**Задание на 09.02.2022**

МДК 03.02 Программно-аппаратные средства защиты информации

1. **Сделать конспект лекции**
2. **Сделать практическую работу №19 Пакетные фильтры и межсетевые экраны**
3. **Отчет отправить на почту** [**svebalch@mail.ru**](mailto:svebalch@mail.ru)
4. **СРС:** [**https://stepik.org/course/94299/promo**](https://stepik.org/course/94299/promo) **пройти онлайн курсы на платформе Степик. Тема: Кибербезопасность**

**Лекция 19**

**Метод генерации изолированной программной среды**

Для создания гарантированно защищенной КС (в смысле выполнения заданной политики безопасности) необходимо:

1. Убедиться в попарной корректности субъектов, замыкаемых в ИПС (либо убедиться в корректности любого субъекта относительно МБО и МБС);

2. Спроектировать и реализовать программно (или программно-аппаратно) МБС так, чтобы:

— для любого субъекта и любого объекта производился контроль порождения субъектов (т.е. чтобы реализация МБС соответствовала его определению);

— порождение любого субъекта происходило с контролем неизменности объекта-источника.

3. Реализовать МБО в рамках априорно сформулированной политики безопасности.

Важную роль при проектировании ИПС играет свойство КС, заключающееся в поэтапной активизации субъектов из объектов различного уровня представления информации. Рассмотрим в таблице 4.1 иерархию уровней при загрузке операционной системы.



В общем случае можно говорить о рекурсивной структуре объектов некоторого уровня, вмещающей объекты предыдущего уровня. На нулевом уровне первичный объект (элементарная структура нижнего уровня) в таблице 1 соответствует термину «сектор».

С учетом иерархической структуры представления объектов можно говорить о том, что в начальные этапы активизации КС декомпозиция на субъекты и объекты динамически изменяется. Следовательно, основная теорема ИПС может быть примёнима только на отдельных интервалах времени, когда уровень представления объектов постоянен и декомпозиция фиксирована. Можно утверждать, что ППС, действующую от момента активизации до момента окончания работы КС, невозможно сформировать в начальный момент активизации КС.

Пусть в КС выделяется конечное число уровней представления объектов *U*={0,….,*R*}, *R* — максимальный уровень представления объекта.

С точки зрения выполнения условий безопасности имело бы смысл говорить о некотором «стационарном» состоянии КС, когда в отображениях ***Stream*** и ***Create*** участвуют только объекты уровня *R*. Тогда реализация МБС может быть значительно упрощена (в том смысле, что все аргументы-объекты операции ***Create*** имеют тот же уровень).

Необходимо обратить внимание на то, что такое требование, с одной стороны, может накладывать ограничительные условия на свойства прикладного ПО (невозможность инициирования потоков, включающих объекты уровня менее *R*, прикладными программами), а с другой стороны, быть следствием проектировочных решений реализации субъекта, локализованного в ядре операционной системы (примером является ОС Windows 4.0, запрещающая операции ниже уровня «файл» со стороны субъектов прикладного уровня).

Практическая реализация всех операционных систем позволяет выделить две фазы их работы: активизация субъектов с ростом уровня представления объектов (фаза загрузки или начальная фаза) и фаза стационарного состояния (когда уровень представления объектов не увеличивается). Конечно, необходимо сделать оговорку, касающуюся возможности реализации Потоков к объектам нижнего уровня (операционные системы типа DOS, в которых возможна операция с любым объектом нижнего уровня (сектор) из программ прикладного уровня).Тогда практическая реализация ИПС может состоять из двух этапов: предопределенное выполнение начальной фазы, включающее в себя момент активизации МБС (и МБО), и работа в стационарной фазе в режиме ИПС (возможно, с, контролем неизменности объектов-источников).

Введем понятие последовательности активизации компонент КС. Смысл вводимых понятий и формулируемых ниже утверждений состоит в необходимости приведения субъектов КС в одно и то же состояние после активизации первичного субъекта аппаратно-программного уровня, или, иначе говоря, в задании предопределенной последовательности активизации субъектов КС.

Обозначим: *Zl*— последовательность пар (*i, j*)*t* ( *t*= 0, 1, 2,..., 1-1 — моменты времени) длины 1, такие, что *Create* (S*i*, О*j*)[*t*]->S*m* [*t*+1].

