Основные понятия

Проблема построения защищенных вычислительных систем связана с синтезом вычислительных систем, одним из требований к которым является защита от угроз безопасности.

Таким образом, для того чтобы начать обсуждение основных теоретических принципов синтеза и анализа защищенных систем, нам необходимо разобраться в вопросе: от чего нам следует защищать компьютерную систему? Для этого необходимо рассмотреть основные типы угроз, в том числе математические модели защищенных систем и основные понятия, связанные с ними, после чего выделить общие принципы, методы и концепции, применяемые при синтезе вычислительных систем.

Основные угрозы ВС

Угроза вычислительной системе - это возможное происшествие, преднамеренное или нет, которое может оказать нежелательное воздействие на активы и ресурсы, связанные с вычислительной системой.

Типы угроз

Для лучшего понимания возможных угроз и соотнесения их со средствами компьютерной защиты исследователи предложили выделить три различных типа угроз : угрозы, относящиеся к *раскрытию, целостности* или *отказу служб* вычислительной системы.

1. *Угроза раскрытия* заключается в том, что информация становится известной пользователю, который не авторизован для этого. В терминах компьютерной безопасности угроза раскрытия имеет место всякий раз, когда получен несанкционированный доступ к некоторой секретной информации, хранящейся в вычислительной системе или передаваемой от одной системы к другой. Иногда в связи с угрозой раскрытия используется термин "утечка информации".

Такой доступ может вызвать незначительные затруднения, как в случае, когда личная информация, например файлы с письмами, обнаружена коллегами. Однако, если доступ влечет за собой более серьезную утечку информации, как, например, получение противником руководства по созданию бомб или стратегии боевого управления военной организацией, то последствия могут быть значительно более серьезными.

За последние два десятилетия в сферах компьютерной безопасности большое внимание уделялось угрозе раскрытия. Фактически, подавляющее большинство исследований и

разработок в области компьютерной безопасности было сосредоточено на угрозах раскрытия. Одной из причин этого являлось значение, которое правительства придавали противостоянию этой угрозе.

2. *Угроза целостности* включает в себя любое несанкционированное изменение информации, хранящейся в вычислительной системе или передаваемой из одной системы в другую. Когда нарушитель преднамеренно изменяет информацию, мы говорим, что целостность этой информации нарушена. Целостность также будет нарушена, если к несанкционированному изменению приводит случайная ошибка. Санкционированными изменениями являются те, которые сделаны определенными лицами с обоснованной целью (таким изменением является периодическая запланированная коррекция некоторой базы данных).

Как и угроза раскрытия, угроза целостности может также иметь незначительные последствия при изменении некритичной информации. Например, если атака на целостность применена к системе, которая сохраняет резервные копии и содержит не очень важную информацию, то последствия атаки не могут быть серьезными. Однако, если хранимая информация чрезвычайно важна (например история болезни пациентов) и если не предприняты никакие меры по предотвращению или ослаблению несанкционированных изменений, то последствия атаки на целостность могут быть серьезными.

3. *Угроза отказа служб* (отказа в обслуживании) возникает всякий раз, когда в результате преднамеренных действий, предпринятых другим пользователем, умышленно блокируется доступ к некоторому ресурсу вычислительной системы. Например, если один пользователь запрашивает доступ к службе, а другой предпринимает что-либо для недопущения этого доступа, мы говорим, что имеет место отказ службы. Блокирование может быть постоянным, так чтобы запрашиваемый ресурс никогда не был получен, или оно может вызвать только задержку запрашиваемого ресурса, достаточно долгую для того, чтобы он стал бесполезным. В таких случаях говорят, что ресурс исчерпан.

Наиболее частые примеры атак, связанных с отказом служб, включают в себя ресурсы общего пользования (принтеры или процессоры) так, чтобы другие пользователи не могли с ними работать. Пока эти ресурсы не являются частью некоторого критического приложения, этот тип атаки может не считаться слишком опасным. Однако, если доступ к ресурсу должен быть получен своевременно, как, например, при вызове системы тревоги или при инициализации развертывания какого-то вооружения), то атак отказа служб надо избежать.

Сферы безопасности не очень подробно изучали отказ служб. Большинство исследователей в принципе договорились о нескольких определениях и подходах, но для

большей части отказ служб и компромиссы для предотвращения этой угрозы в большинстве своем не исследованы. Одной из причин этого могли послужить размытые и нечеткие границы между угрозой отказа службы и вопросами, касающимися объединения систем реального времени. Другим объяснением могло быть то, что поскольку вопросы отказа служб основаны на временных понятиях, они значительно труднее для формализации, чем раскрытие и целостность.

Политики безопасности

Для того чтобы определить, от каких именно угроз и каким образом защищает информацию вычислительная система, необходимо сформулировать ее политику безопасности. *Политика безопасности* подразумевает множество условий, при которых пользователи системы могут получить доступ к информации и ресурсам. С одной стороны, политика безопасности предписывает пользователям, как правильно эксплуатировать систему, с другой - политика безопасности определяет множество механизмов безопасности, которые должны существовать в конкретной реализации вычислительной системы. Политика безопасности компьютерной системы может быть выражена *формальным и неформальным* образом.

Формальные и неформальные политики безопасности

Для *неформальных* политик безопасности широкое распространение получило описание правил доступа субъектов к объектам в виде таблиц, наглядно представляющих правила доступа (данные понятия интуитивно понятны, а их формальное определение будет введено ниже). Обычно такие таблицы подразумевают, что субъекты, объекты и типы доступа для данной системы определены. Это позволяет составить таблицу в виде одной колонки для различных определенных типов доступа и другой колонки для соответствующего отношения, которое должно соблюдаться между субъектом и объектом для данного типа доступа.

Чтобы проиллюстрировать неформальное табличное представление политики безопасности, рассмотрим политику безопасности простой компьютерной системы. Мы будем использовать таблицу, обозначающую тип доступа и связанное с ним отношение, которое должно соблюдаться между производящим запрос субъектом и объектом, которому этот запрос адресован. Это показано в таблице 1.1.

Таблица 1.1.

Выражение неформальной политики безопасности

Тип доступа	Отношение (субъект, объект)

Чтение	Больше
Выполнение	Больше
Запись	Равно

Глядя на таблицу, можно сделать заключение, что уровни безопасности субъектов должны быть выше уровней безопасности объектов для того, чтобы субъектам системы были разрешены операции чтения и выполнения. Подобным образом можно сделать заключение о том, что уровни субъекта и объекта должны быть одинаковыми для получения разрешения на выполнение операции записи.

Преимуществом такого способа представления политики безопасности является то, что она гораздо *легче для понимания* пользователями, чем формальное описание, так как для ее понимания не требуется специальных математических знаний. Это снижает вероятность атак на вычислительную систему по причине ее некорректной эксплуатации вследствие непонимания принципов разработки системы. Основным *недостатком неформального описания политики безопасности системы* является то, что при такой форме представления правил доступа в системе гораздо *легче допустить логические ошибки при проектировании системы и ее эксплуатации*, приводящие к нарушению безопасности системы. Особенно это справедливо для политик безопасности нетривиальных систем, подобных многопользовательским операционным системам.

В результате разработчики (а в дальнейшем и пользователи) безопасных компьютерных систем начали использовать *формальные* средства для описания политик безопасности. *Преимуществом* формального описания является *отсутствие противоречий в политике безопасности и возможность теоретического доказательства безопасности системы* при соблюдении всех условий политики безопасности. Требование формального описания систем характерно для систем, область применения которых критична. В частности, можно отметить тот факт, что для защищенных вычислительных систем высокой степени надежности необходимы формальное представление и формальный анализ системы.

Формальные методы анализа систем

Для лучшего понимания принципов, используемых при создании формальных политик безопасности, рассмотрим основные математические методы, применяемые при формальном анализе и формальном описании политик безопасности систем. При анализе функционирования систем (при этом системы являются необязательно вычислительными), область применения которых является критичной (а к данному классу обычно относятся защищенные

вычислительные системы), желательно рассмотреть все возможные поведения системы (то есть ее реакции на все возможные входные данные).

Хотя количество “всех возможных” реакций системы слишком велико для исследования, существует два метода, позволяющих уменьшить количество реакций, которые необходимо рассмотреть.

Один из методов заключается в доказательстве того, что система всегда “работает корректно”, тогда как другой заключается в демонстрации того, что система никогда не выполняет неверных действий.

При использовании *первого* метода используется комбинация анализа и эмпирического тестирования для определения таких реакций системы, которые могут привести к серьезным сбоям - например функционирование системы при граничных условиях или при условиях, не оговоренных в качестве возможных для компонентов системы. В качестве примера можно привести требование описания поведения системы в ответ на входное воздействие “выключение питания”.

Основной идеей *второго* метода является гипотеза о том, что система делает что-то некорректное[1], поэтому ведется анализ реакций системы с выявлением состояний, в которых возможно проявление данной некорректности. Доказательство корректности работы системы сводится к демонстрации того, что данные состояния недостижимы, то есть к доказательству от противного.

Свойство, характерное как для одной, так и для другой техники анализа безопасности системы, выражается следующим образом: данные техники анализа обеспечивают пути группирования “схожих” реакций системы так, что для рассмотрения всех возможных реакций системы необходимо рассмотреть лишь небольшое количество входных воздействий.

К сожалению, данные методы анализа безопасности систем пригодны в основном для систем, основывающихся на механических, электрических и других компонентах, то есть компонентах, основанных на физических принципах действия. В системах, основанных на физических принципах, отношения между входными и выходными данными являются «непрерывными», то есть незначительные изменения во входных данных влекут незначительные изменения в выходных данных. Это позволяет проанализировать

---

[1] Необходимо отметить, что нежелательные реакции вычислительной системы на некоторые входные данные называются *уязвимостями.* В терминах компьютерной безопасности *уязвимость* вычислительной системы - это некоторая неудачная характеристика, которая делает возможным возникновение потенциальной угрозы. В связи с защищенными вычислительными системами при анализе их безопасности можно говорить о поиске *уязвимостей* системы.

(протестировать) поведение системы, основанной на физических законах конечным количеством тестов, так как непрерывный характер правил функционирования системы позволяет заключить, что отклик системы на непротестированное значение входной величины будет схожим с соответствующими протестированными случаями.

Таким образом, можно сделать вывод о том, что метод непосредственного тестирования хотя и необходимо применять к системам критического назначения, но он способен детектировать только примитивные ошибки в программном обеспечении. Вследствие сложности современных программных систем и вытекающего из этого многообразия реакций системы на входной поток данных, описанная выше техника не достаточна для анализа сложной программной системы.

Сложность программной системы определяется большим количеством дискретных решений, принимаемых системой при исполнении программного обеспечения. Таким образом, при определении взаимоотношений между входом и выходом системы (входным и выходным потоками данных), в случае анализа программной системы, реакцию системы на входное воздействие нельзя рассматривать как непрерывную функцию, так как она является дискретной: небольшие изменения во входных данных системы могут радикально изменить поведение всей системы в целом. Это является главным отличием современных программных систем от систем, основанных на физических процессах.

Отклонение от непрерывности ведет к катастрофическому росту количества возможных реакций системы на изменения во входных данных. Таким образом, *в случае с программными системами метод непосредственного тестирования с последующими выводами о свойствах системы* (то есть описание всех возможных реакций системы на входной поток данных) *не дает должной уверенности вследствие дискреционного характера отношений между входными данными и выходной реакцией системы* (нельзя сказать, что небольшие изменения во входных данных системы приведут к сходной ее реакции). При непосредственном тестировании систем, содержащих программное обеспечение, можно говорить только о вероятностных, но не определенных результатах тестирования.

Исходя из описанных выше недостатков использования непосредственного тестирования при анализе сложных программных систем, выясняется необходимость применения другой техники анализа, то есть формальных методов.

В области программного обеспечения, в отличие от анализа систем, использующих физические принципы, вместо дифференциальных уравнений используются понятия дискретной математики, такие как “множества”, “графы”, “частичный порядок”, “машина конечных состояний” и т.д. “Вычисления” при описании данных систем базируются на методах

формальной логики, а не на численном анализе. Это происходит потому, что результаты, интересующие при анализе системы, являются логическими свойствами. Математическая логика обеспечивает методы доказательства свойств больших или бесконечных множеств связанных сущностей на конечный манер на основе методов, сходных с методом математической индукции.

При формальном анализе системы, для того чтобы сделать выводы о ее логических свойствах, исходя из определенных структур дискретной математики, необходимо начать с аксиом, описывающих данные структуры, и правил манипуляции символами, соответствующими данным структурам. Процесс манипулирования символами называется формальной дедукцией, так как корректность процесса зависит от формы символических выражений, но не от их семантического смысла.

Практическое значение применения формальных методов при анализе систем заключается в возможности анализа всех реакций сложной программной системы. Рассмотрим причины, по которым возможно сделать вышеприведенное заключение.

Прежде всего необходимо отметить, что в случае *формального анализа* мы имеем дело не просто с реакцией системы на некоторые группы входных данных, а с *внутренней реализацией системы*. Таким образом, возникает возможность “декомпозиции” возможных реакций системы на реакции ее компонентов (с помощью анализа формальных спецификаций компонентов) с последующей их композицией, что дает возможность описания всех реакций системы.

С другой стороны, *формальные методы позволяют провести логическое вычисление причин корректности систем с программным обеспечением*. Данное утверждение можно пояснить следующим образом. Программное обеспечение пишется для того, чтобы система выполняла некоторую полезную работу, то есть достигала какой-то цели (желаемая реакция системы). Если можно записать данную цель, то возможно сконструировать аргументы, поясняющие уверенность в том, что система с программным обеспечением функционирует корректно. Формальные методы позволяют уменьшить количество аргументов, необходимых для логического вычисления утверждения о корректности работы системы.

Для автоматизации процесса доказательства свойств системы используются “доказатели теорем” - программное обеспечение, проводящее формальную дедукцию на основе комбинации эвристик и непосредственного поиска. Другим видом программного обеспечения, используемого при формальном анализе систем, являются “контроллеры доказательств”, программы, предоставляющие пользователю проводить последовательность шагов в процессе логических выводов о свойствах системы, но проверяющие корректность каждого шага.

Естественно, самыми эффективными средствами формального анализа являются средства, сочетающие в себе свойства двух приведенных выше видов программного обеспечения.

Описанные выше принципы применения формальных методов для анализа систем, содержащих программное обеспечение, имеют *недостаток*, присущий всем методам моделирования: формальные методы *имеют дело* не с самой системой, а *с* ее *моделью*. Это означает, что модель может не отражать реальность или отражать ее некорректно.

Данная проблема может возникнуть вследствие использования модели, учитывающей не все факторы, оказывающие влияние на реальную систему. С другой стороны, излишняя подробность описания системы ведет к резкому росту временных затрат на проведение формального анализа, что может привести к неэффективности применения данного метода. К модели безопасности должны предъявляться требования, общие для всех моделей:

— адекватность;

— способность к предсказанию;

— общность.

Таким образом, возникает *задача корректного выбора уровня абстракции в описании модели безопасности*. Под уровнем абстракции понимается множество требований к реальной системе, которые должны найти свое отражение в модели, для корректного отображения моделируемой системы.

Модели безопасности. Основные понятия

Для того чтобы перейти к рассмотрению модели безопасности системы, необходимо представить механизм, свойства которого должны уточняться в результате моделирования. Кратко рассмотрим концепцию монитора пересылок, отражающую свойства механизмов безопасности. В данной концепции сеанс работы пользователя в компьютерной системе характеризуется инициированием процесса, который работает в интересах пользователя и выполняет последовательность операций доступа к объектам системы. Очевидно, что должна существовать некая процедура принятия решений о том, какой из запрашиваемых доступов разрешить, а какой нет. По другому это можно представить как фильтр, через который должны пройти все запросы на доступ, сделанные субъектами. Схема такого фильтра получила название *монитора пересылок*.

Основной характеристикой монитора пересылок является то, что он или разрешает запрос на доступ, или запрещает, возможно, уведомляя об этом субъекта. Таким образом, монитор пересылок может быть описан в терминах функции с запросами на обслуживание, разрешениями доступа, другими компонентами состояния системы на входе (т. е. элементами

области определения) и разрешениями или запретами на обслуживание на выходе (т. е. элементами области значения).

Таким образом, монитор пересылок должен удовлетворять трем требованиям:

— ни один запрос на доступ субъекта к объекту не должен проходить мимо монитора пересылок;

— целостность монитора пересылок должна строго контролироваться;

— представление монитора пересылок должно быть достаточно простым для доказательства корректности его работы.

Однако совершенно необязательно, чтобы безопасная система включала в свою архитектуру отдельный модуль (возможно, называемый модулем монитора пересылок), который бы обрабатывал запросы. Способ выполнения обработки запросов определяется проектировщиками и разработчиками системы.

Если в начале проектирования безопасных систем данный монитор реализовывался в виде отдельного модуля, встроенного в операционную систему, и функции, то в настоящее время существует тенденция к реализации данной концепции в форме совместно работающего программного и аппаратного обеспечения, называемого *ТСВ системы*.

Таким образом, можно сделать следующий вывод: на основании компактного ядра безопасности ТСВ системы мы делаем заключение о свойствах безопасности системы в целом. Как было отмечено в параграфе 1.2, говоря о некотором свойстве системы, можно строить доказательство двумя способами: либо показать, что система ведет себя корректно всегда, либо показать, что система никогда не выполняет некорректных действий.

Основной причиной выделения программного обеспечения ТСВ системы является необходимость независимости свойства безопасности системы от программного обеспечения системы, не входящего в состав ТСВ. На уровне пользователя системы функции безопасности системы должны быть выражены в виде функций безопасности системы, предоставляемых ТСВ системы. Данная функциональная зависимость исчезает, если программное обеспечение ТСВ системы совпадает с программным обеспечением системы в целом. Но данное условие никогда не выполняется на практике, и программное обеспечение ТСВ системы обычно много компактнее и проще, чем программное обеспечение системы в целом.

Следствием данной неполноты является невозможность сделать заключение о том, как используется ТСВ системы пользовательским программным обеспечением. Таким образом, рассуждения о безопасности системы на основе выделения ядра безопасности в виде ТСВ системы нельзя строить на предположении о том, что система функционирует корректно. Но с другой стороны, использование ТСВ системы позволяет говорить о том, что некорректное

поведение системы невозможно. Иллюстрацию данного свойства можно найти в главе 2. Большинство теорем о безопасности системы доказываются по принципу от противного, что соответствует негативной природе свойства безопасности.

Выше было отмечено, что на уровне рассмотрения системы в целом свойства безопасности системы являются функцией свойств безопасности ядра безопасности системы. В логике существует понятие предикатов первого и второго порядка. Предикаты второго порядка расширяют предикаты первого порядка в том, что они позволяют квантификацию (применение логических операторов *общности* (∀) и *существования* (∃)) не только над переменными, но и над функциями.

Многие свойства систем могут быть выражены в терминах логики первого порядка. Например, свойство стека системы может быть описано следующим предикатом первого порядка: ∀ x top (push (s, x)) = x. Логика первого порядка подходит для описания позитивных свойств системы, то есть того, что система делает.

Вследствие негативной природы для описания свойств безопасности системы необходимо применение логики второго порядка. Ядро безопасности гарантирует безопасность системы вне зависимости от того, какие его функции и в какой последовательности задействованы. При этом свойства безопасности системы в целом могут быть описаны следующим выражением:

$$\forall \alpha \in op' : P(\alpha), \tag{1.1}$$

где op'- множество всех возможных последовательностей вызовов функций, предоставляемых ядром безопасности системы;

P() - предикат, определяющий выходную реакцию в системе в зависимости от входного воздействия.

Выражение (1.1) является предикатом второго порядка и определяет следующее свойство: любая операция, выполняемая пользовательским программным обеспечением, преобразуется в последовательность вызовов функций ядра безопасности системы (функций в множестве op')[2] и при условии защиты программного обеспечения ядра системы от модификации[3], свойство P() является свойством системы в целом (данное свойство не зависит от поведения программного обеспечения, не принадлежащего ядру системы). Доказательство безопасности системы сводится к демонстрации инвариантности модели системы по отношению к некоторому свойству безопасности, определяемому предикатом P. Таким

[2] Свойство "полноты".

[3] Свойство "изоляции"

образом, для синтеза защищенной вычислительной системы необходимо выполнение следующих условий:

— свойства безопасности системы должны быть реализованы с помощью ядра безопасности системы;

— данные свойства должны быть выражены предикатом второго порядка (1.1).

