, что установка обновлений для систем с высокими требованиями в области непрерывности работы, требует предварительной тщательной проверки совместимости обновлений с использующимися приложениями. Подобная проверка обычно производится на тестовых системах с близкой конфигурацией *ПО*. В то же время, для небольших организаций и пользователей домашних компьютеров такая проверка зачастую неосуществима. Поэтому надо быть готовым к тому, чтобы восстановить систему после неудачного обновления. Для современных ОС семейства *Windows* это можно сделать, например, используя специальные режимы загрузки ОС - безопасный режим или режим загрузки последней удачной конфигурации.

Также надо отметить еще одну особенность. На данный момент **baseline security analyzer** не существует в локализованной русскоязычной версии. И содержащиеся там ссылки на пакеты обновлений могут указывать на иные языковые версии, что может создать проблемы при обновлении локализованных продуктов.