Обозначим также:

*Sz* - множество всех субъектов, включенных в последовательность *Zl*,

*Оz -* множество всех объектов, включенных в последовательность *Zl*.

Для многопотоковых КС можно рассматривать несколько (возможно, зависимых друг от друга) последовательностей *Zl* и соответственно множеств *Sz* и *Оz*.

Состоянием КС в момент времени *t* называется упорядоченная совокупность состояний субъектов.

Напомним, что каждый объект есть слово в априорно определенном языке, а понятие состояния субъекта сформулировано выше.

Состояние КС в моменты времени tx1 и tх2 (tx1 и tх2 исчисляются для двух отрезков активности КС от нулевого момента активизации КС t01 и t02 — например, включения питания аппаратной части) одинаково, если:

1. tx1=tx2,

2. тождественны субъекты *Si* [t01] и *Si* [t02],

З. неизменны все объекты из множества *Оz,*

4. неизменна последовательность Z L.

Необходимо заметить, что последовательность *Zl* локализуется в некотором объекте либо совокупности объектов (например, для DOS последовательность активизации субъектов предопределена содержанием файлов АUТОЕХЕС.ВАТ и СОNFIG.SYS) и неизменность последовательности *Zl* тождественна неизменности указанных объектов, для ОС Windows NT последовательность активизации компонент определена содержанием соответствующих ключей реестра (registry).

Пусть в последовательности *Zl* можно выделить *zi* такое, что для любого *Zk, k> i* отображений ***Stream*** и ***Create*** используют только объекты уровня *R*. Другими словами, с момента времени *i* наступает стационарная фаза функционирования КС.

В этих условиях, а также при попарной корректности субъектов и действии МБС с контролем неизменности объектов- источников на уровне *R* с момента времени *m > k* верно:

При условии неизменности *Zl* и неизменности объектов из *Оz* в КС с момента времени установления неизменности *Zl* и *Оz* действует изолированная программная среда.

Рассмотрим процесс практического проектирования защищенного фрагмента КС.

Первоначально необходимо убедиться в выполнении условий корректности или абсолютной корректности для субъектов, участвующих в порождении ИПС. Указанные субъекты в основном могут быть локализованы на уровне программно-аппаратной компоненты ЭВМ (программы ПЗУ, загрузчики операционных сред), т.е. работать на уровне, близком к взаимодействию с оборудованием КС, либо на уровне операционной средs. Доказательство корректности субъектов программно-аппаратного уровня значительно отличается от соответствующих доказательств для субъектов прикладного уровня. В связи с этим выделим проверку условий корректности субъектов в два шага. Шагом 1 назовем доказательство корректности субъектов программно-аппаратного уровня. Понятие модуль обозначает реализацию объекта источника, а совокупность субъекта, порожденного из объекта источника и всего множества ассоциированных с этим субъектом объектов в течение всего времени существования субъекта, называете 1 как правило, процессом (или задачей, заданием).

Далее необходимо определить состав программных средств базовой вычислительной среды, т.е. определить конкретную операционную среду, дополнительные программные средства сервиса (например программные оболочки или средства теле - коммуникации) и программные средства поддержки дополнительного оборудования (программы управления принтером и др.). После этого наступает самый трудоемкий этап (Шаг 2), на котором необходимо убедиться в корректности субъектов описанного базового набора программных средств. При этом важно заметить следующее.

В составе ПО КС не должно быть целого класса возможностей — назовем их инструментальными. Прежде всего - это возможность изменения состояния ассоциированных объектов со стороны субъекта (например, изменение содержимого оперативной памяти) других субъектов (изменение содержания подразумевает существование операций Stream типа запись), возможность инициирования и прекращения выполнения процессов нестандартным образом (помимо механизмов операционной среды). Кроме того, при реализации МБС и МБО на стационарной фазе функционирования КС необходимо отсутствие в любых субъектах, замкнутых в ИПС, операций порождения потоков Stream к объектам уровня k <R.

Обобщенно достаточные условия к базовому набору ПО можно сформулировать следующим утверждением.

для того чтобы ИПС поддерживалась в течение всего времени активности КС, достаточно, чтобы в составе программного обеспечения, могущего быть инициированным в ИПС, не было функций порождения субъектов и прекращения их работы, кроме заранее предопределенных при реализации МБС, и не существовало возможностей влияния на среду выполнения (под средой выполнения понимается множество ассоциированных объектов) любого процесса, а также инициирования потоков к объектам логического уровня менее R.