В анализируемых в главе 2 моделях безопасности предикат P конкретизируется. Вместо исследования выражений, описываемых логикой второго порядка, изучаются модели состояний системы. *Для модели системы необходимо доказать следующую теорему, называемую основной теоремой безопасности, в которой говорится следующее: "если система начинает работу в безопасном состоянии и все переходы системы из состояния в состояние являются безопасными, то система является безопасной"*.

Таким образом, необходимо ввести понятия, пригодные для описания свойств состояния системы. Самыми естественными понятиями, с помощью которых можно описать состояние системы, являются понятия субъекта, объекта и доступа, которые и являются основой описания состояния моделей безопасности защищенных вычислительных систем.

Субъекты, объекты и доступ

Под *сущностью* мы будем понимать любую именованную составляющую компьютерной системы.

Объект будет определяться как *пассивная сущность*, используемая для хранения или получения информации. Примеры объектов: записи, блоки, страницы, сегменты, файлы, директории, биты, байты, слова, терминалы, узлы сети и т.д.

Субъект будет определяться как *активная сущность*, которая может инициировать запросы ресурсов и использовать их для выполнения каких-либо вычислительных заданий. В процессе исполнения субъекты исполняют операции. Под субъектами мы будем обычно понимать пользователя, процесс или устройство.

Хотя основную концепцию идентификации субъектов и объектов в системе описать просто, при практической реализации часто бывает нетривиальной задачей определить: что есть субъект, а что - объект. Например, в ОС процессы, конечно, являются субъектами, в то время как файлы и связанные с ними директории - объектами. Однако, когда субъекты получают коммуникационные сигналы от других субъектов, встает вопрос, рассматривать ли их как субъекты или же как объекты. Данная трудность может быть преодолена путем усложнения понятия субъекта в модели.

При исполнении субъектами операций происходит взаимодействие субъектов и объектов. Такое взаимодействие называется доступом. *Доступ* - это *взаимодействие между субъектом и объектом, в результате которого происходит перенос информации между ними*. Существуют две фундаментальные операции, переносящие информацию между субъектами и объектами. Под *операцией чтения* понимается операция, результатом которой является перенос информации от объекта к субъекту. Под *операцией записи* понимается операция, результатом которой является перенос информации от субъекта к объекту. Данные операции являются минимально необходимым базисом для описания широкого круга моделей, описывающих защищенные системы.

Уровни безопасности. Доверие и секретность

В данном параграфе рассмотрим характеристики субъектов и объектов, относящиеся к понятию безопасности. Основной характеристикой безопасности субъекта является *уровень безопасности*. Под *уровнем безопасности* будем понимать иерархический атрибут, который может быть ассоциирован с сущностью компьютерной системы для обозначения степени ее чувствительности в смысле безопасности. Данная степень чувствительности может помечать, например, степень ущерба от нарушения безопасности данной сущности в компьютерной системе.

Когда в системе существуют такие иерархические отношения, то требуется некий механизм, помечающий сущности компьютерной системы так, чтобы их безопасностная чувствительность была известна. Один из путей достижения этого - ассоциировать каждую сущность компьютерной системы с уровнем безопасности. Например, в военном деле набор уровней состоит из неклассифицированной информации, конфиденциальной, секретной и совершенно секретной. Иерархия уровней безопасности в военном деле устанавливается тем фактом, что особо секретная информация рассматривается как вышестоящая над секретной, которая, в свою очередь, стоит выше конфиденциальной, а последняя стоит выше неклассифицированной.

Для представления уровней безопасности введем простые математические структуры и соотношения. Обозначим множество уровней безопасности символом L; иерархическое отношение между различными элементами L описываются символами: "<" , "≤", ">" и "≥". Множество L может рассматриваться как полностью упорядоченное множество (то есть любые два элемента L можно сравнить для определения того, равны они между собой или какой-то из них больше другого).

Доверие определяется как атрибут, задающий чувствительность субъекта к безопасности; *секретность* определяется как атрибут объекта, обозначающий его чувствительность к безопасности. Эти концепции можно представить при помощи проекции субъектов и объектов на уровни безопасности посредством двух простых функций: *clearance* и *classification*.

Областью значений каждой из функций является множество L. Функция *clearance* определена на множестве S, а *classification* определена на множестве O. Эти функции записываются в виде:

clearance: $S \rightarrow L$

classification: $O \rightarrow L$

Характеристика моделей безопасности

В параграфе 1.2.4 были введены понятия субъектов, объектов и доступа. В настоящем параграфе рассмотрим общие принципы построения математических моделей безопасности вычислительных систем.

Обычно модель безопасности состоит из двух компонентов: *системного компонента* и *компонента безопасности*. Данные компоненты находят свое отражение в характеристиках субъектов и объектов системы.

Системный компонент определяет функциональные требования к вычислительной системе в контексте модели безопасности. Отсутствие или некорректное определение компонента безопасности ведет к некорректной цели анализа. Современное состояние науки определяет понятие компонента безопасности несколькими способами. Но в основе всех компонентов безопасности (относятся данные модели к целостности, раскрытию или отказу в обслуживании) лежит идея о том, что доступ (по чтению, записи, исполнению и т.д.) к информации должен получать только пользователь (или процесс, действующий в его интересах), который должен быть авторизован для этого.

С другой стороны, компонент безопасности определяет понятие безопасности, используемое в модели. *Соответствие анализируемой модели реальности определяется уровнем абстрактности системного компонента*. Недостаточная детализация данного компонента ведет к тому, что модель системы безопасности не отражает реальной системы, и как следствие, делать выводы о реальной безопасности системы на основании данной модели не представляется возможным. С другой стороны, чрезмерная сложность системного компонента ведет как к увеличению времени на формальный анализ системы, так и к непереносимости доказательства безопасности модели на более широкий круг систем.

Для того, чтобы перейти к описанию моделей безопасности, необходимо обозначить принципы их построения, то есть построить их классификацию. Предлагаемая авторами классификация моделей безопасности построена по принципу используемых в их описании понятий субъекта и объекта. Данная классификация отражает принципы постепенного семантического уточнения характеристик субъектов и объектов модели. Семантическое уточнение касается как уточнения особенностей описания фундаментальных операций доступа, используемых субъектами системы, так и введения в модель новых типов доступа с соответствующими ограничениями на них. Данное уточнение ведет к постепенному усложнению системного компонента модели.

В таблице продемонстрированы модели как для угроз раскрытия, так и для угроз целостности. Соответствующие модели отличаются компонентом безопасности и в основном совпадают в системном компоненте. Следовательно, и это будет показано далее, доказательство безопасности соответствующих моделей будет во многом аналогично. Разница проявится в инверсии требований безопасности, что вызвано различными компонентами безопасности, принятыми в модели. Классификация моделей безопасности представлена в таблице 1.3. Более подробно соответствующие модели будут рассмотрены в главе 2.

Таблица 1.3

Классификация моделей безопасности

Компонент безопасности	Системный компонент	Типы доступа, используемые в модели	Особенности операций доступа субъектов к объектам	Комментарии	Модель секретности, представитель класса	Модель целостности, представитель класса
Отсутствует	Отсутствует	READ, WRITE	Обеспечивается хорошее разделение субъектов друг от друга		Модели дискреционного доступа	
Множество субъектов и множество объектов упорядочены в соответствии с уровнями безопасности	Отсутствует	READ, WRITE	Ограничения накладываются на простейшие операции READ, WRITE		Модель Белла-Лападула	Модель Биба
Множество субъектов и множество объектов упорядочены в соответствии с уровнями безопасности	В множестве субъектов имеются доверенные субъекты	READ, WRITE	Доверенные субъекты не подчиняются ограничениям на операции READ, WRITE, определенные в модели 2		Модель доверенных субъектов	
Множество субъектов и множество объектов упорядочены в	Субъекты выполняются на нескольких устройствах	READ, WRITE	Операции READ, WRITE могут быть удаленными	Удаленный характер операций READ, WRITE может вызвать противоречия в модели 2	Модели распределенных систем (синхронные	

соответствии с уровнями безопасности	обработки				и асинхронные)	

Продолжение таблицы 1.3

Компонент безопасности	Системный компонент	Типы доступа, используемые в модели	Особенности операций доступа субъектов к объектам	Комментарии	Модель секретности, представитель класса	Модель целостности, представитель класса
Множество субъектов и множество объектов упорядочены в соответствии с уровнями безопасности	Переход системы из состояния в состояние в один момент времени может осуществляться под воздействием более одного субъекта	READ, WRITE	В один момент времени несколько субъектов могут получить доступ к нескольким объектам		Модели распределенных систем (асинхронные)	
Множество субъектов и множество объектов упорядочены в соответствии с уровнями безопасности	Субъекты модели имеют сложную структуру и имеют объектную часть	READ, WRITE, SET ACTIVE SUBJECT	Субъект может получить доступ к другому субъекту как к объекту	Субъекты модели включают в свое описание объектную часть. По данному принципу строятся системы, основанные на передаче сообщений. Мощность объединения множества субъектов и объектов постоянна. Мощность множества субъектов и объектов не постоянна		
Множество субъектов и	Мощность множеств	READ, WRITE,	В описание модели	Мощность объединения множества субъектов и		

множество объектов упорядочены в соответствии с уровнями безопасности	субъектов и объектов модели динамически изменяется	CREATE, DELETE	включены операции CREATE и DELETE	объектов не постоянна		

Окончание таблицы 1.3

Компонент безопасности	Системный компонент	Типы доступа, используемые в модели	Особенности операций доступа субъектов к объектам	Комментарии	Модель секретности, представитель класса	Модель целостности, представитель класса
Множество субъектов и множество объектов упорядочены в соответствии с уровнями безопасности	Субъекты могут выполнять специализированные операции над объектами сложной структуры	READ, WRITE, CREATE, DELETE, операции над объектами специфической структуры	Кроме простейших операций в модели могут появиться операции, направленные на специфичную обработку информации		Модель защищенности сети. Модель MMS	
Накладываются ограничения на ввод и вывод информации субъектов системы		READ, WRITE	Ограничения накладываются на поток информации		Модели невмешательства, Модель невыводимости	

Субъекты и объекты модели имеют вероятностные характеристики		READ, WRITE	Вероятностные характеристики операций доступа	Вероятностные модели		

ЛИТЕРАТУРА

1. Теория и практика обеспечения информационной безопасности/ Под ред. П.Д. Зегжды.- М.: Яхтсмен, 1996. - 304 с.
2. Грушо А.А., Тимонина Е.Е. Теоретические основы защиты информации. - М.: Яхтсмен, 1996. - 192 с.
3. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации. Руководящий документ. - М.: Гостехкомиссия, 1992.
4. Lampson, B. W. 1973. A note on the confinement problem, Comm. ACM 16,10 (October), 613-615.
5. J.McLean High assurance computer systems: a research agenda, 1995.
6. Miller, L. Fredriksen, An empirical study of the reliability of UNIX utilites, 1990.
7. B.Miller, D.Kosky, R.Murthy, C.Lee, A.Natarajan, J.Steidl, V.Maganty, Fuzz revisited: a re-examination of the reliability of UNIX utilites and services, 1995.
8. J.Rushby, Formal methods and the certification of critical systems, 1993.
9. J.Rushby, Formal methods and their role in the certification of critical systems, 1995.
10. J.Rushby, Critical System Properties: survey and taxonomy, Reliability Engineering and System Safety, Vol.43, 1994.
11. J.Rushby Kernels for safety?, Safe and Secure Computing Systems, 1986.
12. Departament of Defense Trusted Computer System Evaluation Criteria. 5200.28-S TD, USA, 1983.

Модели безопасности

На основании материала, изложенного в предыдущей главе, необходимо рассмотреть модели безопасности, лежащие в основе политик безопасности, используемых при защите вычислительных системы от угроз, описанных в параграфе 1.1.

Следует отметить, что в настоящей главе модели рассматриваются в соответствии с типом угроз, от которых защищают информацию вычислительные системы, синтезированные на основании данных моделей. В параграфе 2.1 рассматриваются модели, защищающие от угрозы раскрытия информации. В параграфе 2.2 описаны математические основы моделей контроля целостности информации. И, наконец, в параграфе 2.3 описаны теоретические принципы синтеза механизмов, используемых при синтезе политик безопасности для систем, предотвращающих угрозы отказа служб.

Модели разграничения доступа

Модели разграничения доступа служат для синтеза политик безопасности, направленных на предотвращение угрозы раскрытия, заключающейся в том, что информация становится известной тому, кому не следовало бы ее знать. Данные модели могут быть классифицированы следующим образом:

— модели разграничения доступа, построенные по принципу предоставления прав;

— модели разграничения доступа, построенные на основе принципов теории информации;

— модели разграничения доступа, использующие принципы теории вероятностей.

В дальнейших пунктах мы рассмотрим модели разграничения доступа, построенные на основании перечисленных выше принципов.

Модели разграничения доступа, построенные по принципу предоставления прав

Модели разграничения доступа, построенные по принципу предоставления прав, являются самой естественной основой для построения политик разграничения доступа. Система, политика безопасности которой построена на основании данного принципа, впервые была описана в литературе в середине шестидесятых годов.

Неформально право доступа может быть описано как “билет”, в том смысле, что владение “билетом” разрешает доступ к некоторому объекту, описанному в билете.

Основными типами моделей, построенных на предоставлении прав, являются модели дискреционного (2.1.1.1) и мандатного (2.1.1.2.) доступов. Модели данного типа используются в большинстве реальных систем, синтезированных в настоящее время. Требования, на которых основаны данные модели, лежат в основе требований, сформулированных в различных государственных нормативных документах.

Модели дискреционного доступа

Одна из первых моделей безопасности была модель дискреционного доступа, модель АДЕПТ-50[3]. В модели представлено четыре типа объектов, относящихся к безопасности: пользователи(u), задания(j), терминалы(t) и файлы(f), причем каждый объект описывается четырехмерным кортежем (A, C, F, M), включающим параметры безопасности:

Компетенция A - скаляр - элементы из набора иерархически упорядоченных уровней безопасности, таких как НЕСЕКРЕТНО, КОНФИДЕНЦИАЛЬНО, СЕКРЕТНО, СОВЕРШЕННО СЕКРЕТНО.

Категория C - дискретный набор рубрик. Категории не зависят от уровня безопасности. Пример набора рубрик: ОГРАНИЧЕНО, ТАЙНО, ТОЛЬКО ДЛЯ ПРОСМОТРА, ЯДЕРНЫЙ, ПОЛИТИЧЕСКИЙ.

Полномочия F - группа пользователей, имеющих право на доступ к определенному объекту.

Режим M - набор видов доступа, разрешенных к определенному объекту или осуществляемых объектом. Пример: ЧИТАТЬ ДАННЫЕ, ПРИСОЕДИНЯТЬ ДАННЫЕ, ИСПОЛНИТЬ ПРОГРАММУ.

Если U={u} обозначает набор всех пользователей, известных системе, а F(i) - набор всех пользователей, имеющих право использовать объект i, то для модели формулируются следующие правила:

1. Пользователь u получает доступ к системе $\Leftrightarrow$ $u \in U$.
2. Пользователь u получает доступ к терминалу t $\Leftrightarrow$ $u \in F(t)$ (то есть в том и только в том случае, когда пользователь u имеет право использовать терминал t).
3. Пользователь u получает доступ к файлу j $\Leftrightarrow$ $A(j) \geq A(f)$, $C(j) \supseteq C(f)$, $M(j) \supseteq M(f)$ и $u \in F(f)$, то есть тогда и только тогда, когда выполняются условия:

— привилегии выполняемого задания шире привилегий файла или равны им;

— пользователь является членом F(f).

Задавая параметры безопасности A, C, F, M, можно сформировать матрицу определения параметров безопасности (табл. 2.1).

Четырехмерный кортеж безопасности, полученный на основе прав задания, а не прав пользователя, используется в модели для управления доступом. Данный подход обеспечивает однородный контроль права на доступ над неоднородным множеством программ и данных, файлов, пользователей и терминалов. Например, наивысшим полномочием доступа к файлу для пользователя "СОВ. СЕКРЕТНО", выполняющего задание с "КОНФИДЕНЦИАЛЬНОГО" терминала будет "КОНФИДЕНЦИАЛЬНО".

Таблица 2.1

Матрица определения параметров безопасности модели АДЕПТ-50

Объект	A	C	F	M
Пользователь u	Const	Const	{u}	Const
Терминал t	Const	Const	{u(t,i)}	Const
Задание j	min(A(u),A(t))	$C(u) \cap C(t)$	{u(j,i)}	$M(u) \cap M(t)$
Существ. файл f(i)	Const	Const	{u(f,i)}	Const
Нов .файл f=g(f1,f2)	max(A(f1),A(f2))	$C(f1) \cup C(f2)$	{u(f,j)}	$M(f1) \cup M(f2)$

f1, f2 - старые файлы; новый файл f является некоторой их функцией.

Теперь рассмотрим модель, называемую пятимерным пространством безопасности Хартстона[3]. В данной модели используется пятимерное пространство безопасности для моделирования процессов, установления полномочий и организации доступа на их основании. Модель имеет пять основных наборов:

A - установленных полномочий; U - пользователей; E - операций; R - ресурсов; S - состояний.

Область безопасности будет выглядеть как декартово произведение: $A \times U \times E \times R \times S$. Доступ рассматривается как ряд запросов, осуществляемых пользователями u для выполнения операций e над ресурсами R в то время, когда система находится в состоянии s. Например, запрос на доступ представляется четырехмерным кортежем $q = (u, e, R, s)$, $u \in U$, $e \in E$, $s \in S$, $r \subseteq R$. Величины u и s задаются системой в фиксированном виде. Таким образом, запрос на доступ - подпространство четырехмерной проекции пространства безопасности. Запросы получают право на доступ в том случае, когда они полностью заключены в соответствующие подпространства.

Процесс организации доступа можно описать алгоритмически следующим образом. Для запроса q, где q(u, e, R, s), набора U' вполне определенных групп пользователей, набора R' вполне определенных единиц ресурсов и набора P правильных (установленных) полномочий процесс организации доступа будет состоять из следующих процедур:

1. Вызвать все вспомогательные программы, необходимые для предварительного принятия решений.

2. Определить из U те группы пользователей, к которым принадлежит u. Затем выбрать из P спецификации полномочий, которым соответствуют выделенные группы пользователей. Этот набор полномочий F(u) определяет привилегию пользователя u.

3. Определить из P набор F(e) полномочий, которые устанавливают e как основную операцию. Этот набор называется привилегией операции e.

4. Определить из P набор F(R)(привилегию единичного ресурса R) - полномочия, которые определяют поднабор ресурсов из R', имеющего общие элементы с запрашиваемой единицей ресурса R.

Полномочия, которые являются общими для трех привилегий в процедурах 2, 3, 4, образуют D(q) (так называемый домен полномочий для запроса):

$$q: D(q)=F(u)\cap F(e)F(R).$$

5. Удостовериться, что запрашиваемый ресурс R полностью включается в D(q), то есть каждый элемент из R должен содержаться в некоторой единице ресурса, которая определена в домене полномочий D(q).

6. Осуществить разбиение набора D(q) на эквивалентные классы так, чтобы два полномочия попадали в эквивалентный класс тогда и только тогда, когда они специфицируют одну единицу ресурса. Для каждого такого класса логическая операция ИЛИ (или И) выполняется с условиями доступа элементов каждого класса.

Новый набор полномочий - один на каждую единицу ресурса, указанную в D(q), есть F(u, q) - фактическая привилегия пользователя u по отношению к запросу q.

7. Вычислить EAC - условие фактического доступа, соответствующего запросу q, осуществляя логическое И (или ИЛИ) над условиями доступа членов F(u, q). Это И(или ИЛИ) выполняется над всеми единицами ресурсов, которые перекрывают единицу запрошенного ресурса.

8. Оценить EAC и принять решение о доступе:

— разрешить доступ к R, если R перекрывается;

— отказать в доступе в противном случае.

9. Произвести запись необходимых событий.

10. Вызвать все программы, необходимые для организации доступа после "принятия решения".

11. Выполнить все вспомогательные программы, вытекающие для каждого случая из условия 8.

12. Если решение о доступе было положительным - завершить физическую обработку.

Автор модели Хартстон отмечает, что приведенная последовательность шагов не всегда необходима в полном объеме. Например, в большинстве реализаций шаги 2 и 6 осуществляются во время регистрации пользователя в системе.

В заключение рассмотрим проблемы моделей дискреционного доступа, продемонстрированные Харрисоном, Руззо и Ульманом. Рассмотрим типичную модель системы защиты, состоящую из следующих конечных наборов:

— общих прав $A=\{a_1, ..., a_n\}$;

— исходных субъектов S_0 и объектов O_0;

— команд C формы $\alpha(X_1, ..., X_n)$, где α - имя; $X_1, ..., X_n$ - формальные параметры, указывающие на объекты.

Элементами матрицы доступа являются права доступа, взятые из набора общих прав. Состояния системы изменяются при изменении элементов матрицы доступа M. Запросы к системе можно выразить в форме:

if a_1 in $M[s_1, o_1]$ and
a_2 in $M[s_2, o_2]$ and
...
a_m in $M[s_m, o_m]$
then
op_1,
op_2,
...
op_n.

Причем, $\forall a_i \in A$ операция op является одной из следующих примитивных операций:

— enter a into (s, 0);

— delete a from (s, 0);

— create subject s;

— create object o;

— destroy subject o;

— destroy object o.

Семантика данных операций очевидна. Для системы с начальной конфигурацией Q_0 и права a можно сказать, что система безопасна для a, если не существует последовательности запросов к системе в состоянии Q_0 таких, что в результате них право a будет записано в ячейку, не содержащую ее. Существуют две теоремы о безопасности данного типа систем. Первая относится к безопасности моно-операционных систем. Под моно-операционной

системой понимается система, в которой каждый запрос имеет только одну операцию.

Т е о р е м а . *Существует алгоритм для определения, является или нет моно-операционная система безопасной для данного права a.*

Вторая теорема указывает на то, что проблема безопасности для системы с запросами общего вида является неразрешимой.

Т е о р е м а . *Проблема определения безопасности для данного права a в системе с запросами общего вида является неразрешимой.*

Доказательство данных теорем приведено в приложении 1. Харрисон, Руззо и Ульман показали, что безопасными являются монотонные системы (системы, не содержащие операции destroy и delete), системы не содержащие операций create и моно-условные системы (системы, запрос к которым содержит только одно условие).

К достоинствам моделей дискреционного доступа можно отнести хорошую гранулированность защиты и относительно простую реализацию. В качестве примера реализаций данного типа моделей можно привести так называемую матрицу доступа, строки которой соответствуют субъектам системы, а столбцы - объектам; элементы матрицы характеризуют права доступа. Проблемы, возникающие в системах, синтезированных на их основании, показаны в следующем параграфе.

Модели мандатного доступа

Описанные в параграфе 2.1.1.1 модели дискреционного доступа хотя и обеспечивают хорошо гранулированную защиту, но обладают рядом недостатков. В частности, в системах, построенных на основе DAC, существует проблема троянских *программ (троянских коней)*. Троянскую программу следует определять как любую программу, от которой ожидается выполнение некоторого желаемого действия, а она на самом деле выполняет какое-либо неожиданное и нежелательное действие. Так, троянская программа может выглядеть как вполне хороший продукт, но реально она может оказаться даже более опасной, чем можно было бы ожидать.

Для того чтобы понять, как может работать троянский конь, вспомним, что когда пользователь вызывает какую-либо программу на компьютере, в

системе инициируется некоторая последовательность операций, зачастую скрытых от пользователя. Эти операции обычно управляются операционной системой. Троянские программы рассчитывают на то, что когда пользователь инициирует такую последовательность, он обычно верит в то, что система произведет ее, как полагается. При этом нарушитель может написать версию троянской программы, которая будучи запущенной от имени пользователя-жертвы, передаст его информацию пользователю нарушителю.

В отличие от DAC, мандатный доступ (MAC) накладывает ограничения на передачу информации от одного пользователя другому. Это позволяет разрешить проблему троянских коней.

Классической моделью, лежащей в основе построения многих систем MAC и породившей остальные модели MAC, является модель Белла и Лападула. К сожалению, данная модель не лишена недостатков и с целью устранения данных недостатков были порождены некоторые специфичные модели. В заключении параграфа мы опишем специализированные модели, основанные на рассмотрении конкретных требований, в соответствии с которыми синтезируется данная модель.

Модель Белла и Лападула

Модель, получившая название модели Белла и Лападула (БЛМ), до сих пор оказывает огромное влияние на исследования и разработки в области компьютерной безопасности. Об этом свидетельствует огромное количество различных документов, ссылающихся в библиографии на первоначальную БЛМ. Данная модель лежит в основе построения MAC. Идеи, заложенные в БЛМ, могут быть использованы при построении различных политик безопасности.

Идеи, лежащие в основе БЛМ, берут происхождение из “бумажного мира”. Белл и Лападула перенесли модель безопасности, принятую при работе с документами, в мир компьютерных систем. Основным наблюдением, сделанным Беллом и Лападулой, является то, что в правительстве США все субъекты и объекты ассоциируются с уровнями безопасности, варьирующимися от низких уровней (неклассифицированных) до высоких (совершенно секретных). Кроме того, они обнаружили, что для предотвращения утечки

информации к неуполномоченным субъектам этим субъектам с низкими уровнями безопасности не позволяется читать информацию из объектов с высокими уровнями безопасности. Это ведет к первому правилу БЛМ.

Простое свойство безопасности, также известное как правило “нет чтения вверх” (NRU), гласит, что субъект с уровнем безопасности x_s (понятие уровня безопасности введено в главе 1) может читать информацию из объекта с уровнем безопасности x_o, только если x_s преобладает над x_o. Это означает, что если в системе, удовлетворяющей правилам модели БЛМ, субъект с уровнем доступа секретный попытается прочитать информацию из объекта, классифицированного как совершенно секретный, то такой доступ не будет разрешен.

Белл и Лападула сделали дополнительное наблюдение при построении своей модели: в правительстве США субъектам не позволяется размещать информацию или записывать ее в объекты, имеющие более низкий уровень безопасности. Например, когда совершенно секретный документ помещается в неклассифицированное мусорное ведро, может произойти утечка информации. Это ведет ко второму правилу БЛМ.

*Свойство-**, известное как правило “нет записи вниз” (NWD), гласит, что субъект безопасности x_s может писать информацию в объект с уровнем безопасности x_o только если x_o преобладает над x_s. Это означает, что если в системе, удовлетворяющей правилам модели БЛМ, субъект с уровнем доступа совершенно секретный попытается записать информацию в неклассифицированный объект, то такой доступ не будет разрешен. Введение свойства-* разрешает проблему троянских коней, так как запись информации на более низкий уровень безопасности, типичная для троянских коней, запрещена.

Правило запрета по записи является большим упрощением некоторых реализаций БЛМ. Так, некоторые описания включают более детальное понятие типа доступа (например такие, как добавление и выполнение). Помимо этого многие модели БЛМ включают понятие дискретной защиты с целью обеспечения хорошо гранулированной защиты при сохранении всех преимуществ БЛМ.

Правила запрета по записи и чтению БЛМ отвечают интуитивным понятиям того, как предотвратить утечку информации к неуполномоченным источникам.

Рассмотрим формализацию БЛМ. В соответствии с определениями главы 1:

S - множество субъектов;

O - множество объектов;

L - решетка уровней безопасности;

$F : S \cup O \rightarrow L$ - функция, применяемая к субъектам и объектам; данная функция определяет уровни безопасности своих аргументов в данном состоянии;

V - множество состояний - множество упорядоченных пар (F, M), где M - матрица доступа субъектов системы к объектам.

Система представляется начальным состоянием v_0, определенным множеством запросов к системе R и функцией переходов $T : (V \times R) \rightarrow V$ такой, что система переходит из состояния в состояние после исполнения запроса. Сформулируем определения, необходимые для доказательства основной теоремы безопасности (ОТБ), доказанной для БЛМ.

О п р е д е л е н и е 1. *Состояние (F, M) безопасно по чтению (NRU) тогда и только тогда, когда для* $\forall s \in S$ *и для* $\forall o \in O$, *чтение* $\in M[s, o] \rightarrow F(s) \geq F(o)$.

*О п р е д е л е н и е 2. Состояние (F, M) безопасно по записи (NWD, *-свойство) тогда и только тогда, когда для* $\forall s \in S$ *и для* $\forall o \in O$, *запись* $\in M[s,o] \rightarrow F(o) \geq F(s)$.

О п р е д е л е н и е 3. *Состояние безопасно тогда и только тогда, когда оно безопасно по чтению и записи.*

Т е о р е м а *(ОТБ). Система* (v_0, R, T) *безопасна тогда и только тогда, когда состояние* v_0 *безопасно и T таково, что для любого состояния v, достижимого из* v_0 *после исполнения конечной последовательности запросов из R,* $T(v, c) = v^*$, *где* $v = (F, M)$ *и* $v^* = (F^*, M^*)$, *переходы системы (T) из состояния в состояние подчиняются следующим ограничениям для* $\forall s \in S$ *и для* $\forall o \in O$:

- *если чтение* $\in M^*[s, o]$ *и чтение* $\notin M[s, o]$, *то* $F^*(s) \geq F^*(o)$;
- *если чтение* $\in M[s, o]$ *и* $F^*(s) < F^*(o)$, *то чтение* $\notin M^*[s, o]$;
- *если запись* $\in M^*[s, o]$ *и запись* $\notin M[s, o]$, *то* $F^*(o) \geq F^*(s)$;
- *если запись* $\in M[s, o]$ *и* $F^*(o) < F^*(s)$, *то запись* $\notin M^*[s, o]$.

Доказательство.

1. *Необходимость*. Предположим, система безопасна. Состояние v_0 безопасно по определению. Если имеется некоторое состояние v, достижимое из состояния v_0 после исполнения конечной последовательности запросов из R таких, что T(v, c) = v^*, хотя v^* не удовлетворяет одному из двух первых ограничений для T, то v^* будет достижимым состоянием, но противоречащим ограничению безопасности по чтению. Если v^* не удовлетворяет одному из двух последних ограничений для T, то v^* будет достижимым состоянием, но противоречащим ограничению безопасности по записи. В любом случае система небезопасна.

2. *Достаточность*. Предположим, что система небезопасна. В этом случае либо v_0 должно быть небезопасно, либо должно быть небезопасно состояние v, достижимое из состояния v_0 после исполнения конечной последовательности запросов из R. Если v_0 небезопасно - все доказано. Если v_0 безопасно, допустим, что v^* - первое в последовательности запросов небезопасное состояние. Это означает, что имеется безопасное состояние v такое, что T(v, c)=v^*, где v* - небезопасно. Но это противоречит четырем ограничениям безопасности на T.

Несмотря на все достоинства, оказалось, что при использовании БЛМ в контексте практического проектирования и разработки реальных компьютерных систем возникает ряд технических вопросов. Данные вопросы являются логическим следствием достоинства БЛМ - ее простоты. Проблемы возникают при рассмотрении вопросов построения политик безопасности для конкретных типов систем, то есть на менее абстрактном уровне рассмотрения. При данном рассмотрении системный компонент модели усложняется, что может привести к неадекватности БЛМ в ее классической форме. Как следствие, в мире компьютерной безопасности ведется широкая полемика по поводу применимости БЛМ для построения безопасных систем.

Рассмотрим ряд примеров *критики БЛМ*. Некоторые из них взяты из литературы, посвященной вопросам безопасности, другие часто включаются в техническое описание и представляют собой так называемую “обязательную критику” БЛМ.

Начнем данное рассмотрение с обсуждения проблемы, возникающей в распределенных системах, удовлетворяющих правилам БЛМ. В частности,

покажем, что запрос на чтение вызывает протекание потоков информации в обоих направлениях между компонентами, что является нарушением правил модели. Затем рассмотрим проблему использования этой модели для обеспечения безопасности доверенных субъектов, которые выполняют наиболее критичные задачи в компьютерной системе. Завершим обсуждение примером описания модели, известной как система Z.

Удаленное чтение

В свете недавних тенденций использования распределенных конфигураций требуется рассматривать модели безопасности не только для автономных, но и для распределенных компьютерных систем (распределенная система обычно состоит из нескольких объединенных систем). Очевидным способом распространения БЛМ на распределенные системы будет назначение уровней безопасности различным компонентам и соблюдение гарантий выполнения правил-ограничений по чтению и записи.

Например, некоторым компонентам можно назначить уровни безопасности, меняющиеся от неклассифицированного до совершенно секретного уровня, и на основании принципов БЛМ синтезировать соединения между различными компонентами системы. Может показаться, что если конфиденциальному субъекту A будет разрешено чтение информации из неклассифицированного объекта B, никакая конфиденциальная информация не будет раскрыта. Но при более подробном рассмотрении реализации операции удаленного чтения снизу может быть сделано неприятное наблюдение. Операция чтения между удаленными компонентами приводит к протеканию потока информации от читаемого объекта к запросившему доступ на чтение субъекту. Данный поток является безопасным, поскольку информация не разглашается неавторизованному субъекту. Однако в распределенной конфигурации чтение инициируется запросом от одного компонента к другому. Такой запрос образует прохождение потока информации в неверном направлении (запись в объект с меньшим уровнем безопасности). Таким образом, удаленное чтение в распределенных системах может произойти только если ему предшествует операция записи вниз, что является нарушением правил БЛМ.

Многие исследователи рассматривают эту проблему как наиболее убедительное свидетельство неадекватности БЛМ. Однако на практике эта проблема часто является несущественной; достаточно внедрения в систему дополнительных средств обработки удаленных запросов для обеспечения того, чтобы поток информации от высокоуровневого субъекта к низкоуровневому объекту был ограничен запросом на доступ. Фактически, некоторые архитектуры предлагают отдельные компоненты, выполняющие обработку таких запросов и потока информации в распределенных системах.

Доверенные субъекты

В предыдущем описании правил БЛМ не было указано, какие субъекты должны подчиняться этим правилам. Например, компьютерные системы обычно имеют администратора, который управляет системой, добавляя и удаляя пользователей, восстанавливает функционирование после сбоев, устанавливает специальное программное обеспечение, устраняет ошибки в операционной системе или приложениях и т.п. Очевидно, что процессы, действующие в интересах таких администраторов, не могут управляться правилами БЛМ или каких-либо других моделей, не позволяющих им выполнять функции администрирования.

Это наблюдение высвечивает еще одну техническую проблему, связанную с правилами БЛМ. Можно сказать, что эти правила обеспечивают средства для предотвращения угрозы нарушения секретности для нормальных пользователей, но не говорят ничего по поводу той же проблемы для так называемых *доверенных* субъектов. Доверенные субъекты могут функционировать в интересах администратора. Также они могут быть процессами, обеспечивающими критические службы такие, как драйвер устройства или подсистема управления памятью. Такие процессы часто не могут выполнить свою задачу, не нарушая правил БЛМ. Неприменимость БЛМ для доверенных субъектов может быть выражена путем внесения поправки в данное ранее определение операций чтения и записи БЛМ. Но хотя это и делает определение более точным, оно нисколько не облегчает задачу для разработчика, желающего построить безопасный драйвер или утилиту поддержки работы администратора.

Одним из решений, рассматриваемых в литературе по безопасности, было предложение представлять и использовать для потока информации модель, требующую того, чтобы никакая высокоуровневая информация никогда не протекала на более низкий уровень. В параграфе 2.1.3 представлены примеры моделей безопасности, сфокусированных на понятиях, известных как выводимость и вмешательство. В данных моделях низкоуровневые пользователи не могут сделать выводы или затронуть работу высокоуровневых пользователей.

Проблема системы Z

Джон МакЛин [26] разработал концептуальное описание системы, названной Система Z. Данное описание показывает, что система, удовлетворяющая правилам БЛМ, может иметь ряд проблем с секретностью. Система Z выражается в терминах набора субъектов и объектов, с каждым из которых связан уровень безопасности. Совокупность уровней безопасности для каждого субъекта и объекта в некоторый момент времени описывает состояние системы. Система Z удовлетворяет БЛМ, если во всех состояниях системы комбинации уровней субъектов и объектов таковы, что в этом состоянии никакой субъект не может осуществить запись вниз или чтение сверху.

Предположив, что система Z удовлетворяет условиям БЛМ, можно быть уверенным, что любая угроза секретности будет обнаружена. Однако МакЛин указал на техническую деталь, которая не очевидна в таких системах. Если в некотором состоянии секретный субъект захотел прочитать совершенно секретный объект, то до тех пор, пока система удовлетворяет БЛМ, осуществить это будет невозможно. Но МакЛин заявляет, что ничто в БЛМ не предотвращает систему от “деклассификации” объекта от совершенно секретного до секретного (по желанию совершенно секретного пользователя).

В качестве иллюстрации можно привести следующий пример. Допустим, субъект с высокой степенью доверия A читает информацию из объекта, уровень классификации которого также равен A. Далее данный субъект понижает свою степень доверия до уровня B (A > B). После этого он может записать информацию в файл с классификацией B. Нарушения БЛМ формально не произошло, но безопасность системы нарушена.

Фактически, МакЛин описал конфигурацию, в которой все субъекты могут читать и записывать любой объект путем назначения соответствующих уровней безопасности объекта перед выполнением запросов на доступ. В такой системе, которая очевидно не обеспечивает секретность информации, все состояния могут быть рассмотрены как удовлетворяющие требованиям БЛМ.

Все описанное выше является справедливым для модели БЛМ в “ее классической формулировке”, кочующей из книги в книгу и из статьи в статью. Но в оригинальной модели, представленной авторами, было введено требование сильного и слабого спокойствия. Данные требования снимают проблему Z-системы. Рассмотрим их.

Правило *сильного спокойствия* гласит, что уровни безопасности субъектов и объектов никогда не меняются в ходе системной операции. Реализовав это правило в конкретной системе, можно легко сделать заключение, что описанный выше тип потенциальных проблем никогда не произойдет. Очевидным недостатком такой реализации в системе является потеря гибкости при выполнении операций.

Правило *слабого спокойствия* гласит, что уровни безопасности субъектов и объектов никогда не меняются в ходе системной операции таким образом, чтобы нарушить заданную политику безопасности. Это правило может потребовать, чтобы субъекты и объекты воздерживались от действий в период времени, когда меняются их уровни безопасности. Например, может потребоваться, чтобы уровень безопасности объекта никогда не менялся в то время, как к нему обращается некоторый субъект. Однако, если операция чередуется с изменением уровня безопасности, не вызывающего нарушения безопасности (например, субъект повышает свой уровень с секретного до совершенно секретного в ходе выполнения операции чтения некласифицированного объекта), то правило слабого спокойствия будет по прежнему соблюдено.

Фактически система Z описывает алгебру моделей, самой строгой из которых (основание) является БЛМ с сильным спокойствием (ни один субъект модели не может изменить свою классификацию), а самой слабой (вершина) - БЛМ в классической формулировке, без ограничений для субъектов на изменение классификации.

Специализированные модели

Как было отмечено в предыдущем параграфе, одним из недостатков, являющимся логическим следствием достоинства простоты БЛМ, является ее слишком большая абстрактность. С точки зрения требований пользователей, в реальных приложениях ограничения, накладываемые БЛМ, оказываются слишком строгими. Введение в модель доверенных процессов, позволяющих частично решить данную проблему, не является достаточным. С другой стороны, недостатком БЛМ, не рассмотренным нами ранее, является отсутствие в модели поддержки многоуровневых объектов (например наличие несекретного параграфа в секретном файле данных) и отсутствие зависящих от приложения правил безопасности. С целью устранения данных недостатков при проектировании системы передачи военных сообщений(MMS) Лендвером и МакЛином была разработана модель MMS.

Модель MMS

Приведем модель MMS в ее классической форме, не изменяя определений, данных авторами, хотя многие понятия данной модели перекликаются с понятиями, определенными в главе 1. В модели MMS используются следующие определения.

Классификация - обозначение, накладываемое на информацию, отражающее ущерб, который может быть причинен неавторизованным доступом; включающее уровни: TOP SECRET, SECRET и т.д. и множество меток ("CRYPTO", "NUCLEAR" и т.д.). Множество классификаций и отношение между ними образуют решетку.

Степень доверия пользователю - уровень благонадежности персоны. Каждый пользователь имеет степень доверия, и операции, производимые системой для данного пользователя, могут проверить степень доверия пользователю и классификацию объектов, с которыми он оперирует.

Пользовательский идентификатор - строка символов, используемая для того, чтобы отметить пользователя системы. Для использования системы пользователь должен предъявить ей пользовательский идентификатор, и система должна провести аутентификацию пользователя. Данная процедура

называется login. Каждый пользователь должен иметь уникальный идентификатор.