Легко видеть, что данное утверждение есть собранные воедино условия выполнения приводимых выше утверждений. Поясним требование невозможности прекращения выполнения процесса каким-либо иным образом, кроме предопределенного. В данном случае необходимо учитывать, что во множестве субъектов, замкнутых в ИПС, выделены два особых субъекта — МБС и МБО. Прекращение существования МБС означает нарушение условия замкнутости среды, а прекращение существования МБО означает допустимость потоков множества Л т.е. несанкционированный доступ. Шаг З заключается в проектировании и разработке программных или программно-аппаратных средств защиты в КС, а затем и их тестировании. Он подразумевает проектирование и реализацию в заданном множестве субъектов МБС и МБО.

Практически шаги 1—З могут быть выполнены исходя из описанных в литературе методик разработки и тестирования ПО.

Шаг 4 заключается в «замыкании» всего комплекса программного обеспечения, включая, и средства защиты, в изолированную программную среду.

Итак, показано, что основными элементами поддержания изолированности программной среды являются контроль целостности и контроль порождения процессов.

Выше мы уже сформулировали понятия МБС и порождения субъектов с контролем их неизменности. Необходимо заметить, что для достоверного контроля неизменности объекта (т.е. с вероятностью ошибки, равной 0) необходимо убедиться в полном тождестве проверяемого объекта и образца. Из этого следует, что эталон должен содержать не меньше информации, чем проверяемый объект. Из этого в свою очередь следует, что эталонный объект должен быть как минимум одинаковой длины с Проверяемым. На практике такой подход может быть применен е серьезными ограничениями (например, для объектов небольшого объема типа программ ПЗУ или загрузчиков ОС).

В связи с этим для контроля целостности применяют объекты, содержащие информацию, зависящую от всего содержания объекта, но тем не менее значительно меньшего объема, вычисленную при помощи класса функций типа «хэш». Очевидно, Что в этом случае процесс установления неизменности объекта становится вероятностным.

Исходя из данного факта невозможно говорить о гарантированных (детерминированно) свойствах системы, поскольку Неизменность объекта гарантируется лишь с некоторой вероятностью, не равной 1. Следовательно, все условия утверждений выполняются с некоторой вероятностью, зависящей от свойств применяемых для контроля целостности хэш-функций. Для подчеркивания изменившихся условий будем говорить далее не о контроле неизменности объекта, а о контроле целостности (КЦ) объекта.

Необходимо отметить также, что в процедуре контроля неизменности (которая теперь принимает вероятностный характер) участвуют как минимум два объекта: объект контроля и эталонный объект (хэш-значение), а также субъект, реализующий хэш-функцию и производящий сравнение.

Поэтому для субъекта контроля целостности важным является выполнение следующих условий:

— качественный алгоритм контроля целостности (термин «качественный» будет пояснен ниже);

— контроль реальных данных (т.е. отображение состояния контролируемого и эталонного объектов в ассоциированные объекты-данные субъекта контроля целостности, совпадающие с тождественным).

Поясним подробнее второй пункт. Контроль целостности всегда сопряжен с чтением данных (т.е. с инициированием потоков от объектов к ассоциированным объектам-данным субъекта контроля целостности, причем потоки могут соответствовать различному уровню представления информации — чтение по секторам, по файлам и т.д.). Например, встроенный в BIOS ПЭВМ субъект (практически это программная закладка — см. ниже) может навязывать при чтении вместо одного сектора другой или редактировать непосредственно буфер, в котором были прочитаны данные. Аналогичный эффект может быть вызван субъектами операционной среды, например, субъектами, локализованными в первичных загрузчиках ОС. С другой стороны, даже контроль самого ВIОS может происходить «под наблюдением» какой-либо дополнительной аппаратуры и не показать его изменения. Аналогичные эффекты могут возникать и при обработке файла. Цель организации режима чтения реальных данных состоит в тождественном отображении параметров чтения на АО субъекта чтения (поток от АО субъекта КЦ к АО субъекта чтения) и тождественном отображении считываемого объекта (в соответствии с параметрами, переданными субъекту чтения) на ассоциированные объекты-данные субъекта КЦ.