Пользователь - персона, уполномоченная для использования системы.

Роль - работа, исполняемая пользователем (например пользователь, имеющий право удалять, распространять или понижать классификацию объектов). Пользователь всегда ассоциирован как минимум с одной ролью в некоторый момент времени, и он может менять роль в течение сессии. Для действий в данной роли пользователь должен быть уполномочен. Некоторые роли могут быть связаны только с одним пользователем в данный момент времени (например распространитель). С любой ролью связана способность выполнения определенных операций.

Объект - одноуровневый блок информации. Это минимальный блок информации в системе, который имеет классификацию. Объект не содержит других объектов, он не многоуровневый.

Контейнер - многоуровневая информационная структура. Имеет классификацию и может содержать объекты (каждый со своей классификацией) и (или) другие контейнеры. Файл - это контейнер. Некоторые структуры файла могут быть контейнерами. Различие между объектом и контейнером базируется на типе, а не на текущем содержимом: если один из файлов данного типа является контейнером, то все остальные файлы данного типа являются контейнерами, даже если некоторые из них содержат только объекты или пусты. Устройства такие, как диски, принтеры, ленты, сетевые интерфейсы и пользовательские терминалы - контейнеры.

Сущность - Объект или Контейнер.

Требование Степени Доверия Контейнеров - атрибут некоторых контейнеров. Для некоторых контейнеров важно требовать минимум степени доверия, то есть пользователь, не имеющий соответствующего уровня благонадежности, не может просматривать содержимое контейнера. Такие контейнеры помечаются соответствующим атрибутом (CCR). Например, пользователь, имеющий степень доверия CONFIDENTAL, не может просматривать CONFIDENTAL параграф сообщения, помеченного TOP SECRET, если оно содержится в CCR контейнере. Если пользователь должен иметь возможность просматривать данное сообщение, контейнер не должен быть помечен как CCR.

Идентификатор (ID) - имя сущности без ссылки на другие сущности, например, имя файла есть идентификатор этого файла. Обычно все сущности имеют идентификатор.

Ссылка на сущность *Прямая*, если это идентификатор Сущности.

Ссылка на сущность *Косвенная*, если это последовательность двух или более имен Сущностей (из которых только первая - идентификатор). Пример - "текущее сообщение, первый абзац, вторая строка".

Операция - функция, которая может быть применена к сущности. Она может позволять просматривать или модифицировать сущность. Некоторые операции могут использовать более одной сущности (пример - операция копирования).

Множество Доступа - множество троек (Пользовательский Идентификатор или Роль, Операция, Индекс операнда), которые связаны с сущностью. Операция, которая может быть специфицирована для особых сущностей, зависит от типа данной сущности. Если операция требует более одного операнда, индекс операнда специфицирует позицию, на которой ссылка на данный операнд может появиться в операции.

Сообщение - особый тип, реализуемый в MMS. Сообщение является контейнером. Сообщение включает поля Куда, Откуда, Время, предмет, текст, автор. Чертежные сообщения включают поле чертежа.

Неформальная модель MMS

Пользователь получает доступ к системе только после прохождения процедуры login. Для этого пользователь предоставляет системе *Пользовательский идентификатор*, и система производит аутентификацию, используя пароли, отпечатки пальцев или другую адекватную технику. После успешного прохождения аутентификации *Пользователь* запрашивает у системы *Операции* для использования функций системы. *Операции,* которые *Пользователь* может запросить у системы, зависят от его *ID* или *Роли*, для которой он авторизован: с использованием *Операций Пользователь* может просматривать или модифицировать *Объекты* или *Контейнеры*. Система реализует ограничения, описанные ниже.

Предположения Безопасности

Пользователь всегда может скомпрометировать информацию, к которой он имеет законный доступ. Таким образом, надо сформулировать предположения безопасности, которые могут быть выполнены только пользователями системы.

A1. Офицер безопасности системы присваивает уровни доверия, классификацию устройств и множества ролей корректно.

A2. Пользователь вводит корректную классификацию, когда изменяет, объединяет или переклассифицирует информацию.

A3. Пользователь классифицирует сообщения и определяет множества доступа для сущностей, которые он создает, так, что только пользователь с требуемой благонадежностью может просматривать информацию.

A4. Пользователь должным образом контролирует информацию объектов, требующих благонадежности.

Ограничения безопасности

Ограничения безопасности, в отличие от предположений безопасности, должны поддерживаться не пользователями системы, а непосредственно компьютерной системой.

B1. Авторизация - пользователь может запрашивать операции над сущностями, только если пользовательский идентификатор или текущая роль присутствуют во множестве доступа сущности вместе с этой операцией и со значением индекса, соответствующим позиции операнда, в которой сущность относят в требуемой операции.

B2. Классификационная иерархия - классификация контейнера всегда больше или равна классификации сущностей, которые он содержит.

B3. Изменения в объектах - информация, переносимая из объекта, всегда наследует классификацию данного объекта. Информация, вставляемая в объект, должна иметь классификацию ниже классификации этого объекта.

B4. Просмотр - пользователь может просматривать (на некотором устройстве вывода) только сущности с классификацией меньше, чем классификация устройства вывода и степень доверия к пользователю (данное ограничение применяется к сущностям, адресуемым прямо или косвенно).

B5. Доступ к контейнерам, требующим степени доверия - пользователь может получить доступ к косвенно адресованной сущности внутри контейнера, требующего степени доверия, только если его степень доверия не ниже классификации контейнера.

B6. Преобразование косвенных ссылок - пользовательский идентификатор признается законным для сущности, к которой он обратился косвенно, только если он авторизован для просмотра этой сущности через ссылку.

B7. Требование меток - сущности, просмотренные пользователем, должны быть помечены его степенью доверия.

B8. Установка степеней доверия, ролей, классификации устройств - только пользователь с ролью офицера безопасности системы может устанавливать данные значения. Текущее множество ролей пользователя может быть изменено только офицером безопасности системы или самим пользователем.

B9. Понижение классификации информации - никакая классифицированная информация не может быть понижена в уровне своей классификации, за исключением случая, когда эту операцию выполняет пользователь с ролью "пользователь, уменьшающий классификацию информации".

B10. Уничтожение информации - операция уничтожения информации проводится только пользователем с ролью "пользователь, уничтожающий информацию".

Обсуждение

Рассмотрим работу модели в частных случаях.

1. Что запрещает пользователю копировать классифицированную сущность в неклассифицированную?

Классификация данных копируется вместе с данными. В данном случае нарушаются ограничения 2 и 9, если только пользователь не выполняет роль пользователя, уменьшающего классификацию информации. Если данная операция относится к объекту, она попадает под ограничение 3.

2. Что происходит при копировании части объекта в другой объект?

Часть объекта наследует классификацию всего объекта (ограничение 3). Таким образом, перемещение части объекта в другой объект запрещено

ограничениями 2 и 3 до тех пор, пока классификация принимающего объекта меньше классификации объекта-источника.

3. Имеет ли пользователь уровень "login"?

Уровень login - необязательная часть модели, но ее эффект может быть использован посредством классификации терминалов. Классификация терминала - верхняя граница классификации информации, которая может быть просмотрена на нем (ограничение 4). Если пользователь желает ограничить уровень классификации, просматриваемой на терминале, он может вызвать операцию, понижающую классификацию информации, появляющейся на терминале. Корректное определение классификации разделяемых устройств (диски, принтеры и т.д.) устанавливается офицером безопасности. Отметим, что ограничение классификации информации, появляющейся на терминале, не ограничивает классификации информации, с которой может работать субъект.

4. Процесс не присутствует в модели, но присутствует в реализации. Как может быть ограничена его активность?

Операции, также как процессы и программы, присутствуют в модели. Каждая функция системы должна рассматриваться пользователем как операция. При реализации процессы ограничены операциями, которые соблюдают ограничения безопасности.

5. Какие сущности в системе объекты, а какие контейнеры?

Файл - это контейнер, а группа дата-время - это объект. Если некоторые сущности данного типа - объекты, а некоторые - контейнеры, то должна быть определена функция, позволяющая определить тип сущности, принадлежащей данному семейству.

6. Как создаются сущности?

Для каждого типа сущностей, которые пользователь может создавать, в системе существуют операции, создающие экземпляр данного типа. Как и с другими операциями, это могут делать только пользователи, авторизованные для этого. В частности, только пользователи с авторизованной ролью могут создавать определенные типы сущностей.

7. Как пользователь обращается к объектам или контейнерам?

Некоторые сущности имеют идентификатор, который позволяет адресовать их непосредственно. Сущность может иметь ноль, один или более

идентификаторов. Сущность может также быть адресована косвенно с помощью квалифицированного имени.

8. Какая политика управляет доступом к сущности в контейнере?

Ответ на данный вопрос зависит от типа доступа и ссылки (прямой или косвенной). Если сущность адресуется напрямую для просмотра, срабатывает ограничение 4. Если ссылка косвенная, возможны два случая, зависящие от того, находится ли сущность в контейнере, требующем доверия. Если это так, то срабатывают ограничения 4 и 5, иначе срабатывает ограничение 4. Отметим, что пользователь может просмотреть сущность в контейнере, требующем доверия, если он обратился к ней напрямую, но не может просмотреть ее при косвенном обращении.

9. Имеется ли в системе что-то, что является не пользователем и не сущностью?

С точки зрения пользователя - нет. В реализации могут быть структуры, которым трудно назначить правильную классификацию (например системные очереди). Но все, что пользователь может создать, просмотреть или модифицировать, должно быть пользователем или сущностью.

10. В модели нет уровней целостности. Что предотвращает случайную èёè умышленную модификацию данных?

Модификация степеней доверия, множества ролей и классификаций определена данными ограничениями безопасности. Для изменения данных пользователь должен запросить операцию; ограничение 1 требует того, чтобы пользователь был авторизован для данной операции. В принципе, особые случаи можно добавить как специфичные ограничения.

Формализация модели MMS приведена в приложении 3.

Проблемы моделей предоставления прав

Наряду с неоспоримыми достоинствами моделей предоставления прав, выражающимися в их интуитивной понятности и возможности реализации с высокой степенью точности, данные модели имеют ряд недостатков.

В моделях предоставления прав возможно образование скрытых каналов утечки информации. Таким образом, несмотря на кажущуюся простоту реализации систем предоставления прав, перекрытие каналов утечки

информации является нетривиальной задачей. При анализе защищенных вычислительных систем, построенных по принципу предоставления прав, необходим тщательный анализ каналов утечки информации. Для систем высокой степени доверия данный пункт отражен в требованиях к системе.

Анализ скрытых каналов утечки информации базируется обычно на принципах анализа потоков данных в программном обеспечении (данные принципы разработаны Д. Деннинг), контроля совместно используемых ресурсов, которые могут быть применены для организации скрытых каналов утечки информации (каналы утечки информации на основе хранения) и использования программами таймеров (временные каналы утечки информации).

Хотя каналы утечки информации нетрудно обнаружить, их обычно находят уже после того, как система синтезирована. Как следствие, их ликвидация может быть затруднительна.

Информационные модели

Информационные модели определяют ограничения на отношение ввода/вывода системы, которые достаточны для реализации системы. Данные модели накладывают *ограничения на интерфейс программных модулей* системы с целью достижения безопасной реализации. При этом подробности реализации определяются разработчиком системы. Данные модели являются результатом применения теории информации к проблеме безопасности систем. К информационным моделям относятся модели невмешательства и невыводимости. Достоинствами данного типа моделей, в отличие от моделей предоставления прав, являются:

— отсутствие в них скрытых каналов утечки информации;

— естественность их использования для реализации сетевых защищенных вычислительных систем.

Теория данных математических моделей бурно развивается в настоящее время.

Модель невмешательства

Невмешательство - ограничение, при котором ввод высоко уровневого пользователя не может смешиваться с выводом низкоуровневого пользователя.

Модель невмешательства рассматривает систему, состоящую из четырех объектов: высокий ввод (high-in), низкий ввод (low-in), высокий вывод (high-out), низкий вывод (low-out).

Рассмотрим систему, вывод которой пользователю *u* определен функцией out(*u*, hist.read(*u*)), где hist.read(*u*) - история ввода системы (traces), чей последний ввод был read(u) (команда чтения, исполненная пользователем *u*). Для определения безопасности системы необходимо определить термин очищения (*purge*) историй ввода, где *purge* удаляет команды, исполненные пользователем, чей уровень безопасности не доминирует над уровнем безопасности *u*. Функция clearence(u) - определяет степень доверия к пользователю (см. 1.2).

Определение: purge - функция users × traces → traces такая, что:

— purge(u, <>) = <>, где <> - пустая история ввода;

— purge(u, hist.command(w)) = purge(u, hist.command(w)), если command(w) - ввод, выполненный пользователем w; clearence(u)≥clearence(w);

— purge(u, hist.command(w)) = purge(u, hist), если command(w) - ввод, выполненный пользователем w; clearence(u)<clearence(w).

Система удовлетворяет требованию невмешательства, если и только если для всех пользователей u, всех историй T и всех команд вывода c out(u,T.c(u)) = out(u,purge(u,T).c(u)).

Для того, чтобы проверить, удовлетворяет ли система требованиям невмешательства, было разработано множество условий («unwinding conditions»), выполнение которых достаточно для поддержки невмешательства в модели машины состояний. Хотя верификация модели невмешательства труднее, чем верификация БЛМ, после нее в системе не остается скрытых каналов утечки информации. Модель невмешательства ближе к интуитивному понятию безопасности, чем БЛМ.

При сравнении модели невмешательства с БЛМ можно отметить:

1. БЛМ слабее, чем модель невмешательства, так как модель невмешательства запрещает многие скрытые каналы, которые остаются при реализации примитивной БЛМ.

2. Модель невмешательства слабее, чем БЛМ в том, что она разрешает низкоуровневым пользователям копировать один высокоуровневый файл в

другой высокоуровневый, что БЛМ запрещает, так как при этом нарушается ее безопасность по чтению.

Было показано, что для определенных систем модель невмешательства особенно хороша в том, что если последовательность ввода X не смешивается с последовательностью вывода У, и X независима от ввода других пользователей, то I (X, У)=0, где I (X, У) - взаимная для X и У информация и представляет собой поток информации от X к Y.

Модель невыводимости

Рассмотрим модель невыводимости, также базирующуюся на рассмотрении информационных потоков в системе. Модель невыводимости выражается в терминах пользователей и информации, связанных с одним из двух возможных уровней секретности (высокий и низкий).

Система считается *невыводимо безопасной*, если пользователи с низкими уровнями безопасности не могут получить информацию с высоким уровнем безопасности в результате любых действий пользователей с высоким уровнем безопасности. Другими словами, в таких системах утечка информации не может произойти в результате посылки высокоуровневыми пользователями высокоуровневой информации к низкоуровневым пользователям. Интуитивно это определение относится не к информационным потокам, а к разделению информации. Однако такое определение безопасности не предохраняет информацию высокоуровневых пользователей от просмотра низкоуровневыми пользователями. Данное определение требует, чтобы низкоуровневые пользователи не были способны использовать доступную им информацию для получения высокоуровневой информации (это объясняет, почему определение названо невыводимостью).

Многие исследователи предложили рассматривать понятие посылки и интерпретации сигнала шире, чем понятия чтения и записи в модели БЛМ. Иными словами, чтение и запись рассматриваются в контексте этой модели как явные операции, вызываемые пользователями компьютерной системы, и выполняются определенной автоматизированной последовательностью вычислительных действий. Поскольку определение невыводимости, данное выше, неформально, то обычно появляется необходимость представить понятие

невыводимости в более строгой форме. Это делается во избежание двусмысленности или других ошибок. Поэтому рассмотрим это понятие, используя модель машины состояний с ограниченными атрибутами, как средство для более точного определения невыводимости. В частности, машина состояний будет обладать детерминированным поведением и состоять из следующих частей:

— набор пользователей с высоким или низким уровнем безопасности;

— набор возможных последовательностей системных вводов информации от пользователей и выходных реакций системы.

Допустим, что машина принимает ввод от высоко- и низкоуровневых пользователей, обрабатывает эти вводы некоторым незаданным образом и затем выдает на выходах к высоко- и низкоуровневым пользователям информацию. Возможно также, что вводят информацию и получают данные вывода одни и те же пользователи. Единственным различием пользователей является то, какой у них уровень безопасности - высокий или низкий.

Если множество вводов от пользователей в машину связано со множеством выводов, получаемых пользователями от машины каким-либо разумным образом (возможно, основываясь на времени их поступления), то тогда можно рассматривать выходную последовательность как трассировку (traces, см. модель невмешательства) системы. Безопасность невыводимости может быть определена в соответствии со множеством всех трассировок системы и множеством вводов и выводов, видимых пользователями.

Точнее, система может быть признана невыводимо безопасной, если для каждой метки безопасности x и определенной трассировки есть вторая трассировка, показывающая то же поведение, видимое пользователями с меткой безопасности меньшей или равной x, но не имеющая вводов не меньших или равных x. Другими словами, высокоуровневые вводы могут всегда быть удалены из трассировки и это не повлияет на то, что видят низкоуровневые пользователи. Можно заметить, что понятие невыводимости не охватывает ситуаций, основанных на концепции “интерпретации информации” в той степени, в которой этого можно было ожидать. Данный недостаток устраняется с помощью ограничения понятия составляющих вводов и выводов модели.

Например, предположим, что некоторая система принимает вводы и обеспечивает выводы для множества высоко- и низкоуровневых пользователей.

Каждый пользователь связан с определенным взглядом на систему (например видимые вводы и выводы) и может получить информацию, интерпретируя видимое ему поведение. Если система является невыводимо безопасной, то низкоуровневые пользователи не должны получить новой информации, если на вводе системы есть дополнительные высокоуровневые пользователи. Кроме этого, если низкоуровневые пользователи могут получить определенную информацию, основываясь на видимом ими поведении, то удаление высокоуровневых пользователей не должно изменить получаемой низкоуровневыми пользователями информации.

Вероятностные модели

Модели этого типа исследуют вероятность преодоления системы защиты за определенное время T. К достоинствам моделей данного типа можно отнести числовую оценку стойкости системы защиты. К недостаткам - изначальное допущение того, что система защиты может быть вскрыта. Задача модели данного типа - минимизация вероятности преодоления системы защиты.

Игровая модель

Игровая модель системы защиты строится по следующему принципу. Разработчик создает первоначальный вариант системы защиты. После этого злоумышленник начинает его преодолевать. Если к моменту времени T, в который злоумышленник преодолел систему защиты, у разработчика нет нового варианта, система защиты преодолена. Если нет - процесс продолжается. Данная модель описывает процесс эволюции системы защиты в течение времени.

Модель системы безопасности с полным перекрытием

Система, синтезированная на основании модели безопасности с полным перекрытием, должна иметь по крайней мере одно средство для обеспечения безопасности на каждом возможном пути проникновения в систему (рис. 2.1).

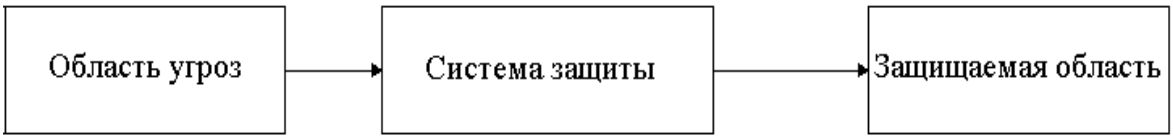

Рис. 0.1. Модель системы защиты с полным перекрытием

В модели точно определяется каждая область, требующая защиты, оцениваются средства обеспечения безопасности с точки зрения их эффективности и их вклад в обеспечение безопасности во всей вычислительной системе. Считается, что несанкционированный доступ к каждому из набора защищаемых объектов O сопряжен с некоторой величиной ущерба, и этот ущерб может быть определен количественно.

С каждым объектом, требующим защиты, связывается некоторое множество действий, к которым может прибегнуть злоумышленник для получения несанкционированного доступа к объекту. Можно попытаться перечислить все потенциальные злоумышленные действия по отношению ко всем объектам безопасности для формирования набора угроз T, направленных на нарушение безопасности. Основной характеристикой набора угроз является вероятность проявления каждого из злоумышленных действий. В любой реальной системе эти вероятности можно вычислить с ограниченной степенью точности.

Рассмотрим более строгое описание вероятностной модели, построенное на основе теории множеств. Множество отношений объект-угроза образуют двухдольный граф (рис. 2.6), в котором ребро $<t_i\ o_j>$ существует тогда и только тогда, когда t_i ($\forall t_i \in T$) является средством получения доступа к объекту o_i ($\forall o_i \in O$). Связь между объектами и угрозами типа "один ко многим", то есть одна угроза может распространяться на любое число объектов и объект может быть уязвим со стороны более чем одной угрозы. Цель защиты состоит в том, чтобы перекрыть каждое ребро графа и воздвигнуть барьер для доступа по этому пути.

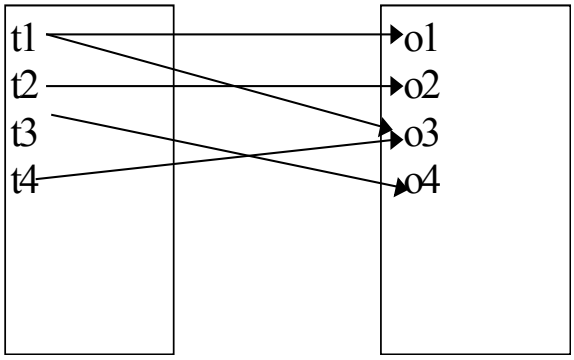

Рис. 2.6. Граф объект - угроза

Завершает модель третий набор, включающий средства безопасности M, которые используются для защиты информации в вычислительной системе. Идеально, каждое m_k ($\forall m_k \in M$) должно устранять некоторое ребро $<t_i\ o_j>$ из графа на рисунке 2.6. Набор M средств обеспечения безопасности преобразует двухдольный граф в трехдольный. В защищенной системе все ребра представляются в виде $<t_i\ m_k>$ и $<m_k\ o_j>$. Любое ребро в форме $<t_i\ o_j>$ определяет незащищенный объект. Одно и то же средство обеспечения безопасности может перекрывать более одной угрозы и (или) защищать более одного объекта. Отсутствие ребра $<t_i\ o_j>$ не гарантирует полного обеспечения безопасности (хотя наличие такого ребра дает потенциальную возможность несанкционированного доступа, за исключением случая, когда вероятность появления t_i равна нулю).

Рассмотрим базовую систему безопасности Клеменса, представляющую собой пятикортежный набор S={O, T, M, V, B},

где O - набор защищаемых объектов;

T - набор угроз;

M - набор средств обеспечения безопасности;

V - набор уязвимых мест - отображение T×O на набор упорядоченных пар $V_i=<t_i\ o_j>$, представляющих собой пути проникновения в систему;

B - набор барьеров - отображение V×M или T×O×M на набор упорядоченных троек $<t_i\ o_j\ m_k>$, представляющих собой точки, в которых требуется осуществлять защиту в системе.

Система с полным перекрытием - это система, в которой имеются средства защиты на каждый возможный путь проникновения. В такой системе $<t_i\ o_j> \in V$ предусматривает $<t_i\ o_j\ m_k> \in B$. Если это условие не соблюдено, то O_j не защищено для некоторого j.

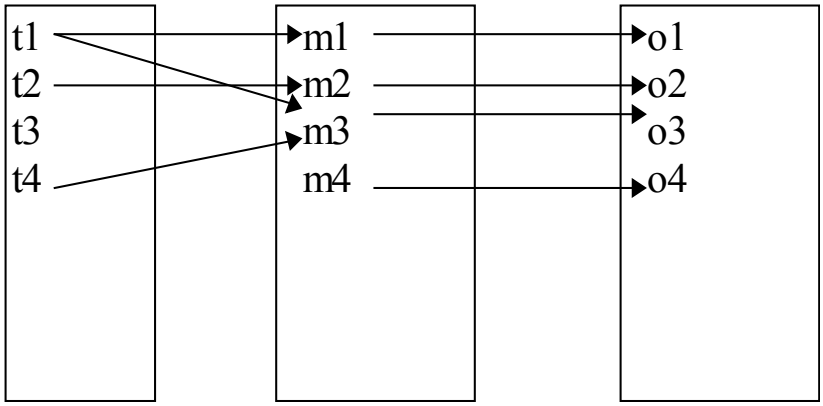

Рис. 0.7. Граф угроза - защита - объект

Таким образом, можно сделать следующие выводы. Основное преимущество вероятностных моделей состоит в возможности численного получения оценки степени надежности системы защиты информации. Данные модели не специфицируют непосредственно механизмы защиты информации, а могут использоваться только в сочетании с другими типами моделей систем защиты информации. При анализе систем защиты информации модели данного типа позволяют оценить вероятность преодоления системы защиты и степень ущерба системе в случае преодоления системы защиты. При синтезе систем защиты информации данный подход полезен тем, что позволяет минимизировать накладные расходы (ресурсы вычислительной системы) для реализации заданного уровня безопасности. Модели данного типа могут использоваться при анализе эффективности внешних по отношению к защищаемой системе средств защиты информации. Ярким примером применимости данной модели является анализ на ее основе вероятности вскрытия за конечный временной промежуток средств защиты, предлагаемых для системы MS-DOS. Для систем защиты, построенных на основании других моделей, данная модель может применяться для анализа эффективности процедуры идентификации / аутентификации.

Модели контроля целостности

Следующей по порядку, но не по значимости в нашем списке идет угроза целостности информации. Данный параграф начинается с рассмотрения

модели, аналогичной БЛМ, используемой для синтеза механизмов контроля целостности информации в системе, модели Биба (2.2.2.1.). Данное рассмотрение заканчивается рассмотрением попытки объединения моделей БЛМ и Биба в одну модель. Далее мы рассмотрим модель Кларка - Вилсона (КВМ) (2.2.2.2.). Эта модель является примером неформального выражения политики безопасности. Данная модель сформулирована в виде набора неформальных правил, и хотя в литературе она названа моделью безопасности, ее скорее можно назвать политикой контроля целостности. Параграф заканчивается рассмотрением особого случая применения механизмов контроля целостности - проблеме контроля целостности ядра системы, механизмов часовых (2.2.2.3.).

Модель Биба

При рассмотрении БЛМ было показано, что важность или чувствительность субъектов и объектов повышается с ростом в иерархии уровней безопасности. При рассмотрении моделей контроля целостности запись наверх может представлять угрозу в том случае, если субъект с низким уровнем безопасности искажает или уничтожает данные в объекте, лежащем на более высоком уровне. Поэтому, исходя из задач целостности, можно потребовать, чтобы такая запись была запрещена. Следуя подобным аргументам, можно рассматривать чтение снизу как поток информации, идущий из объекта нижнего уровня и нарушающий целостность субъекта высокого уровня. Поэтому весьма вероятно, что и такое чтение необходимо запретить.

Два этих наблюдения сделал в середине семидесятых Кен Биба. Они были последовательно внесены в модель безопасности, которая с тех пор называется моделью целостности Биба (или просто моделью Биба). Биба выразил свою модель таким же способом, каким была выражена БЛМ, за тем исключением, что правила его модели являются полной противоположностью правилам БЛМ. В этом параграфе рассмотрим три вариации модели Биба: мандатную модель целостности, модель понижения уровня субъекта и модель понижения уровня объекта. Фактически, общий термин “модель Биба” используется для обозначения любой или сразу всех трех моделей. Для

мандатной модели целостности предлагается формальное описание и дается пример системы, удовлетворяющей модели Биба для иллюстрации определения.

Мандатную модель целостности Биба часто называют инверсией БЛМ. Это довольно точное название, поскольку основные правила этой модели просто переворачивают правила БЛМ. Мы будем ссылаться на эти правила как “нет чтения снизу” (NRD) и “нет записи наверх” (NWU), и определим их в терминах субъектов, объектов, и нового типа уровней безопасности - *уровней целостности*, над которыми может быть введено отношение преобладания.

Правило NRD мандатной модели целостности Биба определяется как запрет субъектам на чтение информации из объекта с более низким уровнем целостности. NRD является полной противоположностью правила NRU БЛМ, за исключением того, что здесь используются уровни целостности, а не безопасности, как в БЛМ. Правило NWU мандатной модели целостности Биба определяется как запрет субъектам на запись информации в объект с более высоким уровнем целостности. Это правило является полной противоположностью правилу NWD БЛМ для случая уровней целостности, а не безопасности.

Одним из преимуществ этой модели является то, что она унаследовала многие важные характеристики БЛМ, включая ее простоту и интуитивность. Это значит, что проектировщики реальных систем могут легко понять суть этих правил и использовать их для принятия решений при проектировании. Кроме того, поскольку мандатная модель целостности Биба, подобно БЛМ, основана на простой иерархии, ее легко объяснить и изобразить пользователям системы.

С другой стороны, модель представляет собой очевидное противоречие с правилами NRU и NWD. Это значит, что если необходимо построить систему, которая предотвращает угрозы как секретности, так и целостности, то одновременное использование правил моделей БЛМ и Биба может привести к ситуации, в которой уровни безопасности и целостности будут использоваться противоположными способами.

Рассмотрим формальное описание модели Биба. Для этого опишем простые математические конструкции, которые помогут описать различные правила, составляющие мандатную модель целостности Биба.

Начнем с представления множества субъектов и объектов. Уровни целостности субъекта или объекта x обозначаются как уровень(x), и для них

введено отношение *преобладания*. Используя эти определения, сформулируем правила NRD и NWU мандатной модели целостности Биба в терминах булевой функции *разрешить*:

NRD: ∀ s ∈ *субъекты*, o ∈ *объекты*:

разрешить (s, o, чтение), если и только если уровень (o) *преобладает* уровень (s).

Данный тип определения предусматривает условия, при которых функция *разрешить* принимает значение истинно. Определение утверждает, что для всех определенных субъектов и объектов операция чтения разрешена только в том случае, если выполняется условие преобладания. Правило NWU просто переворачивает использование отношения преобладания, как показано в следующем определении:

NWU: ∀ s ∈ *субъекты*, o ∈ *объекты*:

разрешить (s, o, запись) ⇔ clerence(s) ≥ classification(o).

Это определение утверждает, что для всех субъектов и объектов операция записи разрешается только в том случае, если выполняется условие преобладания. Подобие определения этих двух правил правилам модели БЛМ может предоставить удобный способ для проектировщиков системы предусмотреть возможность переконфигурирования правил БЛМ таким образом, чтобы поддерживать мандатную модель целостности Биба.

Модель понижения уровня субъекта

Вторая модель Биба заключается в небольшом ослаблении правила чтения снизу. Мандатная модель целостности не позволяет субъектам с высокой целостностью читать информацию из объектов с более низкой целостностью. Это правило гарантирует, что информация из объекта с низкой целостностью не нарушит целостности субъекта. Однако в модели понижения уровня субъекта *ему разрешается осуществлять чтение снизу, но в результате такого чтения уровень целостности субъекта понижается до уровня целостности объекта.*

Мотивом для введения такого правила может являться то, что субъекты с высокой целостностью рассматриваются как “чистые”. Когда к чистому субъекту попадает информация из менее чистого источника, субъект “портится”, и его уровень целостности должен быть соответственно изменен.

Одной из характеристик этой модели является то, что она не накладывает никаких ограничений на то, что субъект может прочитать. Если, например, субъект не должен никогда переходить на более низкий уровень целостности, то не следует использовать эту модель, поскольку она может привести к такому нарушению. Если все же эта модель реализована в реальной системе, то необходимо создание некоторых дополнительных мер, предупреждающих субъекта о возможных последствиях выполнения таких операций чтения перед тем, как они будут выполнены.

Следует также заметить, что модель подразумевает монотонное изменение уровней целостности субъектов. То есть, уровни целостности субъектов или остаются неизменными, или снижаются. Иными словами, целостность субъекта может остаться прежней или ухудшиться, поскольку модель не предусматривает механизмов повышения уровня целостности субъекта.

Модель понижения уровня объекта

Последний тип модели Биба представляет собой ослабление правила для записи наверх, то есть вместо полного запрета на запись наверх эта модель разрешает такую запись, но *снижает уровень целостности объекта до уровня целостности субъекта, осуществлявшего запись*. Мотивы для такого правила те же, что и в модели понижения уровня субъекта.

Данная модель, подобно предыдущей, не накладывает никаких ограничений на то, что субъект может читать или писать. Поэтому ситуации, когда искажения объекта и понижение его уровня целостности могут вызвать серьезные последствия, не допускают использования этой модели. Например, критическая база данных, включающая данные, целостность которых имеет предельно высокое значение, не может быть реализована на основании этой модели. Если данная модель используется в реальной системе, то необходимо возложить на субъекты ответственность за деградацию объектов с высокой целостностью. Для реализации этого потребуется использование дополнительных средств обработки.

В данной модели происходит монотонное снижение уровня целостности объектов. В этой модели, как и в модели понижения уровней субъектов, не предусмотрено никаких механизмов для повышения уровня

целостности объекта. Возможно совместное использование моделей понижения уровня субъекта и объекта в одной системе.

Поскольку модель Биба так похожа на БЛМ, то она обладает большинством достоинств и недостатков этой модели. Например, обе модели просты и интуитивны и могут быть выражены простыми правилами (NRD и NWU). Кроме того, обе модели способствуют введению условий спокойствия для обеспечения того факта, что изменение меток не нарушит безопасности системы (это не относится к двум моделям понижения уровня Биба).

Однако модель Биба также обладает многими проблемами, присущими БЛМ. Например, использование модели Биба в распределенных системах может привести к двунаправленному потоку информации при удаленном чтении. Подобным образом, при отсутствии правил спокойствия, для модели Биба возникает эффект системы Z, описанный ранее. В практическом применении модель Биба слишком сильно полагается на понятие доверенных процессов, то есть проблема необходимости создания доверенных процессов для повышения или понижения целостности субъектов или объектов является весьма существенной. Эта критика последовала за критикой доверенных процессов в БЛМ.

В качестве дополнительной критики модели Биба можно упомянуть то, что она не предусматривает механизмов повышения целостности, что ведет к монотонному снижению целостности системы. Помимо этого, многие исследователи критиковали модель Биба за то, что она использует целостность как некую меру и ставили под сомнение то, что понятие “большей целостности” имеет какой-либо смысл. Их аргументом было то, что целостность субъектов и объектов следует рассматривать как двоичный атрибут, который или есть, или нет. В качестве примера они приводили утверждение, что компьютерная программа работает или правильно, или неправильно. С точки зрения логики это имеет смысл, но можно представить себе некоторые уровни правильности (например, минимальные синтаксические ошибки в программе могут снизить ее правильность намного меньше, чем значительный семантический изъян).

Объединение моделей безопасности

Теперь, после рассмотрения БЛМ и модели Биба, можно рассмотреть вопрос о том, как можно объединить на практике две и более моделей. Другими

словами, рассмотрим, как проектировщики и разработчики могут использовать различные модели безопасности при проектировании одной системы. При этом появляются вопросы о том, имеют ли эти модели взаимные противоречия, дополняют ли они друг друга, можно ли их реализовать, используя одинаковые конструкции, и т.п. Для рассмотрении вопроса об объединении двух моделей можно определить два подхода.

1. Различные модели могут быть выражены одной универсальной структурой так, что их правила работают в пределах одной модели безопасности. Обычно для этого требуется создание весьма общей структуры, которая будет достаточно полна для описания всех концепций различных моделей. Так, например, если модели БЛМ и Биба используются при разработке одной системы, то первым шагом будет объединение этих двух моделей в одну новую модель, которая будет охватывать их необходимые элементы.

2. Модели могут использоваться отдельно, и может быть проведен неформальный анализ соответствующих подходов реализации каждой модели. Это позволит сохранять независимость между отдельными моделями и произвольно объединять модели в зависимости от различных требований системы. Так, например, если модели БЛМ и Биба используются при разработке одной системы, то необходимо рассмотреть соответствующие реализации этих моделей, чтобы определить возможность их объединения.

Для каждого из этих подходов существуют свои аргументы. Создание общей универсальной структуры предоставляет средства для определения потенциальной несовместимости или избыточности различных моделей. Однако это также приводит к такой реализации модели, которая может быть запутанной и сложной в использовании. Анализ реализаций, с другой стороны, может оказаться менее полезным при сравнении логических свойств различных моделей, но он полезен при практической реализации систем, основанных на этих моделях. Приведем пример использования второго подхода для иллюстрации возможности объединения моделей БЛМ и Биба.

Для этого рассмотрим несколько технических вопросов, касающихся объединения моделей БЛМ и Биба в системе, которая должна решать как проблемы секретности, так и целостности.

Первым подходом к решению этой проблемы может являться создание системы, использующей одну политику безопасности, полученную в результате

объединения правил моделей БЛМ и Биба, то есть такая политика будет включать в себя правила из обоих моделей. Для этого потребуется, чтобы субъектам и объектам были назначены различные уровни безопасности и целостности, что позволит проводить оценку каждой модели отдельно. Однако для этого потребуется создание и управление двумя различными наборами уровней в соответствии с двумя различными наборами правил.

Другим подходом может быть логическое объединение правил этих моделей в одну модель безопасности, основанную на одном уровне как для безопасности, так и для целостности. Это приведет к правилам равного чтения и равной записи, которые удовлетворяют обеим моделям, но значительно снизится гибкость операций чтения и записи в компьютерной системе, использующей такую модель.

Еще один подход основан на использовании одного уровня как для целостности, так и для безопасности. В этом случае уровень безопасности обрабатывается в соответствии с БЛМ, но с помощью ловкого подхода будет обеспечиваться также и контроль целостности. Этот подход подразумевает размещение субъектов и объектов с высокой целостностью на нижней ступени иерархии безопасности. Поскольку субъекты и объекты с высокой целостностью находятся внизу иерархии, а компоненты с низкой целостностью - наверху иерархии, то правила NRU и NWD имитируют мандатную модель целостности Биба в структуре БЛМ, то есть чтение сверху в иерархии БЛМ является чтением снизу в иерархии модели Биба. Аналогично, запись наверх в БЛМ является записью вниз в модели Биба.

На практике это позволяет разместить важные системные файлы и администраторов в нижней части иерархии БЛМ. Это защищает целостность важных объектов от обычных пользователей, поскольку правило NWD не позволяет им осуществлять запись в системные файлы. Кроме этого, если рассматривать исполнение как чтение, то администраторы не смогут исполнять программы вне высшего уровня целостности (то есть нижнего уровня иерархии БЛМ). Это обеспечивает дополнительную защиту целостности для администраторов.

Данная схема обеспечивает защиту системных файлов и администраторов от троянских коней. Если троянский конь находится на одном из верхних уровней, он никогда не сможет исказить системные файлы за счет

правила NWD. Таким образом осуществляется защита целостности от троянских коней. Очевидно, такое объединение моделей осуществляет защиту секретности для верхних уровней определенной иерархии и защиту целостности для нижних уровней.

Модель Кларка-Вилсона

В 1987 г. Дэвид Кларк и Дэвид Уилсон представили модель целостности, которая существенно отличалась от уровне-ориентированных моделей безопасности БЛМ и Биба[11]. Созданию этой модели, которая известна как модель Кларка-Вилсона (КВМ), способствовал анализ методов управления коммерческими организациями целостностью своих бумажных ресурсов в неавтоматизированном офисе, то есть был рассмотрен ряд хорошо известных методов учета и сделана попытка распространения их на случай компьютерных приложений. Получившаяся модель целостности представляет собой руководство разработчикам и проектировщикам компьютерных систем для обеспечения целостности определенных вычислительных ресурсов.

Модель КВМ выражается в терминах набора правил функционирования и обслуживания данного компьютерного окружения или приложения. Эти правила вырабатываются для обеспечения уровня защиты целостности для некоторого заданного подмножества данных в этом окружении или приложении. Критическим понятием модели КВМ является то, что эти правила выражаются с использованием так называемых правильно сформированных транзакций, в которых субъект инициирует последовательность действий, которая выполняется управляемым и предсказуемым образом.

В этом параграфе представим несколько исходных концепций модели КВМ, включая правильно сформированные транзакции, и опишем основные правила модели КВМ. Определение каждого правила включает обсуждение проблем практической реализации. Затем следует оценка практического применения модели КВМ. В последнем разделе обсуждается возможность объединения моделей КВМ и Биба.

Исходные концепции модели КВМ

Представим модель КВМ с помощью строгого описания основных компонентов, включенных в модель. Для облегчения обсуждения мы будем

использовать некоторые понятия теории множеств, которые несколько упростят концепции, включенные в оригинальную презентацию модели КВМ.