Поясним теперь понятие качественного КЦ с точки зрения математических свойств функции КЦ. Предположим, что имеется некоторый объект *F* и некоторый алгоритм Н, преобразующий объект *F* в некоторый объект М, который представляется словом того же языка, но меньшей длины. Этот алгоритм таков, что при случайном равновероятном выборе двух объектов *F* 1 и *F* 2 из множества возможных соответствующие им объекты *М*1 = *Н*(*F* 1) и *М*2 = *Н*(*F* 2) с высокой вероятностью различны. Тогда проверка целостности данных строится так: рассматриваем объект *F*, по известному алгоритму *Н* строим *К = Н*(*F*) и сравниваем *М*, заранее вычисленное как *М = Н(F),* с К. При совпадении считаем объект неизменным. Алгоритм *Н* называют, как правило, хэш-функцией или (реже) контрольной суммой, а число *М*— хэш-значением.

Качество КЦ определяется в данном случае выполнением следующих условий:

1. По известному объекту *М = Н(F)* нахождение другого объекта *G*, не тождественного *F*, такого, что *М = Н(G*), является задачей с трудоемкостью не менее заданной *Th*.

2. Объект *М* должен быть недоступен для изменения.

3. Длина объекта *М* должна обеспечивать условную вероятность *Р(Н(F 1) = Н(F 2)/ F 1*не тождествен *F* 2) не более заданной Р*h*.

Поясним смысл этих условий. Пусть программа злоумышленника изменила объект *F* (статическое искажение). Тогда, вообще говоря, хэш-значение *М* для данного объекта изменится. Если субъекту злоумышленника доступен для изменения объект *М* (существует соответствующий поток), то он может по известному алгоритму *Н* вычислить новое хэш-значение для измененного объекта и заместить им исходное.

Пусть хэш-значение недоступно, тогда можно попытаться так построить измененный объект, чтобы хэш-значение его не изменилось; принципиальная возможность этого имеется, поскольку отображение, задаваемое алгоритмом хэширования *Н*, не биективно (неоднозначно).

Таким образом, при условии недоступности хэш-значения для изменения и доступности для изменения объекта-источника трудоемкость нарушения ИПС с КЦ объектов-источников (т.е. возможность породить субъект из объекта-источника, не тождественного исходному объекту) совпадает с *Th*. При однократной попытке инициировать субъект из случайно равновероятно выбранного объекта-источника вероятность нарушения ИПС (успешное порождение субъекта) не превосходит Рь. Итак, «качество» ИПС определяется свойствами хэш-функции *Н*, а именно: величинами *Тh* и *Рh*.

Обобщим приводимые выше рассуждения в методе «безопасной загрузки», или ступенчатого контроля. Он заключается в постепенном установлении неизменности компонент программно-аппаратной среды: сначала проверяется неизменность программ ПЗУ, при положительном исходе через проверенные на целостность программы ПЗУ считывается загрузочный сектор и драйверы операционной системы (по секторам) и их неизменность также проверяется, кроме того, проверяется целостность объекта, определяющего последовательность активизации компонент; через функции чтения проверенной ОС инициируется процесс контроля порождения процессов (реализация МБС); инициирование процесса контроля доступа к объектам завершает проектирование гарантировано защищенной КС.

Рассматривая вопросы программно-технической реализации ИПС, необходимо заметить, что мощность множества субъектов в некотором сегменте КС (выделенном по признаку принадлежности одной ЭВМ) с момента включения питания до момента запуска процессов пользователя увеличивается. Первоначально активизируются субъекты аппаратно-программного уровня (программы ПЗУ), затем указанные субъекты порождают из объектов-источников данного уровня (это, как правило, сектора внешних носителей информации) субъектов уровня операционной среды.

Субъекты уровня операционной среды, как уже отмечалось, также делятся на два подуровня: нижний уровень — субъекты — первичные загрузчики ОС (работающие с информацией уровня секторов) и верхний уровень — субъекты-драйверы (порождаемые субъектами — первичными загрузчиками из объектов-секторов), работающие с объектами уровня «файл» (последовательности секторов). На этапе перехода от субъектов загрузчиков к субъектам-драйверам происходит переход и к другой декомпозиции КС на объекты (от секторов к файлам). Указанная иерархия действует в любой известной на сегодняшний день КС и естественным образом предопределяет архитектуру, в рамках которой формируется и функционирует ИПС.