Модель КВМ выражается в терминах конечного множества, которые мы будем обозначать как D (для данных), которые включают все наборы данных в определенной компьютерной системе. Например, если рассматривается операционная система общего назначения, то D будет обозначать все файлы, структуры и другие хранилища информации, управляемые операционной системой. В предыдущих обсуждениях мы ссылались на такие компоненты, как объекты.

Чтобы различать данные, обладающие и не обладающие целостностью, создатели модели разделили D на два непересекающиеся подмножества, которые называются *ограниченные элементы данных* (CDI) и *неограниченные элементы данных* (UDI). Это можно изобразить следующими определениями:

$$D = CDI \cup UDI$$

$$CDI \cap UDI = \varnothing.$$

Первое определение показывает, что D является объединением CDI и UDI, а второе определение показывает, что нет элементов, принадлежащих и CDI, и UDI. Набор D разделен таким образом, потому что мы хотим показать, как может меняться целостность данных, то есть данные, не имеющие целостности и находящиеся поэтому в UDI, могут быть некоторым образом модернизированы, так чтобы иметь целостность и находиться соответственно в CDI. Для упрощения нашего обсуждения мы будем ссылаться на элементы CDI и UDI.

Субъекты включены в модель как множество компонентов, которые могут инициировать так называемые *процедуры преобразования*. Процедура преобразования определяется как любая ненулевая последовательность элементарных действий. Элементарное действие, в свою очередь, определяется как переход состояния, который может вызвать изменение некоторых элементов данных. Например, субъекты могут устранять элементы данных, изменять информацию в элементах данных, копировать их и т.д. Каждая из этих операций называется процедурой преобразования (или просто ПП), поскольку действительный способ, которым каждая из них выполняется, включает в себя

последовательность элементарных действий (например, копирование A в B обычно состоит из таких операций, как чтение A, создание B, запись в B).

Если мы будем ссылаться на множество субъектов как на *субъекты*, то ПП могут быть представлены как функции, ставящие в соответствие субъект и элемент данных с новым элементом данных следующим образом (A×B является множеством пар (a, b), где $a \in A$, $b \in B$):

ПП: *субъекты* × D → D.

ПП являются просто действиями, которые выполняют над данными субъекты, способные изменить определенные данные. Чтобы проиллюстрировать данное определение, мы можем определить некоторую ПП, называемую *копирование*, который обозначает свойство: $s \in$ субъекты, $d \in D$: копирование (s, d) = d.

Правила модели КВМ

Используя вышеуказанные понятия, мы можем теперь рассмотреть основные правила, составляющие модель КВМ. Модель КВМ можно рассматривать как набор из девяти правил, которые мы выразим с помощью представленных выше понятий. В ходе обсуждения предполагается, что наши замечания делаются в соответствии с интересующей нас компьютерной системой. Также предполагается, что правила приняты все вместе, так что любое правило может ссылаться на любое другое правило без каких-либо проблем.

Первое правило гласит, что система должна содержать так называемые *процедуры утверждения целостности* (IVP). IVP утверждают, что данный CDI имеет надлежащий уровень целостности. Таким образом, IVP можно рассматривать как средство для доказательства целостности CDI.

Правило 1. В системе должны иметься IVP, утверждающие целостность любого CDI.

Можно представить себе IVP как некий тип процедуры проверки для утверждения целостности каждого CDI и подтверждения отсутствия целостности каждого UDI. Простейшим примером такой процедуры утверждения является проверка контрольной суммы. При использовании этого подхода вычисляется контрольная сумма некоей хранимой информации, и

копии этой информации сравниваются с оригиналом путем проверки соответствующих контрольных сумм. Различия в контрольных суммах сигнализируют о внесении изменений.

Одной из проблем, возникших при попытках реализации этой модели в реальных системах, является неясность того, как такие IVP могут быть сконструированы и реализованы. Например, если множество D содержит программное обеспечение (ПО), и мы используем просмотр кода для гарантии целостности CDI, содержащего это ПО, то возникает вопрос об ограниченности просмотра кода. Это остается широким полем для исследователей компьютерной безопасности.

Второе правило модели КВМ гласит, что когда любая ПП применяется к любому CDI , то производимые изменения не должны приводить к снижению целостности этих данных.

Правило 2. Применение любой ПП к любому CDI должно сохранять целостность этого CDI.

Это правило можно рассматривать как свойство скрытия применения ПП над CDI, то есть любое применение ПП над CDI не приведет к нарушению целостности CDI.

Обратите внимание, что это правило не указывает, как каждая ПП поддерживает скрытие в пределах множества CDI. Это происходит потому, что ничего не говорится о том, как применение ПП влияет на целостность других CDI. Более сильным способом определения правила является гарантия того, что каждая ПП сохраняет целостность каждого CDI, а не только того CDI , который изменяет данная ПП. Это бы гарантировало, что никакая ПП не влияет на целостность, но это гораздо труднее реализовать.

Третье правило определяет защитное ограничение на то, какие процедуры могут влиять на множество CDI.

Правило 3. Только ПП может вносить изменения в CDI.

Другими словами, процедуры и действия, не являющиеся ПП, не могут изменить CDI. Это обеспечивает замкнутость в пределах набора CDI. Обратите внимание на отличия от модели Биба понижения уровня объектов. Модель Биба позволяет субъектам с меньшей целостностью изменять объекты с более высокой целостностью, что нарушает их целостность. Данное правило модели КВМ не позволяет субъектам с низкой целостностью, то есть не использующим

ПП, изменять объекты с высокой целостностью, то есть CDI. В этом плане модель КВМ подобна мандатной модели целостности Биба.

Четвертое правило определяет, какие субъекты могут инициировать ПП над соответствующими данными в CDI.

Правило 4. Субъекты могут инициировать только определенные ПП над определенными CDI.

Это правило предполагает, что система должна определять и поддерживать некоторые отношения между субъектами ПП и CDI, так называемые КВМ-тройки. Каждая такая тройка определяет возможность данного субъекта применить данную ПП к данному CDI. Например, если (s, t, d) является элементом отношения, то субъекту s разрешается применить ПП t к CDI d. Если же эта тройка не является элементом отношения, то такой тип применения ПП будет запрещен. Это правило гарантирует, что всегда можно определить, кто может изменить CDI и как это изменение может произойти.

Пятое правило накладывает дополнительное ограничение на отношения КВМ-троек.

Правило 5. КВМ - тройки должны проводить некоторую соответствующую политику разделения обязанностей субъектов.

Это правило предусматривает, что компьютерная система определяет такую политику, чтобы не позволять субъектам изменять CDI без соответствующего вовлечения других субъектов. Это предотвращает субъектов от возможности наносить ущерб целостности CDI. Некоторые системы управления конфигурацией предоставляют уровень разделения обязанностей. Например, в некоторых системах разработчики ПО должны представить свои модули на просмотр менеджеру по разработке ПО перед тем, как они смогут включить их в конфигурацию. Этот подход защищает целостность конфигурации ПО. Однако ничто в модели КВМ не предотвращает от использования неграмотной политики разделения обязанностей.

Шестое правило предоставляет способ модернизации UDI в CDI.

Правило 6. Некоторые специальные ПП могут превращать UDI в CDI.

Это правило позволяет определенным ПП получать на вход UDI и после соответствующего повышения целостности выдавать на выходе CDI. Однако, как и для IVP, остаются неясными способы реализации этого механизма.

Седьмое правило накладывает требование, что все случаи применения ПП регистрируются в специально предназначенном для этого CDI.

Правило 7. Каждое применение ПП должно регистрироваться в специальном CDI, в который может производиться только добавление информации, достаточной для восстановления картины о процессе работы этого CDI.

Это правило требует ведения специального регистрационного журнала, который хранится в определенном CDI.

Восьмое правило накладывает требование аутентификации субъектов, желающих инициировать ПП.

Правило 8. Система должна распознавать субъекты, пытающиеся инициировать ПП.

Это правило определяет механизмы предотвращения атак, при которых один субъект пытается выдать себя за другого.

Последнее правило накладывает административное требование о том, кому дозволено менять списки авторизации (например КВМ- тройки).

Правило 9. Система должна разрешать производить изменения в списках авторизации только специальным субъектам (например офицерам безопасности).

Это правило гарантирует, что основная защита, определяемая КВМ-тройкой, не будет обойдена злоумышленником, пытающимся изменить содержание такого списка.

Можно сказать, что указанные выше девять правил определяют, как может быть проверена целостность, как и кем могут изменяться CDI и как UDI могут быть превращены в CDI.

Следует заметить, что некоторые исследователи ставят под сомнение необходимость включения в модель регистрационных, административных и аутентификационных правил на том основании, что они не дают никакой выгоды с логической точки зрения. Если в модель включена интерактивная регистрация (например запись всех действий в защищенное расписание) для обеспечения корректного использования ПП, то это может привести к невыполнению одного из других правил. Это остается областью активных исследований и дебатов.

Основным преимуществом модели КВМ является то, что она основана на проверенных временем бизнес-методах обращения с бумажными ресурсами. Поэтому модель КВМ не следует рассматривать как академическое исследование, а скорее как комплекс существующих методов. Модель КВМ также предоставляет исследователям методы работы с целостностью, отличные от традиционных уровне-ориентированных подходов, таких как модели БЛМ и Биба.

Основным недостатком модели КВМ является то, что IVP и методы предотвращения CDI от искажения целостности нелегко реализовать в реальных компьютерных системах. Конечно, эти принципы легко реализовать в ограниченном наборе приложений. Например, в случае стекового приложения IVP можно реализовать путем анализа длины стека на основании всех операций push и pop для определения правильной длины стека. Кроме того, ограничения на ПП можно реализовать за счет использования абстрактного типа данных, для которых единственно возможными операциями являются push и pop.

Однако в менее тривиальных приложениях использование IVP и ПП затруднительно. Мы уже упоминали ограничения на просмотр кода как проблему при создании IVP для ПО. Подобные ограничения существуют для других потенциальных методов проверки ПО, таких как контрольная сумма, синтаксическая проверка, анализатор качества ПО и т.д. Заметьте, однако, что даже если эти методы и не гарантируют целостность полностью, они предоставляют дополнительный уровень обеспечения целостности.

Все недостатки данной модели вытекают из ее неформализованности. Ее можно применять при проектировании систем для спецификации пользовательских приложений и использовать на соответствующем уровне иерархии рассмотрения защищенной вычислительной системы.

Объединение модели КВМ с моделью Биба

Дополнительным преимуществом модели КВМ является возможность ее объединения с другими моделями безопасности. Мы рассмотрим один из многих возможных подходов к объединению моделей в терминах разработки ПО, в котором программный продукт можно разделить:

1) на код ПО;
2) прочие вспомогательные данные, файлы и т.д.

Предположим, что компьютерная система, в которой будет защищаться это ПО и данные, обладает многоуровневой целостностью. Однако для наших целей достаточно использования только двух уровней целостности, то есть мы будем считать, что разрабатываемое ПО расположено на высоком уровне целостности, а все, что не относится к нему - на низком уровне. Субъекты верхнего уровня называются администраторами.

Мы также будем считать, что субъекты могут располагаться на обоих уровнях целостности и что администрирование ПО включает в себя обычный набор действий (например управление конфигурацией, учет состояния конфигурации, запросы на модификацию и т.д.). Также необходимо расположить основные инструменты разработки ПО, включая компиляторы, ассемблер и т.д. на верхнем уровне целостности. Используя эти предположения, мы можем теперь рассмотреть специфический подход к защите целостности на основе моделей KBM и Биба.

Защита по модели Биба между уровнями целостности. Первый тип защиты в нашей модели основан на мандатной модели целостности Биба, то есть мы гарантируем, что уровни целостности обрабатываются в соответствии с правилами no read down и no write up. Кроме того, выполнение мы рассматриваем так же, как чтение, следовательно, действует правило no execute down (“нет исполнения снизу”).

Описанные методы имеют те же преимущества, что и модель Биба. В контексте нашего примера пользователи с низким уровнем не могут изменять ПО никаким образом, будь то злонамеренно или случайно. Таким образом, система оказывается защищенной от вирусов и троянских коней, расположенных на низком уровне целостности.

Однако один из видов атак подразумевает спровоцированное выполнение администратором скрытой программы, что может иметь разрушительные последствия. Этот тип атак также предотвращается моделью Биба, поскольку правило no execute down не позволяет системным администраторам с высоким уровнем целостности выполнять программы нижнего уровня.

Защита по модели Clark-Wilson в пределах уровней целостности. Модель Биба эффективно защищает ПО от атак субъектов, находящихся на другом уровне целостности, но не предоставляется никакой защиты ПО от атак в пределах уровня целостности, то есть если злоумышленник находится на высоком уровне целостности (возможно, в результате ошибки при определении уровня целостности для данного субъекта), то этот субъект может стать причиной различных проблем с целостностью ПО.

В результате мы можем выработать некоторые механизмы, которые связаны с моделью КВМ. Точнее, мы можем рассматривать каждый уровень целостности как состоящий из множества субъектов и множества объектов, которые мы будем интерпретировать как набор CDI. Наша цель - установить контроль по модели КВМ над множеством CDI на верхнем уровне целостности, чтобы обеспечить защиту этих CDI от субъектов.

Данные типы управления обеспечивают защиту целостности в соответствии с моделью КВМ.

КВМ-тройки. Для субъектов должны быть определены отношения КВМ-троек, ПП (которые осуществляют чтение и запись ПО) и CDI в пределах каждого из уровней целостности, чтобы обеспечить соблюдение подходящей политики. Такая политика будет реализовывать множество защитных требований, соответствующих группам субъектов, ПП, и CDI в пределах каждого уровня.

Разделение обязанностей. Подходящая политика разделения обязанностей должна быть определена и проводиться для КВМ-троек в пределах уровней для обеспечения повышенного уровня защиты целостности. Примером разделения обязанностей может послужить требование того, что никакой субъект не может изменить CDI без вовлечения другого субъекта.

Механизм защиты от угрозы отказа в обслуживании

До настоящего времени большинство исследований компьютерной безопасности было связано с угрозами раскрытия и целостности. Одной из причин такого внимания этим угрозам является то, что различные международные организации по защите определяли раскрытие и целостность как основные угрозы. В результате, большинство средств, выделяемых на

исследования, тратилось на работы именно в этих направлениях. Другая причина состоит в том, что поскольку отказ в обслуживании очень тесно связан с существующими понятиями доступности системы и разработки систем реального времени (многие годы изучавшимися исследователями и разработчиками), исследователи безопасности все же должны найти подходящую нишу в этой области анализа вычислительных систем.

Несмотря на эти преграды, в области отказа в обслуживании были сделаны некоторые предварительные шаги. В этом параграфе рассмотрим понятия, связанные с описанием и предотвращением угрозы отказа в обслуживании (ОВО). В частности, определим ряд терминов, среди которых важное место занимает понятие максимального времени ожидания, впервые введенное Virgil и Gligor. Требования ОВО будут рассмотрены с использованием положений временной логики. Далее представим мандатную модель ОВО “никаких отказов вверх” (NDU) и сравним ее с аналогичными уровневыми моделями безопасности для угроз раскрытия и целостности. Параграф завершает краткое описание модели распределения ресурсов Миллена, которую можно использовать для описания моделей и стратегий ОВО.

Основные понятия ОВО

Для того чтобы понять угрозу отказа в обслуживании, необходимо вспомнить, что безопасные вычислительные системы служат промежуточным звеном при запросе услуг пользователями посредством монитора пересылок. Такой промежуточный монитор пересылок позволяет рассмотреть запросы услуг в терминах простой модели, в которой пользователи являются зарегистрированными либо незарегистрированными, и обеспечиваются запрашиваемой услугой либо получают отказ. В случаях, когда зарегистрированным пользователям не предоставляется запрашиваемая услуга, говорят, что имеет место отказ в обслуживании.

Gligor первым отметил, что в понятия предоставления или отказа в обслуживании должно быть включено время. Добавление времени к простой модели запроса услуг основано на том, что каждая услуга должна быть связана с некоторым периодом времени, называем *максимальным временем ожидания* (MWT). Для некоторой услуги MWT определяется как длина промежутка

времени после запроса услуги, в течение которого считается приемлемым предоставление этой услуги.

Можно трактовать приведенное выше определение по-другому. А именно, рассматривать MWT как период времени, в течение которого запрашиваемая услуга не устаревает. Иными словами, если после запроса услуга обеспечивается в течение слишком долгого времени, то может случиться так, что ее нельзя будет больше использовать. Заметим, что хотя приведенное выше определение и не выражено в терминах пользователей, запрашивающих услуги, может оказаться, что для данной услуги MWT будет различным для различных пользователей.

Рассмотрим пример, иллюстрирующий MWT. Предположим, что самолет имеет на борту контроллер полета, запрашивающий информацию о положении у узла измерения инерции (например у гироскопа). Очевидно, что во время полета информация о положении должна предоставляться на контроллер немедленно. Например, если контроллер запрашивает информацию о положении, в то время как самолет поворачивается в некотором направлении, то из-за любого отказа информация о положении устаревает по мере того, как меняется положение самолета. Таким образом, услуге должно соответствовать маленькое MWT, а предоставление информации должно происходить прежде, чем пройдет MWT.

Дав определение MWT, мы можем теперь ввести точное определение угрозы ОВО, а именно, будем говорить, что имеет место угроза ОВО всякий раз, когда услуга с соответствующим максимальным временем ожидания (MWT) запрашивается зарегистрированным пользователем в момент времени t и не предоставляется этому пользователю к моменту времени (t + MWT). Это означает, что ответы на запросы зарегистрированных пользователей не должны опаздывать.

При более тщательном рассмотрении угрозы ОВО и предложенного понятия MWT возникают некоторые вопросы. Например, угрозы ОВО можно полностью избежать, если определить MWT для всех услуг равным бесконечности. Однако этот подход не годится для тех случаев, когда величина MWT определяется некоторой значимой операционной характеристикой среды (например в ситуации с измерением инерции самолета). Кроме того, значение MWT можно определить только для конкретного набора услуг, для которых оно

необходимо, то есть можно определить некоторое подмножество активов системы как особенно критическое; тогда значения MWT будут соответствовать услугам, связанным с этими критическими активами.

Выражение требований ОВО во временной логике

Для того чтобы проиллюстрировать приведенные выше понятия ОВО, рассмотрим простое требование ОВО для типичной вычислительной системы и выразим его, используя расширение логики предикатов первого порядка, называемое временной логикой.

Требование ОВО будет выражено по отношению к системе, которая состоит из набора субъектов, активно инициирующих запросы услуг в различных состояниях системы. Значение MWT для каждого запроса услуги r может быть получено вычислением функции mwt(r), областью определения которой являются все запросы услуг, а областью значений - множество положительных целых чисел. Если максимальное время ожидания услуги равняется некоторому целому числу n, то удовлетворение запросов этой услуги должно происходить раньше, чем через n + 1 единицу времени.

Требование заключается в том, что если mwt(r) = n, то удовлетворение запроса услуги r в момент времени t должно произойти к моменту времени t +n. Это требование легко формализуется в терминах временной логики использованием эвентуального оператора (обозначаемого a). Этот оператор определяется следующим образом: если P - предикат в логике первого порядка, то aP - утверждение во временной логике. Если утверждение aP истинно для некоторого состояния s, это означает, что P должно быть истинно для состояния s или для некоторого состояния, имеющего место после s. Можно установить простые временные границы, определяя оператор a следующим образом: если a(t)P истинно для некоторого состояния, это означает, что P должно быть истинно в некотором последующем состоянии, которое достигается раньше, чем через t единиц времени.

Для заданных операторов можно определить требование в приведенном выше примере. Предположим, что всякий раз, когда субъект производит запрос услуги r в некотором состоянии, предикат REQ(r) является истинным. Можно предположить, что этот предикат не будет истинным в последующих

состояниях, в которых ответ на запрос может все еще ожидаться. Более того, предположим, что всякий раз, когда запрос услуги r удовлетворяется в некотором состоянии, истинным является предикат GRANT(r). Для упрощения примера предположим далее, что каждый запрос услуги единственен (это позволит облегчить измерение времени от запроса до завершения). Наш пример можно описать следующим образом:

$$REQ(r) \rightarrow \Diamond (mwt(r)) GRANT(r).$$

Мандатная модель ОВО

Теперь опишем в общих чертах мандатную модель ОВО, включающую в себя многие характеристики моделей БЛМ и Биба. Система услуг, которая будет использоваться для представления этой модели ОВО, выражается в знакомых терминах субъектов и объектов, которые использовались для описания моделей БЛМ и Биба.

Субъектам системы соответствуют приоритеты, которые могут быть одинаковы, ниже или выше по сравнению с приоритетом любого другого субъекта. Объектам соответствуют степени критичности, имеющие аналогичную иерархическую структуру. Субъект может требовать услугу у вычислительной системы, запрашивая доступ к объектам системы. Говорят, что субъект получает отказ в обслуживании, если его запрос зарегистрирован, но не удовлетворен в течение соответствующего MWT.

При описании модели необходимо рассмотреть условия, при которых один субъект может отказать в обслуживании другому субъекту. Такой отказ может быть вполне приемлем для одних случаев и совершенно не допустим для других. Например, администратор может иметь вполне подходящее оправдание, отказывая зарегистрированному пользователю в обслуживании, а нарушитель, как правило, не должен иметь такого оправдания. Правила, составляющие модель, нацелены на то, чтобы определить условия, при которых отказ в обслуживании был бы недопустим.

Рассмотрим правила, описывающие мандатную модель ОВО. Эти правила описывают взаимоотношения субъектов, аналогичные отношениям между субъектами и объектами в моделях БЛМ и Биба. Первое правило - “никаких отказов вверх” (NDU) - основано на том наблюдении, что никаким

объектам с более низким приоритетом не позволено отказывать в обслуживании субъектам с более высокими приоритетами. Однако некоторым субъектам с более высоким приоритетом (например администраторам системы) должна предоставляться возможность отказывать в обслуживании объектам с более низким приоритетом, если первые того желают.

Второе правило, являющееся просто обобщением первого, представляет собой альтернативу, которую можно использовать в тех приложениях, для которых защищенным от угроз отказа в обслуживании должно быть лишь некоторое подмножество объектов. Таким образом, это правило учитывает то, что проблема отказа в обслуживании может стоять только для объектов из некоторого конкретно определенного множества, то есть это правило позволяет обеспечить защиту от отказа в обслуживании только для особого множества объектов.

Это более общее правило требует, чтобы субъекты с более низкими приоритетами не препятствовали запросам услуг субъектов с более высокими приоритетами, производимыми через объекты из некоторого конкретно определенного множества. Это множество, которое мы обозначим через C, обычно содержит те объекты, которые являются наиболее критическими и для которых предоставляемые услуги никогда не должны устаревать.

Правило NDU(C) утверждает, что ни один субъект не может отказать запросам, сделанным субъектом с более высоким приоритетом через объекты из множества C. Второе правило особенно полезно для вычислительных систем, в которых необходимо обеспечивать защиту ОВО только для выбранного множества критических услуг. Например, система, выполняющая некоторую определенную роль, может содержать лишь небольшое множество услуг, напрямую связанных с этой ролью. В результате защиту ОВО с использованием правила NDU(C) можно обеспечить только для этих критических услуг. Это значительно сократит стоимость и трудность реализации стратегии ОВО.

Главным преимуществом двух представленных правил ОВО является то, что они дают средство для предотвращения отказа в обслуживании на основе понятия (приоритета), предположительно уже существующего для данной системы. Например, для большинства операционных систем существует понятие приоритета процессов. Данные правила также являются гибкими в том смысле, что их легко можно приспособить к данной системе. Ограничение на

объекты в правиле NDU(C) можно выразить в терминах какого-либо определения критичности объекта.

Недостаток этих правил заключается в том, что они имеют смысл только для систем, в которых можно определить несколько приоритетов. Если это не так, тогда должны быть определены подходящие аналогичные правила внутри уровня с одним приоритетом. Например, формулировка этих правил для ПЭВМ с одним единственным пользователем не будет иметь большого смысла. Кроме того, как это было в случае модели Clark-Wilson, реализация методов, необходимых для гарантии того, что провозглашенные правила имеют силу, нетривиальна.

И наконец, оказывается, что данные правила содержат многие характеристики (положительные и отрицательные) моделей BLP и Биба. Например, можно легко построить системы типа Системы Z, отвечающие требованиям правил NDU и NDU(C), но не согласующиеся с функциями защиты от отказа в обслуживании.

Модель Миллена распределения ресурсов (MPP)

Приведенные выше правила ОВО касаются условий, которых необходимо избегать, если мы хотим гарантировать, что действия нарушителя не создадут условий для отказа в обслуживании. Однако среди тех ситуаций, которые могут вызвать угрозу отказа в обслуживании, встречается много таких, которые затрагивают вопросы времени и пространства, не выявляемые простыми правилами типа приведенных выше.

Джонатан Миллен представил модель распределения ресурсов, позволяющую определить правила и стратегии отказа в обслуживании в терминах детального распределения ресурсов, включающего предоставление услуг пользователям вычислительных систем. Модель Миллена отличается от традиционных уровневых правил в моделях БЛМ и Biba, поскольку в ее основе лежит идея о том, что для выполнения нужного задания субъектам необходимы определенные пространственные и временные требования к ресурсам. Отказ в обслуживании происходит в том случае, если распределение пространства и времени для некоторого процесса не отвечает соответствующим требованиям. Миллен показал, что применительно к его модели можно легко объяснить такие

понятия, как конечное время ожидания (FWT) и максимальное время ожидания (MWT).

Модель Миллена включает в себя вполне соответствующий стратегии отказа в обслуживании набор правил, характеризующих семейство вычислительных систем, то есть эти правила построены таким образом, чтобы включить в это семейство понятия, которые во многом помогли бы при описании и анализе стратегий отказа в обслуживании. MPP и соответствующие правила представлены ниже.

Пусть P - множество активных процессов, R - множество пассивных ресурсов, c - некоторая фиксированная граница, используемая для обозначения общего максимального числа единиц всех типов ресурсов, доступных в исследуемой системе. Через вектор распределения A_p обозначим для каждого ресурса число единиц ресурса, выделенных для процесса p в некотором состоянии. Таким образом, вектор распределения можно рассматривать как своего рода отображение выделения ресурсов для процесса. Для формирования информации о том, является ли процесс текущим или застывшим, используется особый тип ресурсов - ресурс ЦПУ. В частности, всякий раз, когда $A_p(CPU) = 1$, будем говорить, что истинным является running(p), а всякий раз, когда $A_p(CPU) = 0$, будем говорить, что истинным является asleep(p).

Вектор пространственных требований ${}^{S}Q_p$ означает число единиц каждого ресурса, требуемое процессом p для выполнения необходимого задания в некотором состоянии. Предполагается, что процессы могут определять множество ресурсов, необходимых им для завершения работы, до того, как они ее начнут. Функция T(p) показывает, когда в последний раз изменились часы для процесса с целью отражения реального времени. Вектор временных требований ${}^{T}Q_p$ обозначает объем времени, необходимого каждому ресурсу процесса p для выполнения работы. Как и в случае пространственных требований, предполагается, что процесс может определять временные требования для конкретного задания.

Ниже перечислены восемь правил, составляющих модель Миллена. Каждое правило ограничивает семейство систем, соответствующих этой модели. Прежде чем вводить эти правила, важно заметить, что “помеченные” переменные (например running(p)) показывают значение переменной после одного перехода.

$$(R1)\ \mathbf{S}_{\{p \in P} A_p \leq c.$$

Правило R1 утверждает, что сумма единиц выделенных ресурсов для всех процессов из P должна быть меньше системной границы c. Стратегии, нарушающие это правило, Миллен называет неправдоподобными, поскольку невозможно выделить ресурсов больше, чем имеется в наличии.

$$(R2)\ \text{if running}(p)\ \text{then}\ {}^{S}Q_p = 0.$$

Правило R2 утверждает, что текущие процессы должны иметь нулевые пространственные требования. Миллен аргументирует это тем, что если процесс не обладает всеми ресурсами, которые необходимы ему в некотором состоянии, то не имеет смысла продолжать этот процесс, пока требования не будут выполнены. “Зависание” происходит, когда процесс имеет ненулевые пространственные требования, которые никогда не будут выполнены (или будут выполнены поздно).

$$(R3)\ \text{if running}(p)\ \text{and running}(p)'\ \text{then}\ A_p' = A_p.$$

Конструкция “running(p) and running(p)’ ” в правиле R3 означает, что в некотором состоянии процесс p является текущим и остается текущим и в следующем состоянии. Это правило утверждает, что для текущих процессов распределение ресурсов не меняется. Это действительно мощное предположение, поскольку из него следует, что во время выполнения текущие процессы не выгружаются в случае, если произойдет какое-либо перераспределение ресурсов (в отличие от перераспределения ресурсов ЦПУ).

$$(R4)\ \text{if}\ A_p(CPU)' = A_p(CPU)\ \text{then}\ T(p)' = T(p).$$

Правило R4 утверждает, что часы процесса изменяются только с изменением распределения ЦПУ. Предполагается, что единицы времени - это положительные целые числа, всегда возрастающие с изменением времени (как будет определено ниже).

$$(R5)\ \text{if}\ A_p(CPU)' \neq A_p(CPU)\ \text{then}\ T(p)' > T(p).$$

Правило R5 утверждает, что часы изменяются только для того, чтобы отразить увеличение во времени. Заметим, что у каждого процесса имеются

свои собственные часы, а синхронизации различных часов друг с другом или с какими-либо часами, показывающими реальное время, не предпринимается.

(R6) if asleep(p) then ${}^{S}Q_p' = {}^{S}Q_p + A_p - A_p'$.

Правило R6 утверждает, что пространственные требования устанавливаются для застывших процессов. Другими словами, если процесс не является текущим, он должен определить те ресурсы, которые ему потребуются для выполнения задания. Как только все эти ресурсы получены, процесс возобновляется и становится текущим.

(R7) if asleep(p) then ${}^{T}Q_p' = {}^{T}Q_p$.

Правило R7 утверждает, что для застывших процессов временные требования не устанавливаются, то есть тогда как пространственные требования для застывших процессов устанавливаются, временные требования для таких процессов не устанавливаются.

(R8) if running(p) and asleep(p)' then $A_p' = A_p$ - CPU.

Правило R8 утверждает, что переходы, в результате которых процесс останавливается, перераспределяют только ресурсы ЦПУ. Другие изменения в распределении должны происходить после возобновления процесса.

Миллен использует данную модель распределения ресурсов для описания некоторых стратегий. Например, применяя MPP, можно выразить стратегию конечного времени ожидания (FWT). Для указания интервалов, которые могут возникнуть после многократных переходов, мы используем оператор временной логики *leads_to* (то есть A *leads_to* B означает, что во всех последующих состояниях из A в конце концов следует B). Для указания интервалов, которые могут возникнуть после многократных переходов, можно выразить максимальное время ожидания (MWT) аналогичным образом.

FWT: " p, s : ∃ s' : s'(running(p)) and s *leads_to* s'.

В выражении, приведенном выше, s(x) означает, что x истинно для состояния s, а s *leads_to* s' означает, что " s, s' : s((T(p) =n)) и s'((T(p) =m)) и m>n. FWT утверждает, что пользователи в конце концов получат запрашиваемые ресурсы, то есть они в конечном итоге получат ЦПУ для

выполнения работы. Аналогичным образом можно выразить максимальное время ожидания (MWT).

$$MWT: \exists\, b : \forall p, s : \exists\, s' : s'(\text{running}(p)) \text{ and } s\ leads\_to(b)\ s'.$$

В этом выражении s *leads_to*(b) s' означает, что ∀s, s' : s((T(p) =n)) и s'((T(p) =m)) и m-n ≤ b. MWT отличается от FWT тем, что здесь накладывается ограничение на время ожидания пользователями возможности продвижения в выполнении задания.

ЛИТЕРАТУРА

1. Грушо А.А., Тимнонина Е.Е. Теоретические основы защиты информации. - М.: Яхтсмен, 1996. - 304 с.
2. Теория и практика обеспечения информационной безопасности / Под ред. П.Д. Зегжды.- М.: Яхтсмен, 1996. - 192 с.
3. Хоффман Дж. Современные методы защиты информации. - М.: Сов. радио, 1980.
4. Bell L. LaPadula. Secure Computer System: Mathematical Foundation, ESD-TR-73-278, V 1, MITRE Corportation.
5. LaPadula D. Bell. Secure Computer Systems: A Mathematical Model, ESD-TR-73-278, V. II, MITRE Corportaion.
6. Landwehr. Formal Models for Computer Security. ACM Computing Surveys. Vol. 13, N 3. 1984.
7. Clark D. Wilson. A Comparison of Commercial and Military Computer Securitry Policies. Proceedings of the IEEE Symposium on Security and Privacy.
8. Vijay Varadharajan "A multilevel security policy for networks", 1990.
9. John McLean “A comment on the “Basic Security Teorem” of Bell and La Padula”, Information Processing Letters, 1985.
10. John McLean “The specfcation and modeling of computer security”, Computer, 1990.
11. John McLean “Security models”, Enciclopedia of software engeneering, 1994.
12. John McLean “Securty models and information flow”, IEEE simposium on research in security and privacy, 1990.

13. Carl E. Lendwehr, Constanse L.Heitmeyer, John McLean, “A security models for military message system”, ACM transactions of computer systems, 1984.
14. M.Harrison, W.Ruzzo, J.Uhlman “Protection in operating systems”, Communications of the ACM, 1976.
15. M.Harrison, W.Ruzzo “Monotonic protection systems”, Foundation of secure computation, 1978.
16. J.Millen “Resource allocation model for denial of service”, IEEE simposium on research in security and privacy, 1992.
17. C-F Yu, V.Gligor “A specification and verification method for preventing denial of service”, IEEE transaction on software engeneering, 1990.

Модель Харрисона, Руззо Ульмана

Обозначения, используемые в данных теоремах, введены в параграфе 2.1.1.1.

Т е о р е м а 1 . Существует алгоритм для определения, является или нет моно-операционная система безопасной для данного права a.

Доказательство.

Доказательство строится на том факте, что для любой последовательности запросов r_1, r_2,..., r_n, которая приводит к утечке права a из начальной конфигурации (S_0, O_0, M_0), существует последовательность запросов из данной конфигурации, которая также ведет к утечке права a, которая содержит только запрос enter**,** за исключением начального запроса create subject. Для того, чтобы сформировать данную последовательность, необходимо вставить запрос create subject в ее начало r_1, r_2,..., r_n для создания нового субъекта. Обозначим данный запрос как s_{init}. Далее удалим все запросы delete и destroy из последовательности r_1, r_2,..., r_n. Данная последовательность все равно будет приводить к утечке права a, так как условная часть запроса может протестировать только присутствие права, но не отсутствие прав. Далее рассмотрим самый правый в последовательности запрос create subject. Можно удалить данный запрос и просто заменить все ссылки на нового субъекта в дальнейших запросах ссылкой на s_{init}. Продолжим данную процедуру до тех пор, пока не достигнем начального запроса create subject, который останется без изменений. Далее заменяются все запросы create object к системе с заменой

ссылок на новые объекты в дальнейших запросах ссылками на s_{init}. После этого из последовательности удаляются запросы enter, которые вводят право a_i в ячейку, которая уже содержит это право.

Результирующая последовательность приводит также к утечке права a, но имеет длину $l = (|A| * (|S_0| + 1) * (|O_0| + 1)) + 1$ запросов, так как каждый запрос, за исключением начального запроса create object**,** должен вводить новый символ из A в ячейку матрицы доступа, и количество ячеек в матрице не может быть больше, чем $(|S_0| + 1) * (|O_0| + 1)$. Следовательно, можно определить, является ли данная система безопасной просмотром всех возможных последовательностей запросов enter длиной меньше или равной чем l.

Т е о р е м а 2. Проблема определения безопасности для данного права a в системе с запросами общего вида является неразрешимой.

Доказательство.

Доказательство данной теоремы основывается на том факте, что описанная нами модель достаточно выразительна для того, чтобы быть промоделированной с помощью формализма машины Тьюринга.

Рассмотрим машину Тьюринга. Каждая машина Тьюринга T имеет фиксированное число состояний K и конечный набор символов Г, размещаемых на ленте. Одним из этих символов является пробел B, который первоначально появляется в каждой ячейке ленты. Лента бесконечна только вправо. Машина Тьюринга снабжена сканирующей головкой, которая в каждый момент времени обозревает некоторую ячейку на ленте. Действия машины Тьюринга определяются функцией δ с аргументом из K×Г и значением K×Г×(L, R) (L означает переход головки влево, R - вправо). Если δ (q, X) = (p, Y, L) для состояний p и q и символов на ленте X и Y, то это означает, что машина T, находящаяся в состоянии q и наблюдающая ячейку, содержащую символ X, переходит в состояние p, стирает символ X и печатает символ Y в эту ячейку, сканирующая головка переходит при этом на одну клетку влево.

Первоначально T находится в состоянии q_0 - исходном состоянии и обозревает ячейку 1. Каждая ячейка первоначально содержит пробел. Существует особое состояние q, известное как конечное. В соответствии с тезисом Черча, алгоритм, вычислимый на машине Тьюринга, вычислим и в

интуитивном понятии. Таким образом, наша проблема сводится к проблеме останова машины Тьюринга.

Покажем, что вопрос обеспечения безопасности является неразрешимым. Это можно показать на примере системы защиты, которую можно смоделировать произвольной машиной Тьюринга.

Набор общих прав рассматриваемой системы защиты включит состояния и символы машины Тьюринга. В любой момент времени машина Тьюринга будет иметь некоторый ограниченный начальный набор ячеек ленты, скажем 1, 2, ..., k, который она никогда не обозревала. Эту ситуацию можно представить последовательностью k субъектов s_1, s_2, ..., s_k, так что s_i "владеет" s_{i+1} для $1 \le i \le k$. Таким образом, мы используем отношение собственности для упорядочивания субъектов в линейном списке, представляющем собой ленту машины Тьюринга. Субъект s соответствует ячейке i, а что ячейка i сейчас содержит символ ленты X, представляется присвоением s_i общего права q. Предполагается, что мы рассматриваем состояния отдельно от символов ленты, но это не должно привести к неправильному результату. Существует специальный конец общего права, который приписывается последнему субъекту s_k, s_k имеет конец общего права по отношению к самому себе, показывая, что мы еще не создали субъект s_{k+1}, которым должен владеть s_k. Общее право "собственность" завершает набор общих прав.

Действия машины Тьюринга отражаются в командах следующим образом. Во-первых, если $\delta(q, X) = (p, Y, L)$, то существует команда $C_{qx}(s, s')$ с условиями:

собственность $\in (s, s')$,

$c \in (s', s')$, $x \in (s', s')$.

s и s' должны соответствовать двум последовательным ячейкам на ленте; машина находится в состоянии q, наблюдает ячейку, соответствующую s', и в этой ячейке записан символ X.

Команда C_{qx} интерпретируется следующим образом:

удалить q из (s', s'),

удалить X из (s', s'),

ввести p в (s, s),

ввести Y в (s', s').

Если $\delta(q, X) = (p, Y, R)$, то есть головка ленты перемещается вправо, то мы имеем две команды в зависимости от того, прошла ли головка текущий конец ленты, то есть конец права. Существует команда $C_{qx}(s, s')$ с условиями:

собственность $\in(s, s')$,

$q \in (s, s)$, $X \in (s, s)$

и интерпретация:

удалить q из (s, s),

удалить X из (s, s),

ввести p в (s', s'),

ввести Y в (s, s).

Существует также команда Dqx(s, s') с условиями:

конец $\in(s, s)$,

$q \in (s, s)$, $X \in (s, s)$

и интерпретация:

удалить q из (s, s),

удалить X из (s, s),

создать субъект s',

ввести B в (s, s'),

ввести p в (s', s'),

ввести Y в (s, s).

Если мы начинаем с исходной матрицы, содержащей субъект s_1 с правами q_0, B (пробел) и "собственность" по отношению к самому себе, то матрица доступа будет всегда иметь ровно одно общее право, которое является состоянием системы. Это следует из того, что каждая команда удаляет состояние, известное из условия той команды, которая должна выполняться. Каждая команда также включает одно состояние в матрицу. Нельзя вносить в матрицу более одного общего права, которое является символом ленты по такому же аргументу. Таким же образом конец появляется только в одной записи матрицы, а именно в диагональной записи для каждого созданного субъекта.