Например, аппаратная архитектура ПЭВМ типа IВМ РС задает следующие этапы активизации различных субъектов КС. При включении питания ПЭВМ происходит тестирование ОП, инициализация таблицы векторов прерываний и поиск расширений ВIОS. При их наличии управление передается на них. После отработки расширений ВIOS в память считывается первый сектор дискеты или винчестера и управление передается на него (образуется код загрузчика), затем код загрузчика считывает драйверы операционной системы, далее интерпретируются файлы конфигурации, подгружается командный интерпретатор и выполняется файл автозапуска.

При реализации ИПС на нее должна быть возложена функция контроля запусков программ и контроля целостности.

При описании методологии проектирования ИПС упоминалась проблема контроля реальных данных. Эта проблема состоит в том, что контролируемая на целостность информация может представляться по-разному на разных уровнях.

Внедренный в систему субъект может влиять на процесс чтения-записи данных на уровне файлов (или на уровне секторов) и предъявлять системе контроля некоторые другие вместо реально существующих данных. Этот механизм неоднократно реализовался в STELS-вирусах. Однако верно утверждение.

Если субъект, обслуживающий процесс чтения данных (т.е.указанный субъект инициируется запрашивающим данные субъектом и участвует в потоке), содержал только функции тождественного отображения данных на ассоциированные объекты данные любого субъекта, инициирующего поток чтения, и объекта-источника для этого субъекта зафиксирована, то при его последующей неизменности чтение с использованием порожденного субъекта будет чтением реальных данных.

**Лекция 20**

**Контроль доступа. Разграничения доступа**. **Иерархический доступ к файлам**. **Фиксация доступа к файлам. Способы фиксации фактов доступа**. **Доступ к данным со стороны процесса.** **Надежность систем ограничения доступа.**

**Модели управления доступом**

Успех в достижении высокой степени безопасности АС зависит от тщательности разработки и реализации управления имеющимися в системе механизмами безопасности. Как показывает практика, наилучшие результаты в создании безопасных систем достигаются в том случае, когда разработчики системы учитывают требования безопасности уже на этапе формулирования целей разработки и самых общих принципов построения системы. При этом разработчики должны четко понимать суть требований безопасности.

В таком случае, разрабатываемая система может быть небезопасной в силу одной из двух причин :

1. есть ошибки в реализации механизмов защиты или механизмов управления ими;
2. ошибочно, недостаточно полно, или неверно понято само определение того, что значит в отношении системы выражение "быть безопасной".

Для устранения первой причины необходимо применение современных технологий создания программного обеспечения в сочетании с принципами разработки, специфичными для выполнений требований безопасности.

Для устранения второй причины необходимо как можно точнее сформулировать понятие "быть безопасной" в отношении разрабатываемой системы.

Известно, что при разработке современных автоматизированных систем используется один из двух методов:

1. Нисходящий метод (метод "сверху-вниз"): сначала составляется общее описание системы; выделяются компоненты системы; поэтапно увеличивается степень детализации компонентов системы (выделение компонентов в компонентах и т.д.) - до момента окончания разработки.

2. Восходящий метод (метод "снизу-вверх"): сначала формулируются задачи системы; затем разрабатывается некоторый набор элементарных функций; на базе элементарных функций разрабатываются более крупные компоненты системы - и так поэтапно разработка ведется до момента объединения отдельных компонентов в единую систему.

Наибольшее распространение получил компромиссный вариант, при котором разработка системы в целом ведется нисходящим методом, а разработка отдельных компонентов системы (в основном элементарных) - восходящим.

Для нас больший интерес представляет нисходящий метод создания систем, так как этот метод позволяет задавать требования безопасности ко всей системе в целом и затем их детализировать применительно к каждой подсистеме.

Нисходящий метод разработки системы обеспечения безопасности может быть неформальным или формальным (Рис.9.2.).

Неформальная разработка Формальная разработка

Требования безопасности

Требования безопасности

Абстрактная модель

(демонстрация)

(доказательство)

Функциональная

спецификация

Формальная

cпецификация

(тестирование) (тестирование)

Требования безопасности

Реализация

Рис.9.2.

Метод неформальной разработки применяется при создании относительно простых систем с небольшим числом компонентов и очевидными алгоритмами их взаимодействия.

По мере увеличения сложности системы взаимосвязи ее компонентов становятся все менее очевидными; становится сложно описать эти взаимосвязи с достаточной степенью точности некоторым неформальным образом (например, на естественном языке). При разработке систем обеспечения безопасности точность в описании компонентов и их взаимосвязей является едва ли не решающим условием достижения успеха, поэтому для обеспечения надлежащей степени точности применяется строгий аппарат формальной математики, что и составляет суть формального метода разработки.

Основную роль в методе формальной разработки системы играет так называемая ***модель управления доступом***. В англоязычной литературе для обозначения сходного понятия используются термины "security model" (модель безопасности) и "security policy model" (модель политики безопасности).

Эта модель определяет правила управления доступом к информации, потоки информации, разрешенные в системе таким образом, чтобы система всегда была безопасной.

Целью модели управления доступом является выражение сути требований по безопасности к данной системе. Для этого модель должна обладать несколькими свойствами:

1. быть адекватной моделируемой системе и не избыточной;
2. быть простой и абстрактной, и поэтому несложной для понимания.

Модель позволяет провести анализ свойств системы, но не накладывает ограничений на реализацию тех или иных механизмов защиты. Так как модель является формальной, возможно осуществить доказательство различных свойств безопасности всей системы.

Моделирование требует значительных усилий и дает хорошие результаты только при наличии времени и ресурсов. Если система уже создана и имеется возможность сделать лишь отдельные изменения в отдельных местах существующей системы ("залатать дыры"), в любом случае маловероятно значительное улучшение состояния безопасности системы и моделирование поэтому будет непродуктивным занятием.

На сегодняшний день создан ряд типовых моделей управления доступом, которые можно использовать при разработке системы.

## Системы разграничения доступа

Основную роль в обеспечении внутренней безопасности компьютерных систем выполняют системы управления доступом (разграничения доступа) субъектов к объектам доступа, реализующие концепцию единого диспетчера доступа (в английском варианте "reference monitor"- дословно, монитор ссылок).

### Диспетчер доступа

Сущность концепции диспетчера доступа состоит в том, что некоторый абстрактный механизм является посредником при всех обращениях субъектов к объектам (Рис.9.3).

База данных

защиты

Объект

доступа

Субъект

доступа

Диспетчер

доступа

Системный

журнал

Рис. 9.3.

Диспетчер доступа должен выполнять следующие функции:

1. проверять права доступа каждого субъекта к любому объекту на основании информации, содержащейся в базе данных защиты (правил разграничения доступа);
2. при необходимости регистрировать факт доступа и его параметры в системном журнале.

Основными требованиями к реализации диспетчера доступа являются:

1. требование полноты контролируемых операций, согласно которому проверке должны подвергаться все операции всех субъектов над всеми объектами системы. Обход диспетчера предполагается невозможным;
2. требование изолированности, то есть защищенности диспетчера от возможных изменений субъектами доступа с целью влияния на процесс его функционирования;
3. требование формальной проверки правильности функционирования;
4. минимизация используемых диспетчером ресурсов.

В самом общем виде работа средств управления доступом субъектов к объектам основана на проверке сведений, хранимых в базе данных защиты.

Под ***базой данных защиты*** (security database) понимают базу данных, хранящую информацию о правах доступа субъектов системы к объектам и другим субъектам.

Для внесения изменений в базу данных защиты система разграничения доступа должна включать средства для привилегированного пользователя (администратора безопасности) по ведению этой базы. Такие средства управления доступом должны обеспечивать возможность выполнения следующих операций:

1. добавления и удаления объектов и субъектов;
2. просмотра и изменения соответствующих прав доступа субъектов к объектам.

Форма представления базы данных защиты может быть различной.

Основу базы данных защиты в общем случае составляет матрица доступа или ее представления. Каждый элемент этой матрицы представляет собой кортеж, определяющий права доступа (для всех возможных видов доступа) каждого субъекта к каждому объекту или другому субъекту.

Сложность управления доступом (ведения матрицы доступа) в реальных системах связана не только с большой размерностью матрицы (большим числом субъектов и объектов) и высоким динамизмом ее корректировки, но и с необходимостью постоянного отслеживания при таких корректировках большого числа зависимостей между значениями определенных кортежей. Наличие таких зависимостей связано с объективно существующими в предметной области ограничениями и правилами наследования полномочий в иерархии объектов и субъектов. Например, пользователь должен наследовать полномочия группы пользователей, в которую он входит; права доступа некоторого пользователя к каталогам и файлам не должны превышать соответствующие его права по доступу к диску, на котором они размещены и т.п.).