Таким образом, в каждой конфигурации системы защиты, полученной из исходной конфигурации, существует хотя бы одна команда, которую можно применить. Это объясняется тем, что машина Тьюринга имеет по крайней мере

одно допустимое перемещение в любой ситуации. C_{qx} и D_{qx} никогда не используются одновременно. Система защиты должна поэтому точно моделировать машину Тьюринга, используя описанное представление. Если машина Тьюринга вводит состояние q_j, то система защиты может иметь утечку общего права q_j, в противном случае она безопасна для q_j. Поскольку нельзя предсказать, будет ли машина Тьюринга вводить q_j, нельзя решить, является ли система защиты безопасной для q_j.

ПРИЛОЖЕНИЕ 3

Формальная модель MMS

Разработаем формальную модель MMS, соответствующую неформальным спецификациям, данным выше. Формальное описание приведено по следующим соображениям:

1. Показать, как неформальная модель требований безопасности может быть превращена в формальную.

2. Данная абстрактная модель может быть интерпретирована для доказательства безопасности других систем.

3. Формальная модель является базисом для спецификации и реализации системы.

Предположения

Мы предполагаем существование множества возможных пользователей и множества возможных сущностей. С помощью них мы определяем состояние системы и безопасное состояние. Далее мы определяем систему и ее историю и вводим ограничения на переход из одного состояния в другое. Система, все переходы которой удовлетворяют данным ограничениям, безопасна для переходов. В заключение определяется безопасная история и безопасность системы.

Структура формальной модели спроектирована для упрощения определения предусловий и постусловий для операций системы.

Состояние системы

Определим состояние системы и то, что необходимо для того, чтобы система была безопасной.

OP - множество операций.

L - множество уровней безопасности; ≥ - частичный порядок на L такой, что (L, ≥) - решетка.

UI - множество идентификаторов пользователей.

RL - множество ролей пользователей.

US - множество пользователей. $\forall u \in US$, $CU(u) \in L$ - степень доверия к пользователю u, $R(u) \subseteq RL$ - множество ролей, для которых авторизован пользователь u и $RO(u) \subseteq RL$ - текущая роль пользователя u.

RF - множество ссылок. Данное множество подразделяется на множества DR - прямых ссылок и IR - множество косвенных ссылок. Хотя истинная природа ссылок неважна, мы предполагаем, что прямые ссылки могут быть пронумерованы целыми числами. В модели мы рассматриваем каждую ссылку как последовательность, состоящую из простых чисел; для прямых ссылок - одно число, например <17>, для косвенных - конечная последовательность из двух или более целых чисел, например $<n_1,...,n_m>$, где n_1 - прямая ссылка.

VS - множество строк (символьных или битовых). Данные строки обозначают значение сущности (например содержимое файла).

TY - множество типов данных в системе, включающие "DM" для чертежных сообщений и "RM" для освобожденных сообщений.

ES - множество сущностей системы. $\forall e \in ES\ CE(e) \in L$ - классификация e.

$AS(e) \subseteq (UI \cup RL) \times OP \times N$ - множество троек, которые составляют множество доступа к e. (u, op, k) $\in$ AS(e), если u - идентификатор пользователя или его роль, в которой он авторизован для того, чтобы исполнить операцию op со ссылкой к e как k-параметру операции op. $T(e) \in TY$ - тип сущности e. $V(e) \in$ VS - значение сущности e. Если T(e) = DM или T(e) = RM, то V(e) включает поле освобождения RE(e), которое, если оно не пусто, содержит идентификатор пользователя.

ES содержит подмножество контейнеров. Для любой сущности e в данном множестве контейнеров $H(e) = <e_1,...,e_n>$, где e_i - сущность, содержащаяся в e. CCR(e) - истинно, если e помечено CCR, иначе - ложно. Если $T(e_1) = T(e_2)$, то e_1 и e_2 - оба контейнеры или объекты.

Множество O - устройств вывода - подмножество множества контейнеров. Элемент $o \in O$ рассматривается как домен из двух функций, рассматриваемых далее. D(o) - множество упорядоченных пар $[(x_1,y_1), (x_2,y_2),..., (x_n,y_n)]$, где:

— каждое y_i отображается на o;

— каждое x_i - пользователь или сущность и соответствующее y_i - ссылка, пользовательский идентификатор или результат применения описанных выше функций к x_i. Требуется, чтобы $(x,V(x)) \in D(o) \rightarrow x \in H(o)$.

CD(o) дает максимум классификации информации, которая может быть отображена на o. Это позволяет использовать CE(o) как текущую верхнюю границу классификации для информации, которая может быть отображена на устройстве вывода так, что пользователи могут ограничивать классификацию вывода до уровня, меньшего чем максимально разрешенный.

Состояние отображает подмножество пользовательских идентификаторов и ссылок в элементы US и ES, которые представляют соответствующие свойства. Состояние также отображает подмножество пользовательских идентификаторов, которые "существуют" в ссылках к устройствам вывода. Определим три отображения:.

— id функция U - отображение один к одному от подмножества UI к подмножеству US.

— Функция ссылок E - отображение подмножества RF к ES так, что $\forall n \geq 2$ $E(<i_1,...,i_n>) = e$, если $E(<i_1,...,i_{n-1}>) = e^*$, где e^* - контейнер, так что e - i_n элемент $H(e^*)$. Для некоторой ссылки r, если E(r)=e, мы говорим, что r - ссылка к e (относительно E).

— login функция, LO, - отображение один к одному из подмножества UI в RF.

В соответствии со ссылочной функцией E, каждая ссылка в форме $<n_0,...,n_m>$ к сущности e соответствует пути из сущностей $<e_0,...,e_m>$ таких, что каждая $e_i \in rng(E)$, e_0 - прямая ссылка к $<n_0>$, и для всех положительных целых i, $m \geq i$, e_i - n_i сущность в контейнере e_{i-1}. О таких косвенных ссылках будем говорить, что они базируются на сущности e_j, $m > j \geq 0$.

Определение 1. Состояние системы, s - упорядоченная тройка (U, E, LO), где U - id функция, E - функция ссылок, LO - login функция такие, что $dom(LO) \subseteq dom(U)$ и $rng(LO) \subseteq dom(E \cap (RF \times O)$. Мы также требуем, чтобы если $o \in rng(E) \cap O$ и $(x,y) \in D(o)$, то $x \in rng(E) \cup rng(U)$ для гарантии того, что только информация о пользователях или сущностях, которая существует в данном состоянии может быть просмотрена для некоторой ссылки r, $(x,r) \in D(o) \rightarrow E(r)=x$. В заключение требуется $E(LO(u_1))=E(LO(u_2)) \rightarrow u_1=u_2$ для предотвращения вхождения двух пользователей с одного терминала.

С помощью системного состояния s=(U, E, LO) сократим E(r) как r_s, U(u) как u_s и E(LO(u)) как u^s.

Определение 2. Состояние s безопасно, если $\forall$ x,y $\in$ rng(E), $\forall$o $\in$ O$\cap$rng(E), $\forall$w$\in$dom(LO),и $\forall$u $\in$ rng(U):

$x \in H(y) \rightarrow CE(y) \geq CE(x)$,

$x \in H(w^s) \rightarrow CE(w_s) \geq CE(x)$,

$(x, V(x)) \in D(o) \rightarrow (x, CE(x)) \in D(o)$,

$RO(u) \subseteq R(u)$ и

$CD(o) \geq CE(o)$.

Безопасность системы

Определим, что такое система и что нужно для того, чтобы система была безопасна.

Определение 3. Система Σ - четверка (I, S, s_0, T), где

I - множество правильно сформированных системных запросов, в котором $\forall i \in I$ - форма $<op,x_1, x_2,... x_n>$, $x_j \in RF \cup UI \cup VS$ и $op \in OP$;

S - множество возможных состояний системы;

s_0 - отмечает специальное состояние, называемое начальным;

T - переход системы, функция из UI × I × S в S.

Определение 4. История Π - функция из множества положительных целых N в UI × I × S такая, что третий элемент Π s_0 и $\forall n \in N$, если $\Pi(n) = (u,i,s)$ и $\Pi(n+1) = (u^*, i^*, s^*)$, то $T(u, i, s) = s^*$.

До определения операции, возможно модифицирующей сущность, отметим, что ссылочная функция E и состояние s индуцируют множество функций на ссылках, которые являются двойниками ко множеству функций, определенных выше на сущностях. Например, существует функция V_s такая, что $V_s(r) = V(r_s)$. Также определим предикат H_s такой, что $H_s<r, r^1,...,r^n>$, если $H(r_s) = <r_{s1},...,r_{sn}>$. Каждый двойник является видимой для пользователя соответствующей функции сущностей. Мы называем их ссылочные двойники и используем для определения того, что означает для двух состояний быть эквивалентными, исключая множество ссылок.

Состояния $s = (U, E, LO)$ и $s^* = (U^*, E^*, LO^*)$ эквивалентны, исключая некоторое множество ссылок ρ, если:

1) $U=U^*$,

2) $LO=LO^*$,

3) dom(E)=dom(E*),

4) для некоторой функции сущности F, исключая V, $F=F_s^*$, и

5) для некоторой ссылки $r \in dom(E) \sim \rho$, $V_s(r)=V_s^*(r)$.

Определим потенциальную модификацию : u, i, s потенциально модифицируют r, если $\exists s_1 s_1^*$:s_1 эквивалентно s, исключая возможное множество ссылок и $T(u,i,s_1)=s_1^*$ и для некоторой функции сущностей F, $F(r_{s1}) \neq F(r_{s1}^*)$.

Назовем y вкладывающим фактором в том случае, если y=r или $\exists s_1$, в том же смысле, что и выше и s_2, s_2^*: s_1 и s_2 эквивалентны, исключая [y] и $T(u, i, s_2)=s_2^*$ и $F(r_{s2}^*) \neq F(r_{s1}^*)$.

То есть u, i, s потенциально модифицируют r, если существует некоторое (второе) состояние, которое может отличаться от s значениями некоторых сущностей, и T отображает u, i и это состояние в третье состояние, в котором некоторая функция сущности F (значение, контейнер, множество доступа и т.д.) на r отличается от второго состояния. Вкладывающими факторами являются r и те сущности, чьи значения воздействуют на конечное F(r).

Для каждого соответствующего двойника и каждой функции, определенной на пользователях, определим уникальную операцию, которая изменяет существующую сущность или пользователя в соответствии с этой функцией. Например, операция set_AS(r, new_access_set) есть только операция (кроме delete(r) и возможно create(r)), которая изменяет множество доступа r и не имеет другого видимого для пользователя эффекта. Если переход из состояния s в состояние s^*, $AS_s^*(r)$ есть new_access_set, если new_access_set - строка символов и V_s(new_access_set), если new_access_set - ссылка на сущность. Изменение областей определения E или U (создание или уничтожение сущностей или ссылок) также предполагается происходящим при определенных запросах. Формальное определение операции освобождения, определенное ниже - единственное исключение в этом предположении; оно изменяет тип r и, возможно, поле освободителя r.

Истинная природа этих операций неважна, так как предположения включены единственно для простоты представления. Их назначение не управлять реализацией команд, которые изменяют различные части сущностей, а устранять проблему неспецифицированных эффектов в формальной модели (например разрешение просматривать сообщения, помеченные CCR не есть

разрешение очищать отметку CCR). Реализация команд, изменяющих более чем одну часть отдельной сущности, соответствует последовательности формальных операций. Для данной реализации это соответствие определяется семантикой реализации командного языка. Для однажды определенного соответствия такого, что относящиеся к безопасности эффекты для каждой пользовательской команды ясны, можно изменить множество команд реализации с соответствующе измененным множеством доступа. Несмотря на это, благоразумие диктует то, что модификация, которая может быть сделана только офицером безопасности (изменяющим степень доверия к пользователю), должна быть ограничена так, чтобы в любой реализации существовала единственная команда, выполняющая эту операцию.

Следующие ограничения на переходы системы ведут к определению истории безопасности и безопасности системы. В нижеследующих выражениях там, где не отмечена квантификация, предполагается квантор универсальности.

Определение 5. Переход системы T - безопасен по доступу, если $\forall u, i, s, s^*$, $[(op \in i \cap OP$ и $r_k \in i \cap RF) \rightarrow ((u,op,k) \in AS(E(r_k))$ или $\exists l \in RO(u_s)$ и $(l,op,k) \in AS(E(r_k)))]$ или $s=s^*$.

Определение 6. Переход T безопасен по копированию, если $\forall u, i, s, s^*$: $T(u,i,s)=s^*$ x - потенциально модифицируется через вкладывающий фактор $y \rightarrow CE(x_s) \geq CE(y_s)$.

Определение 7. Переход T безопасен по CCR $\forall u, i, s, s^*$:$T(u,i,s)=s^*$,

$r \in i \cap IR$ базируется на y и CCR(y) и z потенциально модифицируется через вкладывающий фактор $r \rightarrow CE(u_s) \geq CE(y)$.

Определение 8. Переход T безопасен по трансляции, если $\forall u, i, s, s^*$: $T(u,i,s)=s^*$, $x \in DR$ и $(x_s^*, x) \in D(u_s^*) \rightarrow \exists r \in i \cap RF$, $r_s=x_s$ и (r базируется на z и $\rightarrow CE(u_s) \geq CE(z)$).

Определение 9. Переход T безопасен для множества, если $\forall u, i, s, s^*$: $T(u,i,s)=s^*$,

(a) $\exists o \in dom(E \cap (RF \times O))$, $CD(o_s) \neq CD(o_s^*)$ или $\exists x \in dom(U)$, $CU(x_s) \neq CU(x_s^*)$ или $R(x_s) \neq R(x_s^*)$,

(b) $\exists x \in dom(U)$ и $R(x_s) \neq R(x_s^*) \rightarrow u_s=x_s$ или $security\_officer \in RO(u_s)$.

Определение 10. Переход T безопасен по снижению уровня, если $\forall u, i, s, s^*$: $T(u,i,s)=s^*$, $x \in dom(E \sim (RF \times \{u_s\}))$ и $CE(x_s) > CE(x_s^*) \rightarrow downgrader \in RO(u_s)$.

Определение 11. Переход T безопасен по освобождению, если $\forall u, i, s, s^*$: $T(u,i,s)=s^*$, $(T(x_s)=RM \rightarrow T(x_s^*)=RM$ и $RE(x_s)=RE(x_s^*))$

и $(T(x_s) \neq RM$ и $T(x_s^*)=RM \rightarrow RE(x_s^*)=u$, $\exists r: r_s=x_s$, i - операция *освободить* <*releaser , r*>, *releaser*$\in RO(u_s)$ и $T(x_s)=DM)$.

Определение 12. Переход является безопасным, если он безопасен по доступу, копированию, по CCR, трансляции, множеству, снижению уровня и освобождению.

Определение 13. История безопасна, если все ее состояния и переходы безопасны.

Определение 14. Система безопасна, если все ее истории безопасны.

Обсуждение

Основной идеей проведенной формализации является взгляд на компьютерную систему как на взаимоотношения между состояниями системы и системой. В данных обсуждениях состояние системы состоит из сущностей и их отношений, и система добавляет к данным отношениям пользователей и пользовательские операции над сущностями. Следовательно, все ограничения на свойства пользователей (в частности, ограничение, что для любого u $RO(u) \subseteq R(u)$) включены в определение безопасности системы. Данный взгляд разделяет состояние системы и систему в терминах статики в противоположность динамическим свойствам. Статические свойства - свойства, которые сохраняются для всех состояний системы, и, следовательно, могут быть проверены для изолированного состояния системы; динамические состояния - те, которые нуждаются в исследовании взаимоотношений между состояниями безопасности, и, следовательно, могут быть проверены только исследованием двух или более состояний. В определение безопасности состояния включены только статические свойства.

Принципиальной трудностью, возникающей при формализации модели, является представление "копирования", "просмотра", вывода системы и авторизованной операции. Ограничение 3 (изменения в объекте) в неформальной модели требует формальной семантики, отражающей перемещение информации между сущностями, в то время как ограничение 4

(просмотр) требует формальной семантики, отражающей видимость сущности пользователю. Ограничение 5 (доступ к CCR сущностям) опирается на просмотр и копирование. Семантика для копирования, включенная в определения "потенциальной модификации" и "вкладывающего фактора", базируется на граничной интерпретации копирования. Информация считается копируемой не только тогда, когда она непосредственно переносится из одной сущности в другую, но и когда она дает потенциальный вклад в другую сущность. Например, если операция сканирует файл сообщений А и копирует сообщения, выбранные фильтром Ф в файл сообщений Б, то и А, и Ф являются потенциальным вкладом в модификацию Б (и, следовательно, субъектом для ограничений, вызванных безопасностью копирования и CCR безопасностью), даже если и А, и Ф - пусты. Семантика для просмотра - проста: сущность может быть просмотрена, если операция делает ее членом выходного контейнера. В свете этих предположений в ограничении 5 используется доступ вместо просмотра.

В формализации вывод системы интерпретируется как множество контейнеров; другие сущности, части сущностей, ссылки и классификации, которые видны пользователю, интерпретируются как копирующиеся в контейнер выхода. Ссылки ясно включены как часть выхода, так как одинаковые операции, примененные к одинаковым сущностям, могут вызвать разные результаты, в зависимости от того, как они выводятся: напрямую или косвенно. Для реализации этого ограничения система должна распознавать ссылки как особую часть вывода.

Формализация концепции авторизованного вывода трудна, так как семантика авторизованных операций неспецифицирована. Определение безопасности доступа требует того, что если операция изменяет состояние системы(кроме вывода сообщения об ошибке), то для каждой сущности во множестве операций пользователь или роль, операция и индекс операнда присутствовали во множестве доступа. Неавторизованные операции не должны изменять состояние системы, исключая сообщения об ошибке.

Соответствие неформальной модели

Ограничения (2), (4) и (7) неформальной модели, относящиеся к иерархии классификаций, просмотру и меткам, включены в формальное

определение безопасного состояния. Они относятся соответственно к первым трем условиям, которым должно удовлетворять безопасное состояние; последние два условия требуют того, чтобы текущее множество ролей пользователя было подмножеством авторизованного множества ролей и чтобы текущий уровень каждого устройства вывода доминировал над максимально разрешенным уровнем. Данные два условия подразумеваются в неформальной модели.

Остальные ограничения неформальной модели переводятся в ограничения на переход системы. Ограничение (1) (авторизация) соответствует непосредственно безопасности доступа, ограничение (6) - к безопасности трансляции, и ограничения (8) - (10) соответствуют безопасности множества, безопасности понижения уровня и безопасности освобождения.

Ограничения (3) и (5) соответствуют безопасности копирования и безопасности CCR. Определение безопасности копирования перекрывает ограничения 3 и 4, так как устройства вывода рассматриваются как контейнеры. CCR безопасность соответствует ограничению 5.

Основная теорема безопасности для формальной модели MMS

Т е о р е м а 1. Любое состояние системы $\sum$ безопасно, если безопасно s_0 и T удовлетворяет следующим условиям для всех u, i, s, s^*: $T(u, i, s)=s^*$ и для всех $x,y \in RF$, $w \in US$:

1. $x_s \notin H(y_s)$ и $x_s^* \in H(y_s^*) \rightarrow CE(y_s^*) \geq CE(x_s^*)$.
2. $x_s \in H(y_s)$ и $CE(y_s^*) < CE(x_s^*) \rightarrow x_s \notin H(y_s^*)$.
3. $x_s \notin H(w_s)$ и $x_s^* \in H(w_s^*) \rightarrow CU(y_s^*) \geq CE(x_s^*)$.
4. $x_s \in H(w_s)$ и $CU(w_s^*) < CE(x_s^*) \rightarrow x_s \notin H(w_s^*)$.
5. $(x_s, V(x_s)) \notin w_s$ и $(x_s^*, V(x_s^*)) \in w_s^* \rightarrow (x_s^*, CE(x_s^*)) \in w_s^*$.
6. $(x_s, V(x_s)) \in w_s$ и $(x_s^*, CE(x_s^*)) \notin w_s^* \rightarrow (x_s^*, V(x_s^*)) \in w_s^*$.
7. $R(w_s) \neq R(w_s^*)$ или $RO(w_s) \neq RO(w_s^*) \rightarrow RO(w_s^*) \subseteq R(w_s^*)$.
8. $CE(w_s) \neq CE(w_s^*)$ или $CD(w_s) \neq CD(w_s^*) \rightarrow CD(w_s^*) \geq CE(w_s^*)$.

Совместно условия (1) - (8) - необходимые и достаточные условия для того, чтобы система была безопасна в любом состоянии, достижимом из s_0.