При полномочном управлении доступом (категорирование объектов и субъектов и введение ограничений по доступу установленных категорий субъектов к объектам различных категорий) на матрицу доступа накладываются дополнительные зависимости между значениями прав доступа субъектов.

Существующие ограничения и зависимости между полномочиями существенно усложняют процедуры ведения матриц доступа. Это привело к возникновению большого числа способов неявного задания матрицы (списки доступа, перечисление полномочий, атрибутная схема и т.п.).

Основные критерии оценки эффективности различных способов неявного задания следующие:

1. затраты памяти на хранение образа матрицы доступа;
2. время на выборку (вычисление) значений полномочий (элементов кортежей);
3. удобство ведения матрицы при наличии ограничений и зависимостей между значениями ее кортежей (простота и наглядность, количество требуемых операций при добавлении/удалении субъекта или объекта, назначении/модификации полномочий и т.п.).

Рассмотрим основные способы неявного задания матрицы доступа.

### 

### Списки управления доступом к объекту

В данной схеме полномочия доступа к объекту представляются в виде списков (цепочек) кортежей для всех субъектов, имеющих доступ к данному объекту. Это равносильно представлению матрицы по столбцам с исключением кортежей, имеющих все нулевые значения.

Такое представление матрицы доступа получило название "списка управления доступом" (access control list). Этот вид задания матрицы реализован в сетевой ОС Novell NetWare.

Достоинства:

1. экономия памяти, так как матрица доступа обычно сильно разрежена;
2. удобство получения сведений о субъектах, имеющих какой либо вид доступа к заданному объекту;

Недостатки:

1. неудобство отслеживания ограничений и зависимостей по наследованию полномочий субъектов;
2. неудобство получения сведений об объектах, к которым имеет какой либо вид доступа данный субъект;
3. так как списки управления доступом связаны с объектом, то при удалении субъекта возможно возникновение ситуации, при которой объект может быть доступен несуществующему субъекту.

### Списки полномочий субъектов

В данной модели полномочия доступа субъекта представляются в виде списков (цепочек) кортежей для всех объектов, к которым он имеет доступ (любого вида). Это равносильно представлению матрицы по строкам с исключением кортежей, имеющих нулевые значения.

Такое представление матрицы доступа называется "профилем" (profile) субъекта.

В системах с большим количеством объектов профили могут иметь большие размеры и, вследствие этого, ими трудно управлять; изменение профилей нескольких субъектов может потребовать большого количества операций и привести к трудностям в работе системы. Поэтому профили обычно используются лишь администраторами безопасности для контроля работы субъектов, и даже такое их применение весьма ограничено.

Достоинства:

1. экономия памяти, так как матрица доступа обычно сильно разрежена;
2. удобство получения сведений об объектах, к которым имеет какой либо вид доступа данный субъект;

Недостатки:

1. неудобство отслеживания ограничений и зависимостей по наследованию полномочий доступа к объектам;
2. неудобство получения сведений о субъектах, имеющих какой либо вид доступа к заданному объекту;
3. так как списки управления доступом связаны с субъектом, то при удалении объекта возможно возникновение ситуации, при которой субъект может иметь права на доступ к несуществующему объекту.

### 

### Атрибутные схемы

Так называемые атрибутные способы задания матрицы доступа основаны на присвоении субъектам и/или объектам определенных меток, содержащих значения атрибутов.

Наиболее известным примером неявного задания матрицы доступа является реализация атрибутной схемы в операционной системе UNIX.

Основными достоинствами этих схем являются:

1. экономия памяти, так как элементы матрицы не хранятся, а динамически вычисляются при попытке доступа для конкретной пары субъект-объект на основе их меток или атрибутов;
2. удобство корректировки базы данных защиты, то есть модификации меток и атрибутов;
3. удобство отслеживания ограничений и зависимостей по наследованию полномочий субъектов, так как они в явном виде не хранятся, а формируются динамически;
4. отсутствие потенциальной противоречивости.

Недостатки:

1. затраты времени на динамическое вычисление значений элементов матрицы при каждом обращении любого субъекта к любому объекту;
2. при атрибутной схеме затруднено задание прав доступа конкретного субъекта к конкретному объекту.